Texto

Descripción generada automáticamente

Arquitectura del sistema - API

Personas sentadas en una mesa

Descripción generada automáticamente con confianza media

# Editor:

Reinaldo Espinel Torres

Plataforma de Orquestación para Automatizaciones RPA

**26/05/2024**

# Detalles del documento

|  |  |
| --- | --- |
| Seguridad | Publico |
| Version | Version 1.1 |
| Editor | Reinaldo Espinel Torres |

# Resumen

Este documento incluye los requerimientos del Sistema.

Contenido

[Editor: 1](#_Toc172130592)

[Detalles del documento 2](#_Toc172130593)

[Resumen 3](#_Toc172130594)

[1.1. Introducción 4](#_Toc172130595)

[2. Descripción General de la Arquitectura 5](#_Toc172130596)

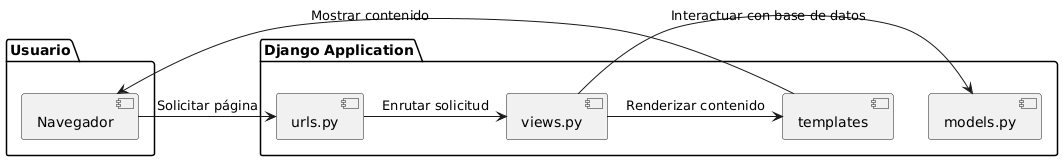
# Introducción

Este documento describe la arquitectura del orquestador de API desarrollado en Django. Incluye una visión general del sistema, diagramas de arquitectura, componentes principales, interacciones y especificaciones técnicas detalladas.

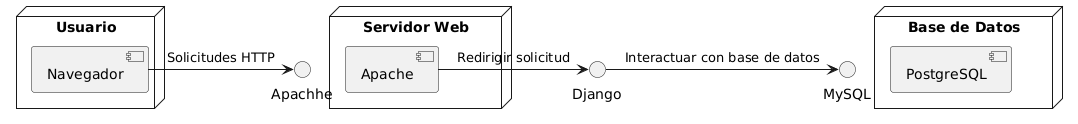
# Descripción General de la Arquitectura

La arquitectura del sistema se basa en un modelo de aplicación web típica, estructurada en varios componentes y capas. El siguiente diagrama proporciona una visión general de los principales componentes del sistema y sus interacciones.

**Diagrama de componentes**



**Diagrama de despliegue**



# Componentes del Sistema

A continuación, se describen los principales componentes del sistema y sus responsabilidades.

**Ejecuciones\_api**

* **settings.py**: Archivo de configuración principal de Django que gestiona las configuraciones globales del proyecto.
* **urls.py**: Define las rutas principales del proyecto y las asigna a las vistas correspondientes.
* **wsgi.py**: Configuración de la interfaz WSGI para el despliegue de la aplicación en servidores web.
* **asgi.py**: Configuración de la interfaz ASGI para aplicaciones asíncronas.

**control\_automations**

* **views.py**: Contiene la lógica de manejo de las solicitudes HTTP, gestionando las operaciones CRUD para los recursos.
* **models.py**: Define la estructura y las relaciones de la base de datos mediante modelos de Django.
* **serializers.py**: Convierte instancias de los modelos a otros formatos como JSON y viceversa.
* **urls.py**: Define las rutas específicas para la aplicación control\_automations y las asigna a las vistas correspondientes.
* **migrations/**: Contiene los archivos de migración para la gestión de cambios en la base de datos.

**Functions.py**

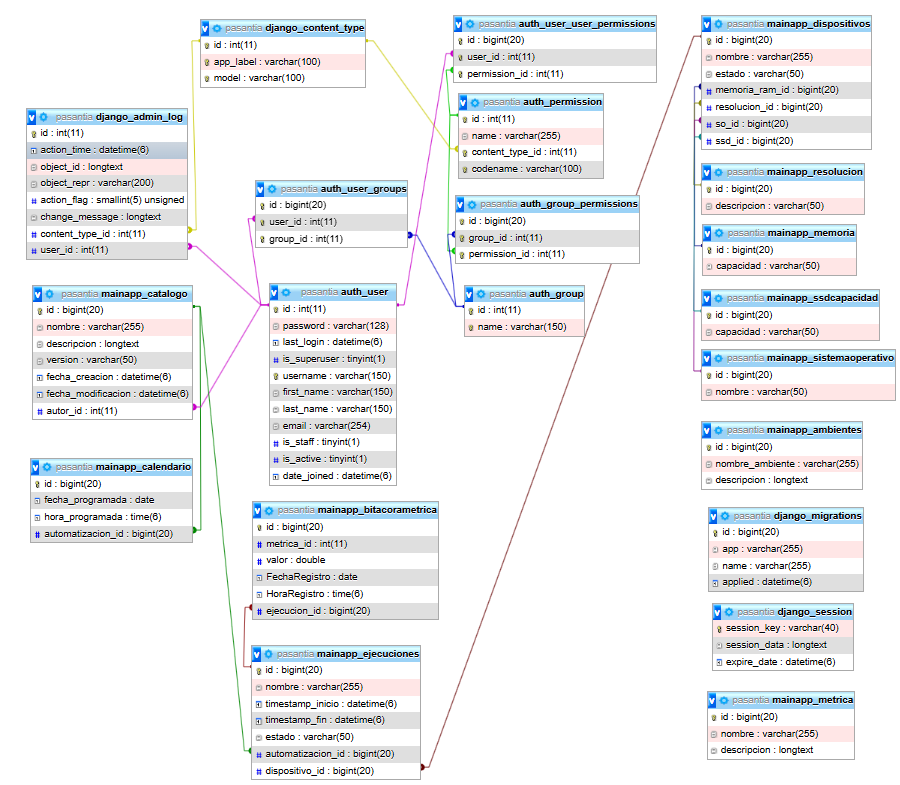
* **Functions.py**: Este archivo contiene funciones auxiliares que encapsulan lógica reutilizable y tareas específicas que se utilizan en diferentes partes del proyecto. Estas funciones son:
  + **Funciones de Validación**: Validan datos antes de procesarlos o almacenarlos en la base de datos.
  + **Funciones de Utilidad**: Realizan tareas comunes, como formateo de datos, cálculos específicos, o conversión de formatos.
  + **Funciones de Integración**: Manejan la comunicación con servicios externos, APIs de terceros, o microservicios internos.
  + **Funciones de Transformación de Datos**: Transforman datos entre diferentes estructuras o formatos según sea necesario para la lógica de negocio.

# Especificaciones técnicas

**Tecnologías utilizadas:**

* **Lenguaje de Programación**: Python
* **Framework**: Django
* **Servidor**: Apache (para servir tanto la aplicación como el servidor web).
* **Base de Datos**: MySQL

# Esquema de Base de Datos



# Diagrama de secuencia

