MANUAL CENTRAL SUITES FERIAL

Python: Descripción General

Python es un lenguaje de programación de alto nivel, interpretado y de propósito general. Fue creado por **Guido van Rossum** y lanzado por primera vez en 1991. Es conocido por su simplicidad y legibilidad, lo que facilita a los desarrolladores escribir, leer y mantener el código. Python es una excelente elección tanto para principiantes como para profesionales debido a su versatilidad y comunidad activa.

Características Principales de Python

1. **Sintaxis Clara y Legible:**
   - Diseñado para ser fácil de aprender y entender, con una sintaxis similar al lenguaje humano.
2. **Multipropósito:**
   - Adecuado para aplicaciones web, ciencia de datos, automatización, inteligencia artificial, desarrollo de software, y más.
3. **Extensa Biblioteca Estándar:**
   - Ofrece módulos pre construidos para tareas comunes como manejo de archivos, HTTP, matemáticas avanzadas, o manipulación de datos.
4. **Lenguaje Multiplataforma:**
   - Funciona en diferentes sistemas operativos, como Windows, macOS y Linux.
5. **Dinámicamente Tipado:**
   - No es necesario declarar el tipo de variable; Python lo infiere automáticamente en tiempo de ejecución.
6. **Ampliable y Flexible:**
   - Puede integrarse con otros lenguajes (como C/C++, Java) para soluciones de alto rendimiento.

Django: Descripción General

Django es un framework web de alto nivel escrito en Python que facilita el desarrollo rápido y limpio de aplicaciones web. Fue diseñado por el equipo de desarrolladores de la organización **Lawrence Journal-World** y lanzado por primera vez en 2005. Django sigue el principio de diseño **DRY (Don't Repeat Yourself)**, promoviendo la reutilización del código y el desarrollo ágil.

Características Principales de Django

1. **Framework "Baterías Incluidas":**
   - Django ofrece soluciones integradas para casi todos los aspectos del desarrollo web: autenticación, bases de datos, formularios, enrutamiento, y más.
2. **ORM (Object-Relational Mapping):**

    - Simplifica las operaciones con bases de datos, permitiendo interactuar con ellas usando modelos Python en lugar de SQL directamente.
3. **Arquitectura MVC (Model-View-Controller):**
    - Organiza el proyecto en tres componentes principales: Modelos, Vistas y Plantillas.
4. **Seguridad Incorporada:**
    - Incluye medidas contra ataques comunes como XSS, CSRF, inyección SQL, entre otros.
5. **Escalabilidad:**
    - Diseñado para manejar aplicaciones de pequeño a gran tamaño con facilidad.
6. **Versatilidad:**
    - Usado para construir blogs, sistemas de e-commerce, APIs RESTful, redes sociales, entre otros.
7. **Enfoque en el Desarrollo Ágil:**
    - Reduce el tiempo de desarrollo gracias a las herramientas integradas y a una comunidad activa.

Relación entre Python y Django

Django es una extensión poderosa del lenguaje Python que proporciona las herramientas necesarias para el desarrollo web estructurado, robusto y eficiente. Mientras Python maneja la lógica general del programa, Django agrega características específicas para trabajar con aplicaciones web de manera segura y escalable.

Implementación de Proyecto con Django 3.2

1. Preparación del Entorno

- **Sistema Operativo:** Ubuntu Server 20.04 (recomendado).
- **Python Version:** Python 3.9 (compatible con Django 3.2).
- **Base de Datos:** PostgreSQL o MySQL.
- **Servidor Web:** Gunicorn con Nginx (para producción).
- **Administración del entorno:** Utilizar un entorno virtual venv.
- Instalar dependencias usando el archivo requirements.**txt**

Implementación del servidor.

En la carpeta service encontramos los servicios a usar para instalar el proyecto en línea. usando system y nginx. mas
info: https://www.digitalocean.com/community/tutorials/how-to-set-up-django-with-postgres-nginx-and-gunicorn-on-ubuntu Podemos apoyarnos en esta documentación el cual tiene todo lo necesario.

Github proyecto.

https://github.com/OMMDIGITAL202/HOTELESSUITEFERIA

Explicación de scripts de búsquedas, booking y suites feria.

Este código implementa un script para realizar búsquedas de reservas en un sitio web usando Selenium y Python. El flujo de trabajo se divide en varias partes clave:

1. **Selección del controlador del navegador**:
    - Dependiendo del sistema operativo (`platform.system()`), el script selecciona el controlador de **Chrome** o **Firefox** con opciones preconfiguradas, como `headless` para evitar la apertura de una ventana visible del navegador.
2. **Acción en la página web**:
    - **Aceptar cookies**: Intenta encontrar y hacer clic en un botón para aceptar las cookies, si está disponible.
    - **Búsqueda de alojamiento**: El script ingresa el término de búsqueda (por ejemplo, el nombre de una ciudad o lugar) en un campo de texto y luego presiona "Enter" para realizar la búsqueda.
    - **Fechas de reserva**: Las fechas de check-in y check-out son manejadas en la URL, reemplazando las fechas actuales en la misma.
3. **Interacción con los elementos**:
    - Se intentan varios clics en botones, como el de aceptar información sobre el inicio de sesión, la selección de la clasificación de estrellas (por ejemplo, hoteles de 3 estrellas), la selección de la categoría de "Hoteles", y los precios.
    - Utiliza Selenium para hacer clic en ciertos botones y configurar filtros de búsqueda.
4. **Rastreo de la página y gestión de datos**:
    - Utiliza BeautifulSoup para obtener el HTML de la página y extraer información relevante (por ejemplo, las opciones de clasificación por estrellas).
    - Además, guarda información relacionada con el número total de resultados de búsqueda.
5. **Proceso de búsqueda cíclico**:
    - El script se ejecuta en un bucle mientras la variable `process` no esté desactivada, realizando la búsqueda y esperando los tiempos necesarios entre interacciones.

Errores comunes que maneja el código:

- El script maneja excepciones comunes como `NoSuchElementException` y `ElementClickInterceptedException`, que ocurren cuando los elementos no se encuentran o cuando un clic es interceptado.

Mejoras potenciales:

- La implementación podría beneficiarse de manejar de manera más eficiente la pausa y sincronización entre las interacciones, ya que el `sleep` puede no ser ideal para manejar tiempos de espera.

Este tipo de scripts suelen ser muy útiles para la automatización de búsquedas web, como en el caso de la búsqueda de hoteles u otros servicios en plataformas en línea.

El código proporcionado implementa un conjunto de funciones en Python que gestionan procesos automatizados para actualizar la disponibilidad de suites en una feria y realizar búsquedas de reservas en un sistema. A continuación, se describe el funcionamiento general de las dos funciones principales:

---

1. `active_process_sf`: Gestión de disponibilidad de Suites Feria

- **Propósito**: Sincroniza la disponibilidad de suites en un evento llamado "Feria".
- **Pasos principales**:
    1. **Inicio de sesión y obtención de datos**:
        - Se conecta a un servicio mediante un objeto `SuitesFeria`.
        - Inicia sesión y obtiene la disponibilidad actual con `suites_feria.disponibilidad()`.
    2. **Almacenamiento en la base de datos**:
        - Revisa o crea registros en el modelo `AvailSuitesFeria` y `CantAvailSuitesFeria` para cada fecha y tipo de disponibilidad obtenida.
        - Actualiza los valores si los registros ya existen.
    3. **Cierre de sesión**:
        - Finaliza la sesión del servicio después de actualizar los datos.
    4. **Condición de terminación**:
        - Verifica si algún proceso en `ProcessActive` tiene la señal para finalizar (`currenct` o `active`).
    5. **Espera y reintento**:
        - Si no hay error, espera 30 segundos antes de la siguiente iteración. En caso de error, registra el problema y sigue intentando.

---

2. `active_process`: Gestión de procesos generales

- **Propósito**: Coordina varios procesos relacionados con búsquedas y sincronización de datos.
- **Pasos principales**:
    1. **Limpieza de datos antiguos**:
        - Elimina registros de `AvailableBooking` cuya fecha sea más antigua que 10 días.
    2. **Preparación**:
        - Activa procesos pendientes (`ProcessActive`) relacionados con búsquedas generales (`GeneralSearch`).
        - Inicia instancias de `BookingSearch` y sus respectivos drivers.
    3. **Ejecución concurrente**:
        - Usa hilos para ejecutar controladores asociados a cada proceso, buscando información en diferentes navegadores o fuentes.

4. **Procesos secundarios**:
    - Activa búsquedas con nombres de hoteles en otro conjunto de procesos (`GeneralSearch.type_search=2`).
5. **Gestión de ciclos y espera**:
    - Al terminar un ciclo de ejecución, verifica tiempos de espera definidos en la configuración (o usa un valor predeterminado de 3 minutos).
    - Repite la ejecución de manera indefinida, controlando el estado de los procesos.

Resumen

- El primer proceso (`active_process_sf`) se enfoca en sincronizar la disponibilidad de suites, interactuando con un sistema externo y gestionando los datos en la base de datos local.
- El segundo proceso (`active_process`) administra tanto las búsquedas de reservas generales como las específicas por nombre, ejecutándolas de forma concurrente y coordinando diferentes procesos en ciclos definidos.

Ambos procesos incluyen manejo de excepciones y registran información clave para diagnóstico en caso de errores. La lógica de control asegura que las tareas se realicen de manera eficiente y escalable mediante el uso de hilos.

Modelo de base de datos y estructura.

1. Booking

- **Propósito**: Representa una reserva, almacenando detalles como el título, dirección y enlace de la reserva.
- **Campos**:
    - start: Hora o inicio de la reserva.
    - title: Título de la reserva.
    - link: Enlace relacionado con la reserva.
    - address: Dirección de la reserva.
    - distance: Distancia o ubicación relevante.
    - description: Descripción detallada.
    - img: URL de la imagen asociada.
    - updated y created: Fechas de última actualización y creación.
- **Uso**:
    - Añadir registros con nuevos datos de reservas.
    - Visualizar todas las reservas en el panel admin.
- **Cómo Usar en Admin**:
    - **Crear**: Proporciona los campos requeridos en el formulario de admin.

    - **Editar**: Accede a un registro y modifica la información deseada.
    - **Eliminar**: Selecciona la reserva desde el listado y elimina.

---

2. Complement

- **Propósito**: Datos complementarios relacionados con búsquedas y ocupación.
- **Campos**:
    - total_search: Número total de búsquedas.
    - occupancy: Capacidad u ocupación de la búsqueda.
    - start: Hora o inicio.
    - date_from, date_to: Intervalo de fechas.
    - updated y created: Tiempos de actualización y creación.
- **Uso**:
    - Estadísticas sobre ocupación y fechas relevantes.
- **Cómo Usar en Admin**:
    - Igual a **Booking**.

---

3. AvailableBooking

- **Propósito**: Define la disponibilidad de reservas en rangos de fecha.
- **Campos**:
    - Incluye referencias a Booking y Complement.
    - date_from, date_to: Fechas disponibles.
    - price: Precio de la reserva.
    - active: Indica si está activo.
    - Otros: total_search, occupancy, start, etc.
- **Uso**:
    - Relaciona disponibilidad con reservas.
    - Proporciona precios y datos específicos por fechas.
- **Cómo Usar en Admin**:
    - Sigue el mismo proceso de gestión en admin para relaciones.

---

4. ProcessActive

- **Propósito**: Define procesos activos con opciones como tipo de búsqueda y posiciones.
- **Campos**:
    - date_end: Fecha de finalización.
    - occupancy, start: Datos de ocupación y comienzo.
    - position: Información en formato JSON.
    - type_proces: Tipo de proceso (Ciudad o Nombre).
- **Uso**:
    - Configuración de procesos de búsqueda activos.
- **Cómo Usar en Admin**:

    - JSON necesita formato válido al editar campos.

---

5. GeneralSearch

- **Propósito**: Datos generales para una búsqueda.
- **Campos**:
    - url: URL base para la búsqueda.
    - city_and_country: Ciudad y país objetivo.
    - time_sleep_minutes: Tiempo de espera en minutos.
    - type_search: Tipo de búsqueda.
    - proces_active: Relación con procesos activos.
- **Uso**:
    - Información básica sobre el proceso de búsqueda.
- **Cómo Usar en Admin**:
    - Añadir configuraciones de búsqueda básicas.

---

6. AvailSuitesFeria

- **Propósito**: Fechas específicas de disponibilidad de suites.
- **Campos**:
    - date_avail: Fecha disponible.
- **Uso**:
    - Registro de disponibilidad específica.
- **Cómo Usar en Admin**:
    - Gestor simple para fechas.

---

7. CantAvailSuitesFeria

- **Propósito**: Cantidades relacionadas con AvailSuitesFeria.
- **Campos**:
    - type_avail: Tipo de disponibilidad.
    - avail: Número de habitaciones disponibles.
    - Relación con AvailSuitesFeria.
- **Uso**:
    - Gestión detallada de capacidades por fecha.
- **Cómo Usar en Admin**:
    - Relacionar correctamente con AvailSuitesFeria.

---

8. TemporadaByDay

- **Propósito**: Define colores y texto para diferentes días de temporada.
- **Campos**:

    - bg_color, text_color: Colores asociados.
    - number: Número identificativo.
    - date_from, updated, created: Fechas relacionadas.
- **Uso**:
    - Decoración y marcado de temporadas en frontend.
- **Cómo Usar en Admin**:
    - Asegurar las opciones válidas para colores.

---

9. Price

- **Propósito**: Precios por fechas y ocupación.
- **Campos**:
    - date_from: Fecha de inicio.
    - price: Valor del precio.
    - occupancy: Ocupación relacionada.
- **Uso**:
    - Información relevante para precios históricos.
- **Cómo Usar en Admin**:
    - Relacionar ocupación correctamente.

AvailWithDate

Descripción

El modelo AvailWithDate se utiliza para registrar información relacionada con la disponibilidad de un elemento específico en una fecha dada. Es adecuado para gestionar datos con fechas asociadas y la capacidad de actualizarlos con marcas de tiempo.

Atributos del Modelo

Campos

1. **date_from**
    - **Tipo:** CharField
    - **Máxima longitud:** 30 caracteres
    - **Descripción:** Indica la fecha desde la cual está disponible el elemento. Se guarda en formato de texto.
2. **avail**
    - **Tipo:** CharField
    - **Máxima longitud:** 50 caracteres

    - **Descripción:** Representa la disponibilidad del elemento asociado.
3. **updated**
    - **Tipo:** DateTimeField
    - **Valor por defecto:** null=True, blank=True
    - **Descripción:** Guarda la fecha y hora de la última modificación del registro. Este campo es opcional.
4. **created**
    - **Tipo:** DateTimeField
    - **Valor por defecto:** null=True, blank=True
    - **Descripción:** Representa la fecha y hora de la creación del registro. Este campo es opcional.

Métodos

- **__str__()**
    - **Descripción:** Método que representa al modelo como una cadena de texto legible.
    - **Salida:** Una concatenación del valor de avail y date_from, separada por el carácter |.
    - **Ejemplo de salida:**
      "Disponible | 2024-12-20"

MessageByDay

Descripción

El modelo MessageByDay se utiliza para gestionar mensajes asociados a fechas específicas y un número determinado de ocupantes. Es adecuado para aplicaciones relacionadas con comunicación o notificaciones personalizadas basadas en ocupación y tiempo.

Atributos del Modelo

Campos

1. **date_from**
    - **Tipo:** CharField
    - **Máxima longitud:** 30 caracteres
    - **Descripción:** Fecha a la que está asociado el mensaje, guardada como una cadena de texto.
2. **occupancy**

    - **Tipo:** IntegerField
    - **Elecciones disponibles:**
        - 2: "2 Personas"
        - 3: "3 Personas"
        - 5: "5 Personas"
    - **Valor por defecto:** 2
    - **Descripción:** Indica la cantidad de personas asociadas al mensaje.
3. **text**
    - **Tipo:** CharField
    - **Máxima longitud:** 512 caracteres
    - **Descripción:** Contenido principal del mensaje.
4. **updated**
    - **Tipo:** DateTimeField
    - **Valor por defecto:** null=True, blank=True
    - **Descripción:** Registra la fecha y hora de la última modificación del registro.
5. **created**
    - **Tipo:** DateTimeField
    - **Valor por defecto:** null=True, blank=True
    - **Descripción:** Registra la fecha y hora de la creación del registro.

Métodos

- **__str__()**
    - **Descripción:** Método que convierte el modelo en una cadena de texto legible.
    - **Salida:** El contenido del campo text.
    - **Ejemplo de salida:**
      "Recordatorio: Confirmar asistencia"

EventByDay

Descripción

El modelo EventByDay se utiliza para registrar eventos asociados a fechas específicas y una cantidad específica de ocupantes. Este modelo es ideal para aplicaciones relacionadas con la planificación y organización de eventos según la ocupación.

Atributos del Modelo

Campos

(Sus atributos son idénticos a MessageByDay, pero destinados a eventos en lugar de mensajes).

1. **date_from**
    - **Tipo:** CharField
    - **Descripción:** Fecha asociada al evento.
2. **occupancy**
    - **Tipo:** IntegerField
    - **Elecciones disponibles:**
        - 2: "2 Personas"
        - 3: "3 Personas"
        - 5: "5 Personas"
    - **Valor por defecto:** 2
3. **text**
    - **Tipo:** CharField
    - **Descripción:** Contenido o descripción del evento.
4. **updated**
    - **Tipo:** DateTimeField
5. **created**
    - **Tipo:** DateTimeField

Métodos

- **__str__()**
    - **Salida:** Contenido del campo text.
    - **Ejemplo de salida:**
      "Evento especial para grupos grandes"

CopyPriceWithDay

Descripción

Este modelo relaciona precios con un objeto de reserva disponible (AvailableBooking), manejando precios para días específicos.

Atributos del Modelo

Campos

1. **price**
    - **Tipo:** CharField
    - **Máxima longitud:** 30 caracteres
    - **Descripción:** Precio asociado a una reserva disponible.
2. **created**
    - **Tipo:** DateField
    - **Valores permitidos:** null=True, blank=True
    - **Descripción:** Fecha de creación del precio.

3. **avail_booking**
    - **Tipo:** ForeignKey
    - **Relacionado con:** AvailableBooking
    - **Acción al eliminar:** CASCADE
    - **Descripción:** Enlace con el objeto de reserva disponible.

Métodos

- **__str__()**
    - **Descripción:** Devuelve el precio en forma de cadena.
    - **Salida: 120**

PriceWithNameHotel

Descripción

Modelo utilizado para gestionar precios y detalles de hoteles en un rango de fechas, con información adicional como dirección, descripción, imágenes y ocupación.

Atributos del Modelo

Campos

1. **start**
    - **Tipo:** CharField
    - **Máxima longitud:** 20 caracteres
    - **Descripción:** Punto de inicio de referencia para el hotel.
2. **title**
    - **Tipo:** CharField
    - **Máxima longitud:** 512 caracteres
    - **Descripción:** Nombre del hotel.
3. **link**
    - **Tipo:** CharField
    - **Máxima longitud:** 3000 caracteres
    - **Descripción:** Enlace al detalle del hotel.
4. **address**, **distance**, **description**, **img**, etc.
    - **Tipo:** CharField
    - **Máxima longitud:** Variada según el atributo.
    - **Descripción:** Información relevante del hotel (dirección, distancia, descripción e imágenes).
5. **date_from, date_to**
    - **Tipo:** CharField
    - **Máxima longitud:** 30 caracteres
    - **Descripción:** Fechas de inicio y fin.
6. **price**

    - **Tipo:** CharField
    - **Máxima longitud:** 30 caracteres
    - **Descripción:** Precio asociado.
7. **occupancy**
    - **Tipo:** IntegerField
    - **Valor por defecto:** 2
    - **Descripción:** Cantidad de personas asociadas al precio.

CopyPriceWithNameFromDay

Descripción

Modelo para relacionar precios con información más detallada del hotel (PriceWithNameHotel).

Atributos del Modelo

1. **price**
2. **created**
3. **avail**
    - Relación con el modelo PriceWithNameHotel.

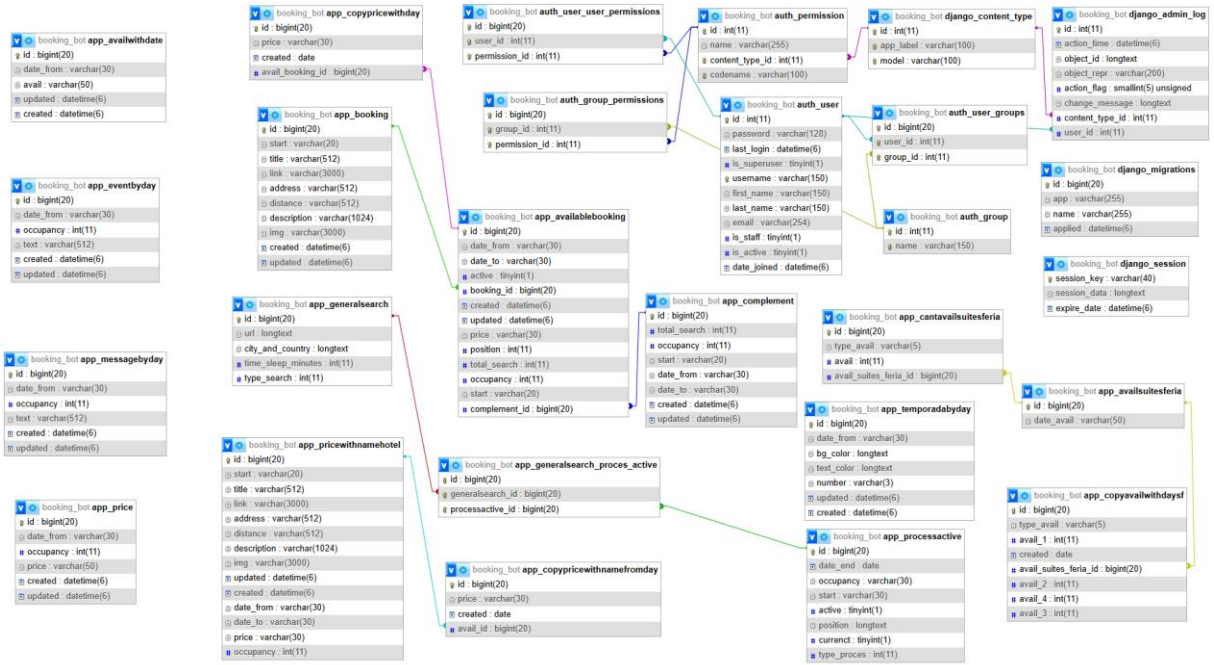
CopyAvailWithDaySF

Descripción

Modelo para gestionar disponibilidad específica por tipo y categorías para Suites de Feria.

Atributos del Modelo

1. **type_avail**
2. **avail_1**, **avail_2**, etc.
3. **avail_suites_feria**



Configuraciones search.

1. vamos al admin django.
2. buscamos la opcion General searchs.
3. agregar general search.
    - url: https://www.booking.com/searchresults.es.html?
    - city and country: Madrid, Comunidad de Madrid, España
    - time sleep minutes: 10 or > 0

Configuraciones Process.

1. vamos al admin django.
2. buscamos la opcion Process actives.
3. agregamos los procesos.
    - Date end: 2024-09-15
    - Occupancy: 3
    - Start: 4

- Position: [0, 1, 2, 3, 4, 9]
- Active: Descheck
- Current: Descheck

4. agregaremos 5 procesos iguales. al anterior cambiando algunos parametros.
   - Occupancy: 2
   - Start: 4
   - Position: [0, 1, 2, 3, 4, 9, 14, 19, 24]

- Occupancy: 2
- Start: 3
- Position: [0, 1, 2, 3, 4]

- Occupancy: 3
- Start: 3
- Position: [0, 1, 2, 3, 4]

- Occupancy: 5
- Start: 3
- Position: [0, 1, 2, 3, 4]

- Occupancy: 5
- Start: 4
- Position: [0, 1, 2, 3, 4, 9]