Proyecto UCT - DJANGO

Presentado por:

María Paula Buenaventura García

Análisis y desarrollo de software, SENA

Instructor:

Juan Guillermo Zuluaga

Ibagué, Tolima

2024

Tabla de contenidos

Introducción

La universidad Ceicotol Tolima (UCT), es una universidad prestigiosa, en la que se ofrecen diversos pregrados y postgrados en diversas áreas y campos de estudio. La universidad, actualmente está pasando una dificultad en cuanto a su sistema de gestión académica, ya que el que tienen actualmente está obsoleto y carece de capacidad de escalabilidad.

El objetivo de este proyecto, es implementar el nuevo sistema de gestión académica que tendrá la universidad UCT, el cual tendrá como nombre “UCT Academic Manager”, este sistema permitirá la gestión eficiente de la información sobre todos los procesos académicos de la universidad tanto para profesores, estudiantes y funcionarios.

Rutas

Generales

```
path('', GestionCrud.views.index, name='index'),
path('registrarse/', GestionCrud.views.paginaRegistro, name='registrarse'),
path('login/', GestionCrud.views.paginaLogin, name='login'),
path('logout/', GestionCrud.views.paginaLogout, name='logout'),
path('quienesSomos/', GestionCrud.views.quienesSomos, name='quienesSomos'),
path('contactos/', GestionCrud.views.contactos, name='contactos'),
path('academicManager/', GestionCrud.views.academicManager, name='academicManager'),
```

Clase Estudiante

```
#?Rutas clase Estudiante
path('estudiantes/', GestionCrud.views.mostrarEstudiantes, name='mostrarEstudiantes'),
path('estudiantes/<int:id>/', GestionCrud.views.mostrarEstudiante, name='mostrarEstudiante'),
path('estudiantes/añadir/', GestionCrud.views.añadirEstudiante, name='añadirEstudiante'),
path('estudiantes/modificar/<int:id>/', GestionCrud.views.modificarEstudiante, name='modificarEstudiante'),
path('estudiantes/eliminar/<int:id>/', GestionCrud.views.eliminarEstudiante, name='eliminarEstudiante'),
```

Clase Profesor

```
#?Rutas clase Profesor
path('profesores/', GestionCrud.views.mostrarProfesores, name='mostrarProfesores'),
path('profesores/<int:id>/', GestionCrud.views.mostrarProfesor, name='mostrarProfesor'),
path('profesores/añadir/', GestionCrud.views.añadirProfesor, name='añadirProfesor'),
path('profesores/modificar/<int:id>/', GestionCrud.views.modificarProfesor, name='modificarProfesor'),
path('profesores/eliminar/<int:id>/', GestionCrud.views.eliminarProfesor, name='eliminarProfesor'),
```

Clase Carrera

```
#?Rutas clase Carrera
path('carreras/', GestionCrud.views.mostrarCarreras, name='mostrarCarreras'),
path('carreras/<int:id>/', GestionCrud.views.mostrarCarrera, name='mostrarCarrera'),
path('carreras/añadir/', GestionCrud.views.añadirCarrera, name='añadirCarrera'),
path('carreras/modificar/<int:id>/', GestionCrud.views.modificarCarrera, name='modificarCarrera'),
path('carreras/eliminar/<int:id>/', GestionCrud.views.eliminarCarrera, name='eliminarCarrera'),
```

Clase Materia

```
#?Rutas clase Materia
path('materias/', GestionCrud.views.mostrarMaterias, name='mostrarMaterias'),
path('materias/<int:id>/', GestionCrud.views.mostrarMateria, name='mostrarMateria'),
path('materias/añadir/', GestionCrud.views.añadirMateria, name='añadirMateria'),
path('materias/modificar/<int:id>/', GestionCrud.views.modificarMateria, name='modificarMateria'),
path('materias/eliminar/<int:id>/', GestionCrud.views.eliminarMateria, name='eliminarMateria'),
```

Vistas

Generales

```
def index(request):
    return render(request, "inicio.html")

def quienesSomos(request):
    return render(request, "quienesSomos.html")

def contactos(request):
    return render(request, "contactos.html")

def academicManager(request):
    return render(request, "academicManager.html")
```

```python
def paginaRegistro(request):
    if request.user.is_authenticated:
        return redirect('index')

    else:
        formRegistro = FormRegistro()

        if request.method == 'POST':
            formRegistro = FormRegistro(request.POST)

            if formRegistro.is_valid():
                formRegistro.save()
                messages.success(request, f'El usuario se ha registrado con exito')
                return redirect('index')

    return render(request, 'users/registro.html', {
        'formRegistro' : formRegistro
        })
```

```python
def paginaLogin(request):
    if request.user.is_authenticated:
        return redirect('index')

    else:
        if request.method == "POST":
            username = request.POST.get("username")
            password = request.POST.get("password")

            user = authenticate(request, username=username, password=password)

            if user is not None:
                login(request, user)
                return redirect("index")

            else:
                messages.warning(request, "El Usuario o Contraseña son incorrectas")

    return render(request, 'users/login.html')


def paginaLogout(request):
    logout(request)
    return redirect('index')
```

Clase Estudiante

```python
def mostrarEstudiantes(request):
    estudiantes = Estudiante.objects.raw("SELECT id, estNombre, estApellido, estCarrera FROM GestionCrud_estudiante")

    return render(request, 'estudiante.html', {
        'estudiantes' : estudiantes,
    })
```

```python
def mostrarEstudiante(request, id):
    estudiante = Estudiante.objects.get(pk = id)

    return render(request, 'estudianteDetalles.html', {
        'estudiante' : estudiante,
    })
```

```python
def añadirEstudiante(request):
    formulario = FormEstudiante()

    if request.method == 'POST':
        formulario = FormEstudiante(request.POST, request.FILES)

        if formulario.is_valid():
            data_form = formulario.cleaned_data

            estNombre = data_form.get('nombre')
            estApellido = data_form.get('apellido')
            estEmail = data_form.get('email')
            estTelefono = data_form.get('telefono')
            estFechaNacimiento = data_form.get('fechaNacimiento')
            estFoto = data_form.get('foto')
            estCarrera = data_form.get('carrera')

            estudiante = Estudiante(
                estNombre = estNombre,
                estApellido = estApellido,
                estEmail = estEmail,
                estTelefono = estTelefono,
                estFechaNacimiento = estFechaNacimiento,
                estFoto = estFoto,
                estCarrera = estCarrera
            )

            estudiante.save()

            messages.success(request, f'El estudiante "{estudiante.estNombre} {estudiante.estApellido}" se ha añadido con exito')
            return redirect('mostrarEstudiantes')

        else:
            return render(request, 'estudianteAñadir.html', {'formulario': formulario})

    else:
        formulario = FormEstudiante()
        return render(request, 'estudianteAñadir.html', {'formulario': formulario})
```

```python
def modificarEstudiante(request, id):
    estudiante = Estudiante.objects.get(pk=id)

    if request.method == 'POST':
        formulario = FormActualizarEstudiante(request.POST, request.FILES, instance=estudiante)

        if formulario.is_valid():
            formulario.save()
            messages.success(request, f'Los datos del estudiante "{estudiante.estNombre} {estudiante.estApellido}" se han modificado con éxito')
            return redirect('mostrarEstudiantes')
    else:
        formulario = FormActualizarEstudiante(instance=estudiante)

    return render(request, 'estudianteModificar.html', {'formulario': formulario})
```

```python
def eliminarEstudiante(request, id):
    estudiante = Estudiante.objects.get(pk=id)

    estudiante.delete()
    messages.success(request, f'El estudiante se ha eliminado con éxito')

    return redirect('mostrarEstudiantes')
```

Clase Profesor

```python
def mostrarProfesores(request):
    profesores = Profesor.objects.raw("SELECT id, profNombre, profApellido, profEmail FROM GestionCrud_profesor")

    return render(request, "profesor.html", {
        'profesores' : profesores,
    })

def mostrarProfesor(request, id):
    profesor = Profesor.objects.get(pk = id)

    return render(request, "profesorDetalles.html", {
        'profesor' : profesor,
    })
```

```python
def añadirProfesor(request):
    formulario = FormProfesor()

    if request.method == 'POST':

        formulario = FormProfesor(request.POST, request.FILES)

        if formulario.is_valid():
            data_form = formulario.cleaned_data

            profNombre = data_form.get('nombre')
            profApellido = data_form.get('apellido')
            profEmail = data_form.get('email')
            profTelefono = data_form.get('telefono')
            profFechaNacimiento = data_form.get('fechaNacimiento')
            profFoto = data_form.get('foto')
            profMaterias = data_form.get('materias')

            profesor = Profesor(
                profNombre = profNombre,
                profApellido = profApellido,
                profEmail = profEmail,
                profTelefono = profTelefono,
                profFechaNacimiento = profFechaNacimiento,
                profFoto = profFoto,
            )

            profesor.save()

            if profMaterias:
                profesor.profMaterias.set(profMaterias)

            messages.success(request, f'El profesor "{profesor.profNombre} {profesor.profApellido}" se ha añadido con exito')
            profesor.save()

            return redirect('mostrarProfesores')

        else:
            return render(request, 'profesorAñadir.html', {'formulario': formulario})
```

```python
def modificarProfesor(request, id):
    profesor = Profesor.objects.get(pk=id)

    if request.method == 'POST':
        formulario = FormActualizarProfesor(request.POST, request.FILES, instance=profesor)

        if formulario.is_valid():
            formulario.save()
            messages.success(request, f'Los datos del profesor "{profesor.profNombre} {profesor.profApellido}" se han modificado con éxito')
            return redirect('mostrarProfesores')
    else:
        formulario = FormActualizarProfesor(instance=profesor)

    return render(request, 'profesorModificar.html', {'formulario': formulario})


def eliminarProfesor(request, id):
    profesor = Profesor.objects.get(pk=id)

    profesor.profMaterias.clear()
    profesor.delete()

    messages.success(request, f'El Profesor se ha eliminado con éxito')

    return redirect('mostrarProfesores')
```

Clase Carrera

```python
def mostrarCarreras(request):
    carreras = Carrera.objects.raw("SELECT id, carNombre FROM GestionCrud_carrera")

    return render(request, "carrera.html", {
        'carreras' : carreras,
    })


def mostrarCarrera(request, id):
    carrera = Carrera.objects.get(pk = id)

    return render(request, "carreraDetalles.html", {
        'carrera' : carrera,
    })
```

```python
def añadirCarrera(request):
    formulario = FormCarrera()

    if request.method == 'POST':

        formulario = FormCarrera(request.POST)

        if formulario.is_valid():
            data_form = formulario.cleaned_data

            carNombre = data_form.get('nombre')
            carDuracion = data_form.get('duracion')
            carMaterias = data_form.get('materias')

            carrera = Carrera(
                carNombre = carNombre,
                carDuracion = carDuracion,
            )

            carrera.save()

            if carMaterias:
                carrera.carMaterias.set(carMaterias)

            messages.success(request, f'La carrera "{carrera.carNombre}" se ha añadido con exito')
            carrera.save()

            return redirect('mostrarCarreras')

        else:
            return render(request, 'carreraAñadir.html', {'formulario': formulario})

    else:
        formulario = FormCarrera()
        return render(request, 'carreraAñadir.html', {'formulario': formulario})
```

```python
def modificarCarrera(request, id):
    carrera = Carrera.objects.get(pk=id)

    if request.method == 'POST':
        formulario = FormActualizarCarrera(request.POST, instance=carrera)

        if formulario.is_valid():
            formulario.save()
            messages.success(request, f'Los datos de la carrera "{carrera.carNombre}" se han modificado con éxito')
            return redirect('mostrarCarreras')
    else:
        formulario = FormActualizarCarrera(instance=carrera)

    return render(request, 'carreraModificar.html', {'formulario': formulario})


def eliminarCarrera(request, id):
    carrera = Carrera.objects.get(pk=id)

    carrera.carMaterias.clear()
    carrera.delete()
    messages.success(request, f'La carrera "{carrera.carNombre}" se ha eliminado con éxito')

    return redirect('mostrarCarreras')
```

Clase Materia

```python
def mostrarMaterias(request):
    materias = Materia.objects.raw("SELECT id, matNombre FROM GestionCrud_materia")

    return render(request, "materia.html", {
        'materias' : materias,
    })

def mostrarMateria(request, id):
    materia = Materia.objects.get(pk = id)

    return render(request, "materiaDetalles.html", {
        'materia' : materia,
    })
```

```python
def añadirMateria(request):
    formulario = FormMateria()

    if request.method == 'POST':

        formulario = FormMateria(request.POST)

        if formulario.is_valid():
            data_form = formulario.cleaned_data

            matNombre = data_form.get('nombre')
            matDescripcion = data_form.get('descripcion')
            matCreditos = data_form.get('creditos')
            matCarrera = data_form.get('carrera')
            matProfesores = data_form.get('profesor')

            materia = Materia(
                matNombre=matNombre,
                matDescripcion=matDescripcion,
                matCreditos=matCreditos,
                matProfesores=matProfesores,
            )

            materia.save()

            if matCarrera:
                materia.matCarrera.set(matCarrera)

            messages.success(request, f'La materia "{materia.matNombre}" se ha añadido con exito')
            materia.save()

            return redirect('mostrarMaterias')

        else:
            return render(request, 'materiaAñadir.html', {'formulario': formulario})

    else:
        formulario = FormMateria()
        return render(request, 'materiaAñadir.html', {'formulario': formulario})
```

```python
def modificarMateria(request, id):
    materia = Materia.objects.get(pk=id)

    if request.method == 'POST':
        formulario = FormActualizarMateria(request.POST, instance=materia)

        if formulario.is_valid():
            formulario.save()
            messages.success(request, f'Los datos de la materia "{materia.matNombre}" se han modificado con éxito')
            return redirect('mostrarMaterias')
    else:
        formulario = FormActualizarMateria(instance=materia)

    return render(request, 'materiaModificar.html', {'formulario': formulario})


def eliminarMateria(request, id):
    materia = Materia.objects.get(pk=id)

    materia.matCarrera.clear()
    materia.delete()
    messages.success(request, f'La Materia se ha eliminado con éxito')

    return redirect('mostrarMaterias')
```

Ficheros registrados

```
class EstudianteAdmin(admin.ModelAdmin):
    readonly_fields = ('estFechaCreacion', 'estFechaActualizacion')

admin.site.register(Estudiante, EstudianteAdmin)


class ProfesorAdmin(admin.ModelAdmin):
    readonly_fields = ('profFechaCreacion', 'profFechaActualizacion')

admin.site.register(Profesor, ProfesorAdmin)


class MateriaAdmin(admin.ModelAdmin):
    readonly_fields = ('matFechaCreacion', 'matFechaActualizacion')

admin.site.register(Materia, MateriaAdmin)


admin.site.register(Carrera)


title = "Proyecto UCT"
admin.site.site_header = title
admin.site.site_title= title
admin.site.index_title = "Panel de gestión"
```

Modelos

Clase Estudiante

```python
class Estudiante(models.Model):
    estNombre = models.CharField(max_length=30, verbose_name='Nombre')
    estApellido = models.CharField(max_length=50, verbose_name='Apellido')
    estEmail = models.EmailField(verbose_name='Email')
    estTelefono = models.IntegerField(verbose_name='Telefono')
    estFechaNacimiento = models.DateField(verbose_name='Fecha de nacimiento')
    estFoto = models.ImageField(default='null', verbose_name='Foto', upload_to="estFotos")
    estFechaCreacion = models.DateTimeField(auto_now_add=True, verbose_name='Fecha de creación')
    estFechaActualizacion = models.DateTimeField(auto_now=True, verbose_name='Fecha de actualización')
    estCarrera = models.ForeignKey(Carrera, verbose_name='Carrera', on_delete=models.CASCADE)

    class Meta:
        verbose_name="Estudiante"
        verbose_name_plural="Estudiantes"
        ordering=['id']

    def __str__(self):
        return f"{self.estNombre} {self.estApellido}"
```

Clase Profesor

```python
class Profesor(models.Model):
    profNombre = models.CharField(max_length=30, verbose_name='Nombre')
    profApellido = models.CharField(max_length=50, verbose_name='Apellido')
    profEmail = models.EmailField(verbose_name='Email')
    profTelefono = models.IntegerField(verbose_name='Telefono')
    profFechaNacimiento = models.DateField(verbose_name='Fecha de nacimiento')
    profFoto = models.ImageField(upload_to="profFotos",  verbose_name='Foto')
    profFechaCreacion = models.DateTimeField(auto_now_add=True, verbose_name='Fecha de creación')
    profFechaActualizacion = models.DateTimeField(auto_now=True, verbose_name='Fecha de actualización')
    profMaterias = models.ManyToManyField(Materia, verbose_name='Materias')

    class Meta:
        verbose_name="Profesor"
        verbose_name_plural="Profesores"
        ordering=['id']

    def __str__(self):
        return f"{self.profNombre} {self.profApellido}"
```

Clase Carrera

```python
class Carrera(models.Model):
    carNombre = models.CharField(max_length=50, verbose_name='Nombre')
    carDuracion = models.CharField(max_length=50, verbose_name='Duración')
    carMaterias = models.ManyToManyField('Materia', verbose_name='Materias')

    class Meta:
        verbose_name="Carrera"
        verbose_name_plural="Carreras"
        ordering=['id']

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

Clase Materia

```python
class Materia(models.Model):
    matNombre = models.CharField(max_length=50, verbose_name='Nombre')
    matDescripcion = models.TextField(verbose_name='Descripcion')
    matCreditos = models.IntegerField(verbose_name='Creditos')
    matFechaCreacion = models.DateTimeField(auto_now_add=True, verbose_name='Fecha de creación')
    matFechaActualizacion = models.DateTimeField(auto_now=True, verbose_name='Fecha de actualización')
    matCarrera = models.ManyToManyField(Carrera, verbose_name='Carrera')
    matProfesores = models.ForeignKey('Profesor', verbose_name='Profesor', on_delete=models.CASCADE)

    class Meta:
        verbose_name="Materia"
        verbose_name_plural="Materias"
        ordering=['id']

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

Formularios

Clase Estudiante

```python
class FormEstudiante(forms.Form):
    nombre = forms.CharField(
        label="Nombre",
        max_length=30,
        required=True,
        widget=forms.TextInput(
            attrs={
                'class':'formEstNombre',
                'placeholder':'Ingrese su nombre'
            }
        ),
        validators= [
            validators.RegexValidator('^[a-zA-Z\sáéíóúüñÁÉÍÓÚÜÑ]*$', 'Error(Nombre): solo se permiten letras y espacios.')
        ]
    )

    apellido = forms.CharField(
        label="Apellido",
        max_length=50,
        required=True,
        widget=forms.TextInput(
            attrs={
                'class':'formEstApe',
                'placeholder':'Ingrese su apellido'
            }
        ),
        validators= [
            validators.RegexValidator('^[a-zA-Z\sáéíóúüñÁÉÍÓÚÜÑ]*$', 'Error(Apellido): solo se permiten letras y espacios.')
        ]
```

```python
    email = forms.EmailField(
        label="E-mail",
        required=True,
        widget=forms.EmailInput(
            attrs={
                'class':'formEstEmail',
                'placeholder':'Ingrese su E-mail'
            }
        ),
        validators= [
            validators.EmailValidator('Error(Email): El email está mal digitado.')
        ]
    )

    telefono = forms.IntegerField(
        label="Telefono",
        required=True,
        widget=forms.NumberInput(
            attrs={
                'class':'formEstTel',
                'placeholder':'Ingrese su numero de telefono'
            }
        ),
        validators= [
            validators.RegexValidator('^[0-9]+$', 'Error(Telefono): solo se permiten numeros.')
        ]
```

```
fechaNacimiento = forms.DateField(
    label="Fecha De Nacimiento",
    required=True,
    widget=forms.DateTimeInput(
        attrs={
            'class':'formEstFeNa',
            'type': 'date'
        }
    )
)

foto = forms.ImageField(
    label="Foto",
    required=False,
)

carrera = forms.CharField(
    label="Carrera",
    max_length=50,
    required=True,
    widget=forms.TextInput(
        attrs={
            'class':'formEstCar',
            'placeholder':'Ingrese el nombre de su carrera'
        }
    ),
    validators= [
        validators.RegexValidator('^[a-zA-Z\sáéíóúüñÁÉÍÓÚÜÑ]*$', 'Error(Carrera): solo se permiten letras y espacios.')
    ]
```

```
class FormActualizarEstudiante(forms.ModelForm):
    class Meta:
        model = Estudiante
        fields = ['estNombre', 'estApellido', 'estEmail', 'estTelefono', 'estFechaNacimiento', 'estFoto', 'estCarrera']
```

Clase Profesor

```
class FormProfesor(forms.Form):
    nombre = forms.CharField(
        label="Nombre",
        max_length=30,
        required=True,
        widget=forms.TextInput(
            attrs={
                'class':'formProfNombre',
                'placeholder':'Ingrese su nombre'
            }
        ),validators= [
            validators.RegexValidator('^[a-zA-Z\sáéíóúüñÁÉÍÓÚÜÑ]*$', 'Error(Nombre): solo se permiten letras y espacios.')
        ]
    )
```

```
apellido = forms.CharField(
    label="Apellido",
    max_length=50,
    required=True,
    widget=forms.TextInput(
        attrs={
            'class':'formProfApe',
            'placeholder':'Ingrese su apellido'
        }
    ),validators= [
        validators.RegexValidator('^[a-zA-Z\sáéíóúüñÁÉÍÓÚÜÑ]*$', 'Error(Apellido): solo se permiten letras y espacios.')
    ]
)

email = forms.EmailField(
    label="E-mail",
    required=True,
    widget=forms.EmailInput(
        attrs={
            'class':'formProfEmail',
            'placeholder':'Ingrese su E-mail'
        }
    ),
    validators= [
        validators.EmailValidator('Error(Email): El email está mal digitado.')
    ]
)
```

```
telefono = forms.IntegerField(
    label="Telefono",
    required=True,
    widget=forms.NumberInput(
        attrs={
            'class':'formProfTel',
            'placeholder':'Ingrese su numero de telefono'
        }
    ),
    validators= [
        validators.RegexValidator('^[0-9]+$', 'Error(Telefono): solo se permiten numeros.')
    ]
)

fechaNacimiento = forms.DateField(
    label="Fecha De Nacimiento",
    required=True,
    widget=forms.DateInput(
        attrs={
            'class':'formProfFeNa',
            'type': 'date'
        }
    )
)
```

```
    foto = forms.ImageField(
        label="Foto",
        required=False,
    )

    materias = forms.CharField(
        label="Materias",
        max_length=50,
        required=True,
        widget=forms.TextInput(
            attrs={
                'class':'formProfMat',
                'placeholder':'Ingrese el nombre de su materia'
            }
        ),
        validators= [
            validators.RegexValidator('^[a-zA-Z\sáéíóúüñÁÉÍÓÚÜÑ]*$', 'Error(Materia): solo se permiten letras y espacios.')
        ]
    )
```

```
class FormActualizarProfesor(forms.ModelForm):
    class Meta:
        model = Profesor
        fields = ['profNombre', 'profApellido', 'profEmail', 'profTelefono', 'profFechaNacimiento', 'profFoto', 'profCMaterias']
```

Clase Carrera

```
class FormCarrera(forms.Form):
    nombre = forms.CharField(
        label="Nombre",
        max_length=50,
        required=True,
        widget=forms.TextInput(
            attrs={
                'class':'formCarNombre',
                'placeholder':'Ingrese el nombre de la carrera'
            }
        ),
        validators= [
            validators.RegexValidator('^[a-zA-Z\sáéíóúüñÁÉÍÓÚÜÑ]*$', 'Error(Nombre): solo se permiten letras y espacios.')
        ]
    )

    duracion = forms.CharField(
        label="Duración",
        max_length=50,
        required=True,
        widget=forms.TextInput(
            attrs={
                'class':'formCarDur',
                'placeholder':'Ingrese la duración de la carrera (ej: 9 semestres)'
            }
        ),
        validators= [
            validators.RegexValidator('^[a-zA-Z0-9 ]*$', 'Error(Duracion): solo se permiten letras, numeros y espacios.')
        ]
    )
```

```
    materias = forms.CharField(
        label="Materias",
        max_length=50,
        required=True,
        widget=forms.TextInput(
            attrs={
                'class':'formCarMat',
                'placeholder':'Ingrese las materias que se darán en la carrera'
            }
        ),
        validators= [
            validators.RegexValidator('^[a-zA-Z\sáéíóúüñÁÉÍÓÚÜÑ]*$', 'Error(Materia): solo se permiten letras y espacios.')
        ]
    )


class FormActualizarCarrera(forms.ModelForm):
    class Meta:
        model = Carrera
        fields = ['carNombre', 'carDuracion', 'carMaterias']
```

Clase Materia

```
class FormMateria(forms.Form):
    nombre = forms.CharField(
        label="Nombre",
        max_length=50,
        required=True,
        widget=forms.TextInput(
            attrs={
                'class':'formMatNombre',
                'placeholder':'Ingrese el nombre de la materia'
            }
        ),validators= [
            validators.RegexValidator('^[a-zA-Z0-9 ]*$', 'Error(Nombre): solo se permiten letras, numeros y espacios.')
        ]
    )

    descripcion = forms.CharField(
        label="Descripcion",
        required=False,
        widget=forms.Textarea(
            attrs={
                'class':'formMatDesc',
                'placeholder':'Ingrese la descripción de la materia'
            }
        ),
        validators= [
            validators.RegexValidator('^[\w\sáéíóúÁÉÍÓÚüÜ,.-]*$', 'Error(Descripcion): solo se permiten letras y espacios.')
        ]
    )
```

```
    creditos = forms.IntegerField(
        label="Creditos",
        required=True,
        widget=forms.NumberInput(
            attrs={
                'class':'formMatCred',
                'placeholder':'Ingrese la cantidad de creditos (Ej: 2)'
            }
        ),
        validators= [
            validators.RegexValidator('^[0-9]+$', 'Error(Creditos): solo se permiten numeros.')
        ]
    )

    carrera = forms.CharField(
        label="Carrera",
        max_length=50,
        required=True,
        widget=forms.TextInput(
            attrs={
                'class':'formMatCar',
                'placeholder':'Ingrese el nombre de la carrera'
            }
        ),
        validators= [
            validators.RegexValidator('^[a-zA-Z\sáéíóúüñÁÉÍÓÚÜÑ]*$', 'Error(Carrera): solo se permiten letras y espacios.')
        ]
    )
```

```
    profesor = forms.CharField(
        label="Profesor",
        max_length=30,
        required=True,
        widget=forms.TextInput(
            attrs={
                'class':'formMatProf',
                'placeholder':'Ingrese el nombre del profesor que dará la materia'
            }
        ),
        validators= [
            validators.RegexValidator('^[a-zA-Z\sáéíóúüñÁÉÍÓÚÜÑ]*$', 'Error(Profesor): solo se permiten letras y espacios.')
        ]
    )


class FormActualizarMateria(forms.ModelForm):
    class Meta:
        model = Materia
        fields = ['matNombre', 'matDescripcion', 'matCreditos', 'matProfesores', 'matCarrera']
```

Conclusiones

Tras un análisis en el proceso de planeación y construcción del sistema de gestión académica de la Universidad Ceitocol, se han identificado varias conclusiones relevantes de este proceso:

- Tras completar este proyecto con sus fases posteriores, la Universidad contará con un sistema de gestión mucho mejor que el que tenía anteriormente, en cuanto a su usabilidad, fluidez, experiencia de usuario y escalabilidad, ya que estos puntos que fueron expuestos antes de comenzar con el proyecto con respecto a su anterior sistema de gestión, fueron las que más se tomaron en cuenta para realizar este proyecto.

- En cuanto a el apartado visual, el sistema de gestión cuenta con una interfaz mucho más amigable con los usuarios de este, a comparación con la que se tenía anteriormente, ya que, al estar el sistema anterior obsoleto, la interfaz de este también necesitaba unas mejoras las cuales se implementaron y se seguirán implementando según se requiera en el proyecto actual.

- En general, el proyecto hasta el momento ha logrado satisfacer las necesidades y carencias del sistema anterior, se prevé que con las fases posteriores, se puedan mejorar muchos más aspectos y que quede un sistema completamente mejorado y eficiente.

Bibliografia

- *Django documentation | Django documentation*. (s. f.). Django Project. https://docs.djangoproject.com/en/5.0/

- Zuluaga, J. G. (2024). *Django_Zuluaga_Manual*.

- [Fotos de stock gratis, imágenes libres de regalías y sin derechos de autor]. (s. f.). Recuperado 10 de abril de 2024, de https://www.pexels.com/es-es/