Informe de Laboratorio Proyecto Web

Tema: Proyecto Web - Final

Nota

Estudiantes	Escuela	Asignatura
Carrasco Choque Arles, Chara Condori Jean Carlo, Choquecondo Aspilcueta Daniela, Zapata Butron Reyser	Escuela Profesional de Ingeniería de Sistemas	Programación Web 2 Semestre: III

Laboratorio	Tema	Duración
Proyecto Web	Proyecto Web - Final	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2023 - A	Del 1 Agosto 2023	Al 5 Agosto 2023

1. REPOSITORIO DE GITHUB:

- Link del Repositorio en github: `https://github.com/carrascoArles/ProyectoFinalFinal.git`

2. REQUISITOS DEL SISTEMA:

- RQ01 : El sistema debe estar disponible en Internet a traves de una URL.
- RQ02 : El sistema debe permitir el inicio/cierre de sesión.
- RQ03 : El sistema debe permitir gestionar el año académico, cursos, profesores y las asignaciones de carga académica.

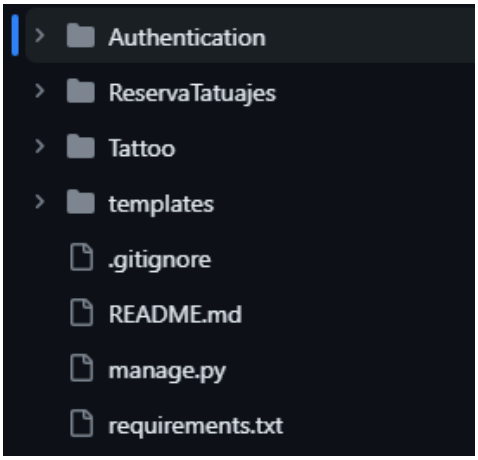
3. ACTIVIDADES:

- Modelo de datos
- Diccionario de datos
- Diagrama Entidad-Relación
- Administración con Django

- Plantillas Bootstrap
- CRUD - Core Business - Clientes finales
- Servicios mediante una API RESTful
- Operaciones asíncronas AJAX
- Investigación: Email, Upload.

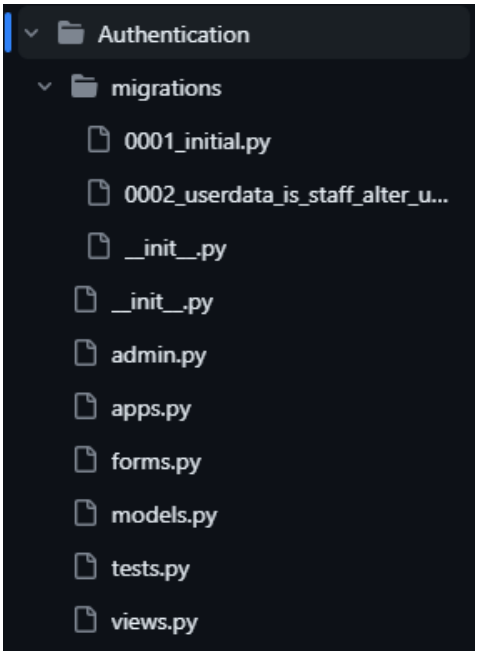
4. DESARROLLO:

1. Estructura del Proyecto:



Contenido de cada carpeta y archivos dentro:

- Authentication:



- Archivos correspondientes a la carpeta 'Authentication':
  - admin.py: Habilita la interfaz de administración web preconstruida, facilitando la gestión de datos de la aplicación mediante una interfaz gráfica.

```
        from django.contrib import admin
```

  - apps.py: Usa BigAutoField como campo automático de modelo para las migraciones de base de datos.

```
        from django.apps import AppConfig
class AuthenticationConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'Authentication'
```

  - forms.py:

```
        from django import forms
from django.core.validators import MaxLengthValidator
...
```

  - models.py:

```
        from django.db import models
from django.contrib.auth.models import AbstractBaseUser, BaseUserManager,
    PermissionsMixin
...
```

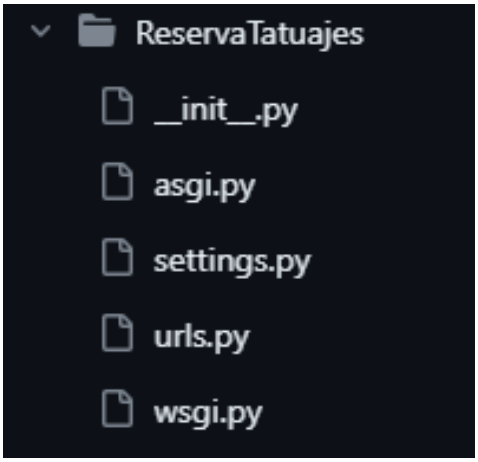
  - test.py: Importa la herramienta TestCase de Django para crear pruebas que verifiquen el funcionamiento de la aplicación.

```
        from django.test import TestCase
```

  - views.py:

```
        import requests
from django.shortcuts import render, redirect
from django.contrib.auth import login
from .forms import LoginForm, RegisterForm
from .models import UserData
from Tattoo import views

def homeView(request):
    return render(request, 'home.html', {'user': request.user})

...
```

- ReservasTatuajes:



- asgi.py: Establece la variable de entorno DJANGO-SETTINGS-MODULE, se define una instancia de la aplicación ASGI y se utilizará para manejar solicitudes asincrónicas.

```
        """
ASGI config for ReservaTatuajes project.

It exposes the ASGI callable as a module-level variable named ‘‘application‘‘.

For more information on this file, see
https://docs.djangoproject.com/en/4.2/howto/deployment/asgi/
"""

import os

from django.core.asgi import get_asgi_application

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'ReservaTatuajes.settings')

application = get_asgi_application()
```

- settings.py: Establece la clave secreta, las aplicaciones instaladas y la base de datos, define la configuración para plantillas y correo electrónico. También, configura la validación de contraseñas y el modelo de usuario personalizado.

```
        """
Django settings for ReservaTatuajes project.

Generated by 'django-admin startproject' using Django 4.2.4.

For more information on this file, see
https://docs.djangoproject.com/en/4.2/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/4.2/ref/settings/
...
```

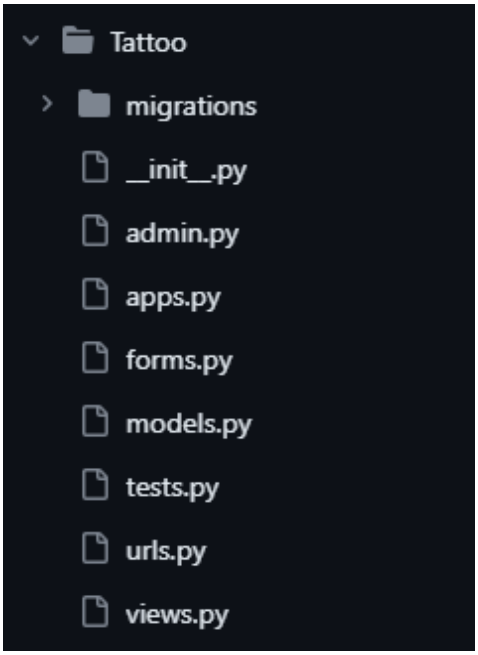
- urls.py: configura las rutas del proyecto, incluyendo administración, autenticación, vistas personalizadas y manejo de archivos estáticos y multimedia. También incorpora URLs de la app "Tattoo".

```
        from django.contrib import admin
from django.urls import path,include
from django.conf import settings
from django.conf.urls.static import static
from Tattoo import views
from Authentication.views import loginView, registerView
from django.contrib.auth import views as auth_views

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

- wsgi.py: Configura el servidor WSGI para ReservaTatuajes", exponiendo la aplicación WSGI como .ªpplication". Establece la configuración del proyecto y utiliza la función get-wsgi-application() para obtener la aplicación WSGI configurada.

```
        """
WSGI config for ReservaTatuajes project.

It exposes the WSGI callable as a module-level variable named ``application``.

...
```

- Tattoo:



- admin.py: Se registra los modelos 'Tatuador', 'ImagenTattoos' y 'Usuario' en la interfaz de administración de Django, permitiendo su gestión y visualización en la sección de administración del sitio web.

```
        from django.contrib import admin
from Tattoo.models import *
# Register your models here.
```

```
admin.site.register(Tatuador)
admin.site.register(ImagenTattoos)
admin.site.register(Usuario)
```

- apps.py: Se configura la aplicación 'Tattoo' en Django al establecer el tipo de campo automático predeterminado para los modelos como 'BigAutoField' y asignar el nombre de la aplicación como 'Tattoo'.

```
        from django.apps import AppConfig
class TattooConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'Tattoo'
```

- forms.py: Se define formularios en Django para los modelos 'Tatuador', 'ImagenTattoos' y 'Usuario'. Estos se utilizan para capturar y validar datos al crear o editar instancias de esos modelos en la aplicación.

```
        from django import forms
from .models import *

class TattooForm(forms.ModelForm):
    class Meta:
        model = Tatuador
        fields = '__all__'

class ImagenForm(forms.ModelForm):
    class Meta:
        model = ImagenTattoos
        fields = '__all__'
        ...
```

- models.py: Captura información de tatuadores, imágenes de tatuajes y usuarios en la aplicación.

```
        from django.db import models

# Create your models here.

class Tatuador(models.Model):
    nombre =models.CharField(max_length=50)
    apellido = models.CharField(max_length=50)
    correo = correo = models.EmailField(unique=True)
    telefono = models.CharField(max_length=9,)
    ...
```

- test.py:

```
        from django.test import TestCase
```

- urls.py: establece la navegación para la aplicación 'ReservaTatuajes', incluyendo vistas para ver tatuajes, pedir citas con tatuadores, mostrar formularios, eliminar tatuadores y acceder a una vista de administración.

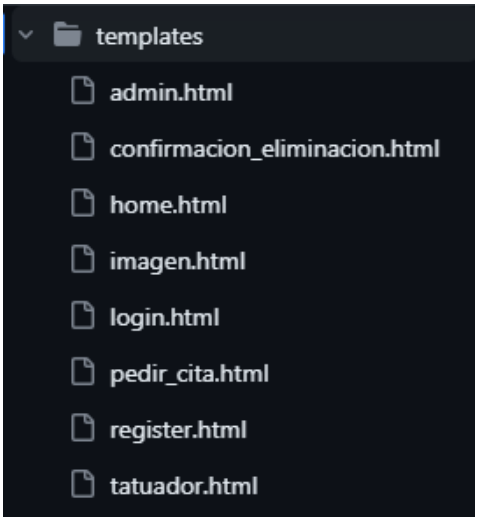
```
        from django.urls import path
```

```
from . import views


urlpatterns = [
    path ('', views.vista_tatuajes, name = 'vista_tatuajes'),
    path ('pedir_cita/<int:tatuador_dni>',views.pedir_cita,name='pedir_cita'),
    path('formulario/', views.Tatuador, name='mi_f'),
    path('eliminar_tatuador/<str:tatuador_dni>/', views.eliminar_tatuador,
        name='eliminar_tatuador'),
    path ('admin/', views.vista_tatuadores, name = 'admin'),

]
```

- views.py:

```
        from .models import ImagenTattoos,Tatuador
from django.shortcuts import render, redirect,get_object_or_404
from django.core.mail import send_mail
from .forms import *

def vista_tatuajes(request):
    tatuadores = Tatuador.objects.all()
    tatuajes = ImagenTattoos.objects.all()
    ...
```

- Templates: En base a los archivos dentro de la carpeta se mostrará cada página (como se ve / se tomará fragmentos del código completo para la explicación).



- admin.html: muestra una lista de tatuadores con la opción de eliminar cada uno.

```
        <body>
    <header>
        <nav>
            <ul>
                <li><a href="/">Inicio</a></li>
                    <h1>Vista de Administrador</h1>
                    <a href="{% url 'creartatuador' %}">Agregar Tatuador</a>
```

```
                <a href="{% url 'imagentatu' %}">Agregar Trabajo</a>
            </ul>
        </nav>
    </header>
    \item ...
```

- confirmacion-eliminacion.html: permite al administrador confirmar o cancelar la eliminación de un tatuador específico.

```
<body>
    <h1>Confirmar eliminacin del tatuador: {{ tatuadores.nombre }} {{
        tatuadores.apellido }}</h1>
    <p> Ests  seguro de que deseas eliminar a {{ tatuadores.nombre }} {{
        tatuadores.apellido }}?</p>

    <form method="post">
        {% csrf_token %}
        <input type="submit" value="Eliminar">
    </form>

    <p><a href="{% url 'admin' %}">Cancelar y volver</a></p>
</body>
```

- home.html: Proporciona una barra de navegación que cambia según el estado de inicio de sesión del usuario. Muestra información sobre tatuadores, sus trabajos y ofrece la posibilidad de pedir citas. Si el usuario es administrador, muestra un enlace a una vista de administrador adicional.

```
<body>
    <header>
        <nav>
            <ul>
                <li><a href="/">Inicio</a></li>

                {% if user.is_authenticated %}
                    <h1>Bienvenido, {{ user.nombres }}</h1>
                    <form action="{% url 'logout' %}" method="post">
                        {% csrf_token %}
                        <button type="submit">Cerrar sesin</button>
                    </form>
                    \item ...
```

- imagen.html: Los usuarios pueden completar el formulario, guardarlo y regresar a la vista de administrador. Se utiliza el método POST y se incluye un token CSRF.

```
<body>
    <h1>Rellenar datos</h1>
    <form method="post" enctype="multipart/form-data">
        {% csrf_token %}
        {{ form1.as_p }}
        <button type="submit">Guardar</button>
        <a href="{% url 'vistaadmin' %}">
            <button type="button">Volver</button>
        </a>
    </form>
```

```
</body>
```

- login.html: Los usuarios pueden enviar sus datos con un botón "Login". Se utiliza el método POST y se incluye un token CSRF para la seguridad.

```
<body>
    <h1>Login</h1>
    <form method="post">
        {% csrf_token %}
        {{ form.as_p }}
        <button type="submit">Login</button>
    </form>
</body>
```

- pedir-cita.html: Ofrece opciones de contacto como Instagram, Whatsapp y correo electrónico. Incluye un formulario para enviar mensajes, muestra el nombre del usuario y proporciona un mensaje de éxito si se envía correctamente. Utiliza un token CSRF y campos generados de 'user' y 'tatuador'.

```
<body>
    <a href="/" class="sub">Inicio</a>

    <h1>Contactarse con {{ tatuador.nombre }}</h1>

    <a href="{{tatuador.ig}}">Comunicarse por Instagram</a>
    <a href="https://wa.me/{{tatuador.telefono}}">Comunicarse por Whatsapp</a>
    <p>Correo: {{tatuador.correo}}</p>
...
```

- register.html: enerando campos automáticamente desde un formulario llamado 'form'. Los usuarios pueden registrarse y, si hay un mensaje de alerta, se muestra en caso de error. Utiliza un token CSRF para seguridad.

```
<body>
    <h1>Register</h1>
    <form method="post">
        {% csrf_token %}
        <div>
            {{ form.as_p }}
        </div>
        <button type="submit">Register</button>
    </form>
```

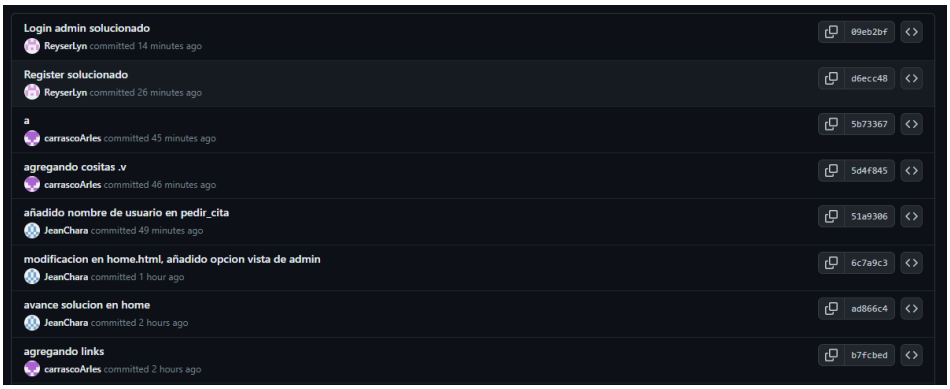
- tatuador.htlm: se guarda la información y regresa a la vista de administrador., admite el envío de archivos con un token CSRF para seguridad.

```
<body>
    <h1>Rellenar datos</h1>
    <form method="post" enctype="multipart/form-data">
        {% csrf_token %}
        {{ form.as_p }}
        <button type="submit">Guardar</button>
        <a href="{% url 'vistaadmin' %}">
            <button type="button">Volver</button>
        </a>
```

```
    </form>
</body>
```

5. COMMITS IMPORTANTES EN GITHUB:

Para este Proyecto, se necesitó de varios commits, en los cuales se puede apreciar el progreso que hubo para poder completar con éxito el trabajo..



6. Referencias

- https://desarrolloweb.com/home/angular
- https://www.djangoproject.com/
- https://www.redhat.com/es/topics/api/what-is-a-rest-api
- https://www.redhat.com/es/topics/api/what-is-a-rest-api