

ServiceMatch Documentation

version 1.0

2024, Tiffani Bastidas, Francisco Pinol, Raquel Guarda

diciembre 08, 2024

Contenido

Documentación de ServiceMatch	1
Vistas del Proyecto ServiceMatch	1
1. Vistas del admin	1
2. Vistas de la autenticación (login y logout)	1
3. Vistas del cliente	2
4. Vistas para las Estadísticas y Filtros	3
5. Vistas para Gestionar y Validar Servicios	3
6. Vistas para Gestionar Profesionales	5
7. Vistas para Profesionales	6
8. Vistas para las Reservas	7
Models en el Proyecto ServiceMatch	8
1. Modelo Usuario	8
2. Modelo Servicio	8
3. Modelo Subcategoría de Servicio	8
4. Modelo Reserva	8
5. Modelo Reseña	8
Middleware	9
Bloquear Navegación Middleware	9
Settings del Proyecto	10
Configuraciones clave	10
URLs del Proyecto	10
Rutas principales	11
Índice	13

Documentación de ServiceMatch

Vistas del Proyecto ServiceMatch

Este archivo contiene la documentación de las vistas del proyecto.

1. Vistas del admin

`app.views.admin_views.admin_home` (request)

Renderiza la página principal del administrador con estadísticas clave.

Parámetros: `request` (*HttpRequest*) – Objeto de solicitud HTTP.

Devuelve: Renderiza la plantilla con datos de resumen administrativo.

Tipo del valor devuelto: `HttpResponse`

Estadísticas proporcionadas:

- Total de reservas realizadas.
- Número de reservas completadas.
- Total de usuarios registrados.
- Ganancias totales basadas en las reservas completadas.
- Promedio mensual de ganancias (calculado si hay datos disponibles).

Context:

`total_reservas` (int): Total de reservas realizadas en el sistema. `reservas_completadas` (int): Total de reservas completadas. `total_usuarios` (int): Número de usuarios registrados. `promedio_mensual` (float): Promedio mensual de ganancias. `ganancias` (float): Total de ganancias generadas por las reservas completadas.

`app.views.admin_views.es_admin` (user)

Verifica si el usuario es un administrador.

Parámetros: `user` (*User*) – Objeto usuario autenticado.

Devuelve: True si el usuario es administrador.

Tipo del valor devuelto: `bool`

Muestra: `PermissionDenied` – Si el usuario no está autenticado o no es administrador.

2. Vistas de la autenticación (login y logout)

`app.views.auth_views.user_register` (request)

Maneja el registro de nuevos usuarios.

Parámetros: `request` (*HttpRequest*) – Objeto de solicitud HTTP.

Devuelve: Renderiza la página de registro o redirige tras el registro exitoso.

Tipo del valor devuelto: `HttpResponse`

`app.views.auth_views.user_login` (request)

Maneja el inicio de sesión de los usuarios.

Parámetros: `request` (*HttpRequest*) – Objeto de solicitud HTTP.

Devuelve: Redirige según el rol del usuario autenticado o muestra mensajes de error.

Tipo del valor devuelto: HttpResponse

`app.views.authProfesional_views.login_profesional(request)`

Maneja el inicio de sesión para usuarios con rol de "profesional".

Este endpoint valida las credenciales ingresadas por el usuario y verifica si el usuario tiene el rol de profesional antes de permitir el acceso.

Parámetros: `request (HttpRequest)` – Solicitud HTTP que incluye los datos del formulario.

Devuelve: Redirección a la página del profesional si las credenciales son correctas; de lo contrario, redirige de nuevo a la página de inicio de sesión con mensajes de error.

Tipo del valor devuelto: HttpResponse

`app.views.auth_views.user_logout(request)`

Maneja el cierre de sesión de los usuarios.

Parámetros: `request (HttpRequest)` – Objeto de solicitud HTTP.

Devuelve: Redirige a la página de inicio de sesión después de cerrar sesión.

Tipo del valor devuelto: HttpResponse

`app.views.auth_views.terminos(request)`

Renderiza la página de términos y condiciones.

Parámetros: `request (HttpRequest)` – Objeto de solicitud HTTP.

Devuelve: Renderiza la plantilla de términos y condiciones.

Tipo del valor devuelto: HttpResponse

3. Vistas del cliente

`app.views.cliente_views.cliente_home(request)`

Renderiza la página de inicio del cliente.

Proporciona estadísticas sobre las reservas del cliente y muestra una lista de los profesionales asociados con él.

Parámetros: `request (HttpRequest)` – Solicitud HTTP.

Devuelve: Página HTML con la información del cliente y sus estadísticas.

Tipo del valor devuelto: HttpResponse

`app.views.cliente_views.actualizar_cliente(request)`

Actualiza los datos personales del cliente autenticado.

Permite modificar la información personal y la contraseña del cliente.

Parámetros: `request (HttpRequest)` – Solicitud HTTP con datos del formulario.

Devuelve: Página HTML actualizada o redirección con mensajes de error o éxito.

Tipo del valor devuelto: HttpResponse

`app.views.cliente_views.calificar_profesional(request, profesional_id)`

Permite a un cliente calificar a un profesional.

Registra una reseña con una calificación y un comentario opcional.

Parámetros:

- `request (HttpRequest)` – Solicitud HTTP con datos del formulario.
- `profesional_id (int)` – ID del profesional a calificar.

Devuelve: Página HTML con el resultado de la operación.

Tipo del valor devuelto: `HttpResponse`

`app.views.cliente_views.reservas_totales_cliente(request)`

Lista todas las reservas realizadas por el cliente autenticado.

Muestra información sobre cada reserva, incluyendo el profesional asociado y la subcategoría.

Parámetros: `request (HttpRequest)` – Solicitud HTTP.

Devuelve: Página HTML con la lista de reservas.

Tipo del valor devuelto: `HttpResponse`

4. Vistas para las Estadísticas y Filtros

`app.views.estadisticas_views.obtener_estadisticas_reservas(request)`

Genera estadísticas de reservas agrupadas por mes y estado.

Parámetros: `request (HttpRequest)` – Objeto de solicitud HTTP.

Devuelve: Datos de reservas organizados por estado y mes: completadas: Lista con el número de reservas completadas por mes. pendientes: Lista con el número de reservas pendientes por mes. canceladas: Lista con el número de reservas canceladas por mes. meses: Lista de los nombres de los meses en orden cronológico.

Tipo del valor devuelto: `JsonResponse`

`app.views.estadisticas_views.obtener_estadisticas_tarjetas(request)`

Calcula y retorna estadísticas generales para la vista administrativa.

Parámetros: `request (HttpRequest)` – Objeto de solicitud HTTP.

Devuelve: Datos de estadísticas generales: total_reservas: Total de reservas realizadas. reservas_completadas: Número de reservas completadas. total_usuarios: Número de usuarios registrados. promedio_mensual: Promedio mensual de ganancias. ganancias: Total de ganancias generadas por reservas completadas.

Tipo del valor devuelto: `JsonResponse`

`app.views.filtrarReservas_views.filtrar_reservas(request)`

Filtra las reservas entre dos fechas proporcionadas por el usuario.

Este endpoint permite filtrar las reservas entre una fecha de inicio y una fecha de fin proporcionadas en los parámetros de la solicitud GET. Si las fechas son válidas, se devuelve una lista de las reservas filtradas.

Parámetros: `request (HttpRequest)` – Objeto de solicitud HTTP que contiene los parámetros de fechaInicio y fechaFin.

Devuelve: Un objeto JSON que contiene las reservas filtradas entre las fechas especificadas. En caso de error, devuelve un mensaje de error en formato JSON con un código de estado.

Tipo del valor devuelto: `JsonResponse`

Errores posibles:

- 400 Bad Request: Si las fechas no son proporcionadas o son inválidas.
- 400 Bad Request: Si la fecha de inicio es posterior a la fecha de fin.
- 400 Bad Request: Si el formato de las fechas es incorrecto.

5. Vistas para Gestionar y Validar Servicios

`app.views.gestionProfesion_views.gestionar_profesion(request)`

Muestra una lista de servicios disponibles para la administración.

Solo los usuarios con permisos de administrador pueden acceder a esta vista. Se muestra una lista con todos los servicios registrados en el sistema.

Parámetros: **request** (*HttpRequest*) – Objeto de solicitud HTTP que contiene los datos de la solicitud.

Devuelve: Renderiza la plantilla “gestionar_profesion.html” con la lista de servicios.

Tipo del valor HttpResponse

devuelto:

`app.views.gestionProfesion_views.agregar_profesion(request)`

Permite agregar un nuevo servicio y sus subcategorías asociadas.

Solo los usuarios con permisos de administrador pueden acceder a esta vista. Se debe ingresar un nombre para el servicio y luego las subcategorías correspondientes. Si algún dato de las subcategorías es incompleto, se muestra un mensaje de error.

Parámetros: **request** (*HttpRequest*) – Objeto de solicitud HTTP que contiene los datos del formulario.

Devuelve: Redirige a la vista de gestión de profesiones después de agregar el servicio.

Tipo del valor HttpResponse

devuelto:

`app.views.gestionProfesion_views.validar_subcategoria(nombre, precio, duracion)`

Valida los datos de una subcategoría.

Verifica que todos los campos de la subcategoría sean válidos. El precio debe ser un número decimal y la duración debe ser un número entero. También verifica que todos los campos estén presentes.

Parámetros:

- **nombre** (*str*) – Nombre de la subcategoría.
- **precio** (*str*) – Precio base de la subcategoría.
- **duracion** (*str*) – Duración estimada de la subcategoría.

Devuelve: Una tupla con un valor booleano que indica si la validación fue exitosa y un mensaje que describe el resultado de la validación.

Tipo del valor tuple

devuelto:

`app.views.gestionProfesion_views.actualizar_profesion(request, servicio_id)`

Permite actualizar un servicio y sus subcategorías asociadas.

Solo los usuarios con permisos de administrador pueden acceder a esta vista. Se pueden actualizar el nombre del servicio, las subcategorías existentes, eliminar subcategorías y agregar nuevas subcategorías.

Parámetros:

- **request** (*HttpRequest*) – Objeto de solicitud HTTP que contiene los datos del formulario.
- **servicio_id** (*int*) – ID del servicio que se desea actualizar.

Devuelve: Renderiza la plantilla “actualizar_profesion.html” con el servicio y sus subcategorías.

Tipo del valor HttpResponse

devuelto:

`app.views.gestionProfesion_views.eliminar_profesion(request, servicio_id)`

Permite eliminar un servicio del sistema.

Solo los usuarios con permisos de administrador pueden acceder a esta vista. Esta vista muestra un mensaje de confirmación antes de eliminar el servicio.

Parámetros:

- **request** (*HttpRequest*) – Objeto de solicitud HTTP que contiene la confirmación de eliminación.
- **servicio_id** (*int*) – ID del servicio que se desea eliminar.

Devuelve: Redirige a la vista de gestión de profesiones después de eliminar el servicio.

Tipo del valor HttpResponse

devuelto:

`app.views.gestionProfesion_views.validar_disponibilidad` (profesional, fecha, hora)
 Verifica la disponibilidad de un profesional para una reserva en una fecha y hora específicas.

Parámetros:

- **profesional** (*Profesional*) – El profesional que se está verificando.
- **fecha** (*date*) – La fecha en la que se quiere hacer la reserva.
- **hora** (*time*) – La hora en la que se quiere hacer la reserva.

Muestra: **ValidationError** – Si el horario de la reserva no está disponible o si el profesional ya tiene una reserva en ese horario.

`app.views.gestionProfesion_views.validar_dia_habil` (fecha)

Valida si la fecha de la reserva corresponde a un día hábil (lunes a viernes).

Si la fecha corresponde a un sábado o domingo, se genera un error indicando que solo se puede realizar reservas de lunes a viernes.

Parámetros: **fecha** (*date*) – La fecha de la reserva a validar.

Muestra: **ValidationError** – Si la fecha corresponde a un sábado o domingo.

`app.views.gestionProfesion_views.validar_fecha_futura` (fecha)

Valida si la fecha de la reserva es futura y tiene al menos 24 horas de anticipación.

Compara la fecha de la reserva con la fecha actual. Si la fecha es pasada o no cumple con el requisito de al menos 24 horas de anticipación, se genera un error.

Parámetros: **fecha** (*date*) – La fecha de la reserva a validar.

Muestra: **ValidationError** – Si la fecha es pasada o no cumple con el requisito de 24 horas de anticipación.

6. Vistas para Gestionar Profesionales

`app.views.gestionProfesionales_views.gestionar_profesionales` (request)

Gestiona la visualización de los profesionales registrados en el sistema.

Solo los administradores tienen acceso a esta vista. Obtiene todos los profesionales de la base de datos y los pasa al template para su visualización.

Parámetros: **request** (*HttpRequest*) – Objeto de solicitud HTTP.

Devuelve: Renderiza el template “gestionar_profesionales.html” con la lista de profesionales.

Tipo del valor devuelto: `HttpResponse`

`app.views.gestionProfesionales_views.agregar_profesional` (request)

Permite agregar un nuevo profesional al sistema.

Solo los administradores pueden agregar profesionales. Los datos del nuevo profesional se reciben por POST. Valida que los campos sean correctos y que no haya duplicados en los campos de email y RUT. Si todo es válido, se guarda el nuevo profesional en la base de datos.

Parámetros: **request** (*HttpRequest*) – Objeto de solicitud HTTP.

Devuelve: Renderiza el template “agregar_profesionales.html” si es GET o redirigea “gestionar_profesionales” si la operación de POST es exitosa.

Tipo del valor devuelto: `HttpResponse`

`app.views.gestionProfesionales_views.actualizar_profesional` (request, profesional_id)

Permite actualizar los datos de un profesional en el sistema.

Solo los administradores pueden actualizar los datos de los profesionales. Los datos actualizados se reciben por POST y se validan para evitar duplicados de email y RUT. Si la operación es exitosa, el profesional se actualiza en la base de datos.

Parámetros:

- **request** (*HttpRequest*) – Objeto de solicitud HTTP.
- **profesional_id** (*int*) – ID del profesional a actualizar.

Devuelve: Renderiza el template “actualizar_profesional.html” si es GET o redirigea “gestionar_profesionales” si la operación de POST es exitosa.

Tipo del valor devuelto: HttpResponse

`app.views.gestionProfesionales_views.eliminar_profesional(request, profesional_id)`

Permite eliminar un profesional del sistema.

Solo los administradores pueden eliminar profesionales. La operación de eliminación se realiza solo si el método de solicitud es POST.

Parámetros:

- **request** (*HttpRequest*) – Objeto de solicitud HTTP.
- **profesional_id** (*int*) – ID del profesional a eliminar.

Devuelve: Renderiza el template “eliminar_profesional.html” si es GET o redirigea “gestionar_profesionales” si la operación de eliminación es exitosa.

Tipo del valor devuelto: HttpResponse

7. Vistas para Profesionales

`app.views.profesional_views.profesional_home(request)`

Muestra el panel principal del profesional con estadísticas clave.

Incluye las reservas de la semana, el total de reservas del mes, la calificación promedio, y la cantidad de clientes atendidos.

Parámetros: **request** (*HttpRequest*) – Solicitud HTTP.

Devuelve: Renderiza el template “profesional_home.html” con las estadísticas.

Tipo del valor devuelto: HttpResponse

`app.views.profesional_views.dashboard_profesional(request)`

Renderiza el dashboard del profesional con información básica.

Parámetros: **request** (*HttpRequest*) – Solicitud HTTP.

Devuelve: Renderiza el template “dashboard_profesional.html”.

Tipo del valor devuelto: HttpResponse

`app.views.profesional_views.editar_perfil_profesional(request)`

Permite al profesional actualizar su perfil, incluyendo contraseña y datos personales.

Realiza validaciones para evitar duplicados de correo y asegurar que las contraseñas coincidan.

Parámetros: **request** (*HttpRequest*) – Solicitud HTTP.

Devuelve: Renderiza “editar_perfil.html” para GET o redirige tras un POST exitoso.

Tipo del valor devuelto: HttpResponse

`app.views.profesional_views.calendario_reservas(request)`

Renderiza el calendario de reservas del profesional.

Parámetros: **request** (*HttpRequest*) – Solicitud HTTP.

Devuelve: Renderiza el template “calendario_reservas.html”.

Tipo del valor devuelto: HttpResponse

`app.views.profesional_views.editar_disponibilidad(request)`

Permite al profesional cambiar su estado de disponibilidad (activo/inactivo).

Parámetros: **request** (*HttpRequest*) – Solicitud HTTP.

Devuelve: Renderiza “editar_disponibilidad.html” o redirige tras un POST exitoso.

Tipo del valor devuelto: HttpResponse

`app.views.profesional_views.reservas_totales_profesional(request)`
Muestra todas las reservas realizadas para el profesional.
Incluye detalles como usuario y subcategoría.

Parámetros: `request (HttpRequest)` – Solicitud HTTP.

Devuelve: Renderiza “reservas_totales_profesional.html”.

Tipo del valor devuelto: HttpResponse

`app.views.profesional_views.reseñas_profesional(request)`
Muestra todas las reseñas asociadas al profesional, ordenadas por fecha.

Parámetros: `request (HttpRequest)` – Solicitud HTTP.

Devuelve: Renderiza “reseñas_profesional.html”.

Tipo del valor devuelto: HttpResponse

8. Vistas para las Reservas

`app.views.reserva_views.crear_reserva(request)`
Permite a los usuarios crear una nueva reserva.

- Verifica si el cliente ya tiene alguna reserva.
- Aplica un descuento del 20% en la tarifa del servicio si es la primera reserva del cliente.
- Redirige al usuario para confirmar el pago de la reserva.

Parámetros: `request (HttpRequest)` – Solicitud HTTP.

Devuelve: Redirige al usuario a la página de confirmación de pago o muestra un error.

Tipo del valor devuelto: HttpResponse

`app.views.reserva_views.ver_mis_reservas(request)`
Muestra las reservas del usuario (cliente).

Parámetros: `request (HttpRequest)` – Solicitud HTTP.

Devuelve: Renderiza el template “ver_mis_reservas.html” con las reservas del usuario.

Tipo del valor devuelto: HttpResponse

`app.views.reserva_views.eliminar_reserva(request, reserva_id)`
Permite al usuario eliminar una reserva previamente creada.

Parámetros:

- `request (HttpRequest)` – Solicitud HTTP.
- `reserva_id (int)` – ID de la reserva a eliminar.

Devuelve: Redirige a la lista de reservas o muestra una confirmación de eliminación.

Tipo del valor devuelto: HttpResponse

`app.views.reserva_views.reservas_totales(request)`
Muestra todas las reservas del sistema en el panel de administración.

Parámetros: `request (HttpRequest)` – Solicitud HTTP.

Devuelve: Renderiza el template “reservas_totales.html” con todas las reservas.

Tipo del valor devuelto: HttpResponse

```
app.views.reserva_views.confirmar_pago(request, reserva_id)
```

Permite al usuario confirmar el pago de una reserva.

- Cambia el estado de la reserva a “pendiente” después del pago.
- Muestra un mensaje de éxito y redirige al usuario a sus reservas.

Parámetros:

- **request** (*HttpRequest*) – Solicitud HTTP.
- **reserva_id** (*int*) – ID de la reserva a confirmar.

Devuelve: Renderiza la página de confirmación de pago o redirige tras la confirmación.

Tipo del valor devuelto: HttpResponse

```
app.views.reserva_views.resena_profesional(request, profesional_id)
```

Muestra las reseñas de un profesional específico.

Parámetros:

- **request** (*HttpRequest*) – Solicitud HTTP.
- **profesional_id** (*int*) – ID del profesional.

Devuelve: Renderiza el template “reseña.html” con las reseñas del profesional.

Tipo del valor devuelto: HttpResponse

Models en el Proyecto ServiceMatch

Este archivo contiene la documentación de los modelos del proyecto.

1. Modelo Usuario

```
app.models.Usuario(*args, **kwargs)
```

Modelo de usuario extendido para incluir roles y datos personalizados. Los roles disponibles son: - Cliente: Usuario que solicita servicios. - Profesional: Usuario que ofrece servicios específicos.

2. Modelo Servicio

```
app.models.Servicio(*args, **kwargs)
```

Representa un servicio general que pueden ofrecer los profesionales. Ejemplo: «Plomería», «Electricidad», etc.

3. Modelo Subcategoría de Servicio

```
app.models.Subcategoria(*args, **kwargs)
```

Representa una subcategoría de un servicio. Ejemplo: Para el servicio «Plomería», una subcategoría podría ser «Reparación de tuberías».

4. Modelo Reserva

```
app.models.Reserva(*args, **kwargs)
```

Representa una reserva realizada por un cliente para un servicio específico. Las reservas tienen un estado y están asociadas a un cliente y un profesional.

5. Modelo Reseña

```
app.models.Reseña(*args, **kwargs)
```

Representa una reseña realizada por un cliente sobre un profesional. Incluye un comentario, una calificación y la fecha en que se realizó.

Middleware

Este archivo documenta los middlewares personalizados creados para el proyecto. Los middlewares son utilizados para procesar solicitudes y respuestas globalmente, antes de que lleguen a las vistas o después de que se generen.

Bloquear Navegación Middleware

El middleware *BloquearNavegacionMiddleware* bloquea el acceso a ciertas rutas específicas para usuarios no autenticados.

Ubicación: *middleware.py*

Propósito: - Restringir rutas relacionadas con la gestión administrativa. - Redirigir a la página de inicio de sesión si un usuario no autenticado intenta acceder.

Código del Middleware:

```
from django.shortcuts import redirect
from django.urls import reverse, NoReverseMatch

class BloquearNavegacionMiddleware:
    """
    Middleware para bloquear el acceso a ciertas rutas específicas si el usuario no está autenticado.

    - Bloquea rutas relacionadas con la gestión de profesiones y profesionales.
    - Redirige a la página de inicio de sesión si se intenta acceder a estas rutas sin autenticación.

    Uso:
    Añadir 'BloquearNavegacionMiddleware' en la lista MIDDLEWARE del archivo settings.py.
    """
    def __init__(self, get_response):
        """
        Inicializa el middleware.

        Args:
            get_response (callable): La función que se encarga de procesar la respuesta.
        """
        self.get_response = get_response

    def __call__(self, request):
        """
        Intercepta las solicitudes entrantes y verifica si el usuario está autorizado para acceder a ciertas rutas.

        Args:
            request (HttpRequest): Objeto de solicitud HTTP.

        Returns:
            HttpResponse: Respuesta procesada o redirección en caso de no autorización.
        """
        # Lista de rutas específicas que deben estar bloqueadas para usuarios no autenticados
        urls_bloqueadas = []
        try:
            # Agrega rutas estáticas a la lista de URLs bloqueadas.
            urls_bloqueadas.extend([
                reverse('admin_home'),
                reverse('gestionar_profesion'),
                reverse('agregar_profesion'),
                reverse('gestionar_profesionales'),
                reverse('agregar_profesionales')
            ])
        except NoReverseMatch:
            # Ignora rutas que no se puedan resolver en el momento.
```

```

    pass
# Verifica si el usuario está accediendo a rutas dinámicas restringidas.
if not request.user.is_authenticated and (
    request.path.startswith(reverse('actualizar_profesion', kwargs={'servicio_id': 1})) or
    request.path.startswith(reverse('eliminar_profesion', kwargs={'servicio_id': 1})) or
    request.path.startswith(reverse('actualizar_profesional', kwargs={'profesional_id': 1})) or
    request.path.startswith(reverse('eliminar_profesional', kwargs={'profesional_id': 1}))
):
    # Redirige al usuario no autenticado a la página de inicio de sesión.
    return redirect('login')

# Procesa la respuesta normalmente si no se cumple ninguna condición de bloqueo.
response = self.get_response(request)
return response

```

Configuración: Para habilitar este middleware, agréguelo a la lista *MIDDLEWARE* en el archivo *settings.py*:

```

MIDDLEWARE = [
    ...
    'app.middleware.BloquearNavegacionMiddleware',
]

```

Settings del Proyecto

Este archivo documenta las configuraciones principales del proyecto, destacando las opciones más relevantes como la base de datos, las aplicaciones instaladas y la configuración de seguridad.

Configuraciones clave

1. **BASE_DIR** - Define la ruta base del proyecto.

```

"""python BASE_DIR = Path(__file__).resolve().parent.parent
"""

```

2. **SECRET_KEY** - Clave secreta utilizada para la seguridad del proyecto. No debe ser compartida.

```

"""python SECRET_KEY = "django-insecure-r71piut5p==jo3v*xgp%zo)^+o7())vq-^lkgu*0*^il5a01y3"
"""

```

3. **INSTALLED_APPS** - Lista de las aplicaciones instaladas en el proyecto.

```

"""python INSTALLED_APPS = [ "django.contrib.admin", "django.contrib.auth", "app", "drf_yasg",
"django.contrib.humanize", "rest_framework", ] """

```

4. **MIDDLEWARE** - Define los middlewares usados en el proyecto, incluidos los personalizados.

```

"""python MIDDLEWARE = [ "django.middleware.security.SecurityMiddleware",
"app.middleware.BloquearNavegacionMiddleware", # Middleware personalizado # Otros middlewares
estándar... ] """

```

5. **DATABASES** - Configuración de la base de datos, actualmente usando SQLite.

```

"""python DATABASES = { "default": { "ENGINE": "django.db.backends.sqlite3", "NAME": BASE_DIR /
"db.sqlite3", } } """

```

6. **LOGIN_URL** - URL de inicio de sesión personalizada para redirigir cuando sea necesario.

```

"""python LOGIN_URL = "login_profesional" """

```

URLs del Proyecto

Este archivo documenta las rutas y las vistas asociadas a las diferentes secciones del proyecto.

Rutas principales

1. Inicio:

- **Ruta:** `path("", views.inicio, name="inicio")`
- Página de inicio del sitio, vista accesible por todos los usuarios.

2. Autenticación:

- **Ruta:** `path("login/", views.user_login, name="login")`
- Vista de inicio de sesión.
- **Ruta:** `path("register/", views.user_register, name="register")`
- Vista para registro de usuarios.
- **Ruta:** `path("logout/", views.user_logout, name="logout")`
- Vista de cierre de sesión.
- **Ruta:** `path("login_profesional/", views.login_profesional, name="login_profesional")`
- Vista de inicio de sesión para profesionales.

3. Gestión Profesional:

- **Ruta:** `path("gestionar_profesionales/", views.gestionar_profesionales, name="gestionar_profesionales")`
- Vista para gestionar los profesionales en el sistema.
- **Ruta:** `path("agregar_profesionales/", views.agregar_profesional, name="agregar_profesionales")`
- Vista para agregar un nuevo profesional.

4. Gestión de Servicios (Profesión):

- **Ruta:** `path("gestionar_profesion/", views.gestionar_profesion, name="gestionar_profesion")`
- Vista para gestionar los servicios (profesiones).
- **Ruta:** `path("agregar_profesion/", views.agregar_profesion, name="agregar_profesion")`
- Vista para agregar una nueva profesión.

5. Reservas:

- **Ruta:** `path("reservas_totales/", views.reservas_totales, name="reservas_totales")`
- Vista que muestra todas las reservas.
- **Ruta:** `path("crear_reserva/", views.crear_reserva, name="crear_reserva")`
- Vista para crear una nueva reserva.

6. Dashboard Profesional:

- **Ruta:** `path("dashboard_profesional/", views.dashboard_profesional, name="dashboard_profesional")`
- Vista del panel de control para profesionales.
- **Ruta:** `path("calendario_profesional/", views.calendario_reservas, name="calendario_reservas")`
- Vista del calendario de reservas para profesionales.

7. Reseñas y Calificación:

- **Ruta:** `path("reseña/<int:profesional_id>/", views.resena_profesional, name="resena_profesional")`

- Vista para dejar una reseña de un profesional específico.
- **Ruta:** `path("calificar/<int:profesional_id>/", views.calificar_profesional, name="calificar_profesional")`

- Vista para calificar a un profesional.

8 . Estadísticas:

- **Ruta:** `path("estadisticas/", views.obtener_estadisticas_reservas, name="estadisticas_reservas")`

- Vista para obtener estadísticas sobre las reservas.

- **Ruta:** `path("estadisticas/tarjetas/", obtener_estadisticas_tarjetas, name="obtener_estadisticas_tarjetas")`

- Vista para obtener estadísticas de reservas en formato tarjeta.

9 . Rutas para Cliente:

- **Ruta:** `path("cliente_home/", views.cliente_home, name="cliente_home")`

- Página principal para clientes.

- **Ruta:** `path("reservas_totales_cliente/", views.reservas_totales_cliente, name="reservas_totales_cliente")`

- Vista para que los clientes vean todas sus reservas.

Índice

A

[actualizar_cliente\(\)](#) (en el módulo `app.views.cliente_views`)

[actualizar_profesion\(\)](#) (en el módulo `app.views.gestionProfesion_views`)

[actualizar_profesional\(\)](#) (en el módulo `app.views.gestionProfesionales_views`)

[admin_home\(\)](#) (en el módulo `app.views.admin_views`)

[agregar_profesion\(\)](#) (en el módulo `app.views.gestionProfesion_views`)

[agregar_profesional\(\)](#) (en el módulo `app.views.gestionProfesionales_views`)

C

[calendario_reservas\(\)](#) (en el módulo `app.views.profesional_views`)

[calificar_profesional\(\)](#) (en el módulo `app.views.cliente_views`)

[cliente_home\(\)](#) (en el módulo `app.views.cliente_views`)

[confirmar_pago\(\)](#) (en el módulo `app.views.reserva_views`)

[crear_reserva\(\)](#) (en el módulo `app.views.reserva_views`)

D

[dashboard_profesional\(\)](#) (en el módulo `app.views.profesional_views`)

E

[editar_disponibilidad\(\)](#) (en el módulo `app.views.profesional_views`)

[editar_perfil_profesional\(\)](#) (en el módulo `app.views.profesional_views`)

[eliminar_profesion\(\)](#) (en el módulo `app.views.gestionProfesion_views`)

[eliminar_profesional\(\)](#) (en el módulo `app.views.gestionProfesionales_views`)

[eliminar_reserva\(\)](#) (en el módulo `app.views.reserva_views`)

[es_admin\(\)](#) (en el módulo `app.views.admin_views`)

F

[filtrar_reservas\(\)](#) (en el módulo `app.views.filtrarReservas_views`)

G

[gestionar_profesion\(\)](#) (en el módulo `app.views.gestionProfesion_views`)

[gestionar_profesionales\(\)](#) (en el módulo `app.views.gestionProfesionales_views`)

L

[login_profesional\(\)](#) (en el módulo `app.views.authProfesional_views`)

O

[obtener_estadisticas_reservas\(\)](#) (en el módulo `app.views.estadisticas_views`)

[obtener_estadisticas_tarjetas\(\)](#) (en el módulo `app.views.estadisticas_views`)

P

[profesional_home\(\)](#) (en el módulo `app.views.profesional_views`)

R

[resena_profesional\(\)](#) (en el módulo `app.views.reserva_views`)

[Reseña\(\)](#) (en el módulo `app.models`)

[reseñas_profesional\(\)](#) (en el módulo `app.views.profesional_views`)

[Reserva\(\)](#) (en el módulo `app.models`)

[reservas_totales\(\)](#) (en el módulo `app.views.reserva_views`)

[reservas_totales_cliente\(\)](#) (en el módulo `app.views.cliente_views`)

[reservas_totales_profesional\(\)](#) (en el módulo `app.views.profesional_views`)

S

[Servicio\(\)](#) (en el módulo `app.models`)

[Subcategoria\(\)](#) (en el módulo `app.models`)

T

[terminos\(\)](#) (en el módulo `app.views.auth_views`)

U

[user_login\(\)](#) (en el módulo `app.views.auth_views`)

[user_logout\(\)](#) (en el módulo `app.views.auth_views`)

[user_register\(\)](#) (en el módulo `app.views.auth_views`)

[Usuario\(\)](#) (en el módulo `app.models`)

V

validar_dia_habil() (en el módulo
app.views.gestionProfesion_views)

validar_disponibilidad() (en el módulo
app.views.gestionProfesion_views)

validar_fecha_futura() (en el módulo
app.views.gestionProfesion_views)

validar_subcategoria() (en el módulo
app.views.gestionProfesion_views)

ver_mis_reservas() (en el módulo
app.views.reserva_views)