HINTS

CONSEJOS CONCEPTUALES

- El sistema de autenticación de Django no contempla ciertas características que son comúnmente encontradas en otros sistemas de autenticación, tales como: chequeo de contraseñas seguras, control de intentos de inicio de sesión, autenticación frente a terceros ni permisos a nivel de objetos. Estas soluciones se deben implementar con paquetes externos.

- La configuración necesaria para hacer uso de la autenticación de Django se establece al momento de crear la aplicación, utilizando el comando: `django-admin startproject`.

- Las tablas de la base de datos para el manejo de usuarios y el modelo de permisos es creada la primera vez que se ejecuta: `python manage.py migrate`

- El formato básico de una URL que se incluye en `urlpatterns` tiene las siguientes características:
  - Expresión regular: que no es más que una cadena de caracteres.
  - Función de vista de vistas: generalmente una función de vista o una cadena que especifica la ruta de la función de vista.
  - Parámetros: parámetros predeterminados opcionales que se pasarán a la función de vista.
  - Alias: nombre opcional.

```
urlpatterns = [url (expresión regular, función de vista, parámetros, alias)]
```

- Es cierto que Django brinda una buena protección de seguridad, sin embargo, es importante implementar correctamente su aplicación y aprovechar la protección de seguridad del servidor web, el sistema operativo y otros componentes.

- Es bueno asegurarse de que el código Python esté fuera del directorio root del servidor web.

- Estar atento a los archivos cargados por los usuarios, si es el caso es una buena medida de seguridad de una aplicación web.

- Django no acelera las solicitudes para autenticar a los usuarios. Para protegerse contra los ataques de fuerza bruta contra el sistema de autenticación, puede considerar implementar un complemento de Django, o también un módulo de servidor web para acelerar estas solicitudes.

- Las migraciones son compatibles con todos los backends con los que trabaja Django, así como con backends de terceros si se han programado para admitir la alteración del esquema (realizado a través de la clase `SchemaEditor`). Sin embargo, algunas bases de datos son más capaces que otras cuando se trata de migraciones de esquemas. Por ejemplo: PostgreSQL es el que mayor compatibilidad con esquemas ofrece.

- Las migraciones se ejecutarán de la misma manera en el mismo conjunto de datos, y producirán resultados consistentes, lo que significa que lo que ve en el desarrollo y la preparación es, bajo las mismas circunstancias, exactamente lo que sucederá en la producción.

- Django realizará migraciones para cualquier cambio en sus modelos o campos, incluso las opciones que no afectan la base de datos, ya que la única forma en que puede reconstruir un campo correctamente es tener todos los cambios en el historial, y es posible que necesite esas opciones en algunas migraciones de datos más adelante (por ejemplo: si ha configurado validadores personalizados).