Diseño del módulo Permisología

Diseño de arquitectura de la librería Auditorías Turpial incluyendo el módulo de Permisología.

En el estado actual de la librería Auditorías Turpial consta de tres módulos: *Core*, Estadísticas y Reportes. Estos dos últimos, son un microservicio que ofrece facilidades para presentar los datos recaudados por el módulo *Core*.

El módulo Permisología debe adaptarse a la arquitectura existente del proyecto, por ende, se presenta como otro microservicio. Su única propósito es proveer un conjunto de funciones que sean capaces de restringir el acceso o impedir ciertas acciones sobre las auditorías del sistema que lo instale.

Al ser un microservicio, el programador puede decidir no instalarlo, de esta manera se mantiene el funcionamiento de la librería presentado hasta ahora. El módulo Permisología, depende directamente del *Core*, puesto que necesita agregar los permisos que corresponden, sobre dicha base de datos. En la figura 1 se muestra el diseño de la arquitectura planteada con la adición de este módulo.

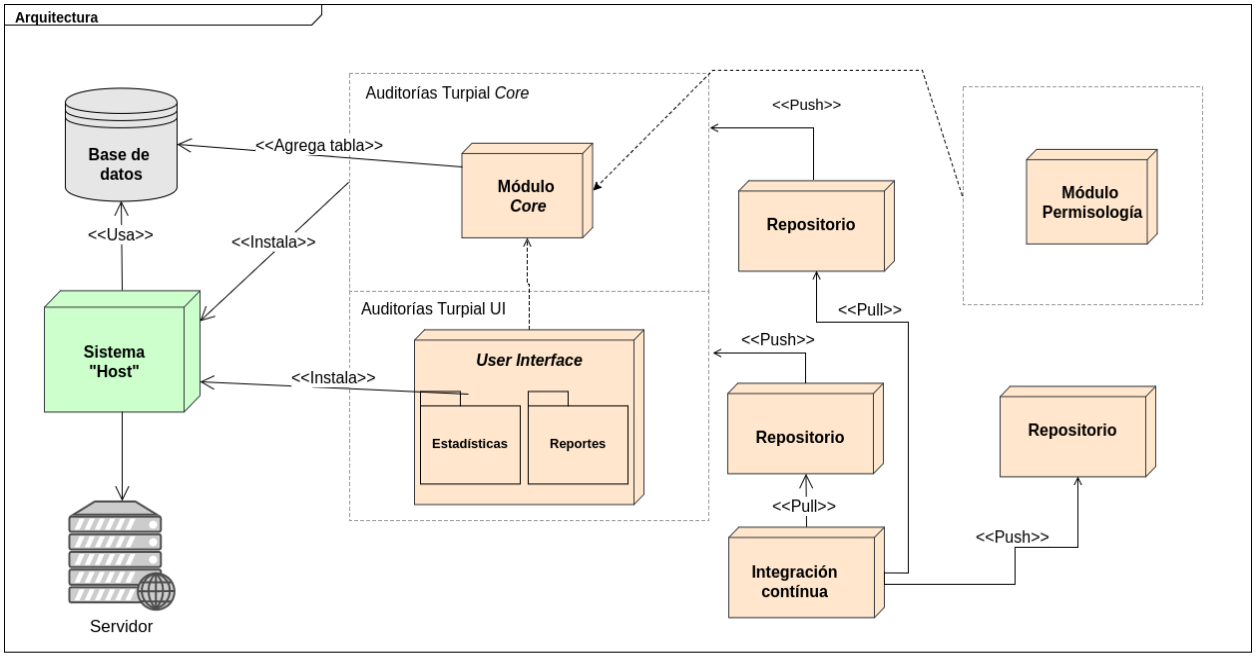

Figura 1. Arquitectura de la librería Auditorías Turpial incluyendo el módulo Permisología.

Creación de permisos

Django, crea nativamente una tabla de Permisos en la cual el programador puede agregar objetos cuando necesite nuevos permisos. Para esto, debe crear un objeto con al menos un código, una descripción y a que modelo pertenece.

El módulo Permisología, debe disponer del código que permita generar nuevos permisos en esta base de datos y que estén asociados con la tabla "AuditableAction". Para facilitar la inclusión de los mismo, debe crearse una migración que automatice su inclusión al momento de instalar el módulo.

El manejo de la inclusión de un permiso particular a un usuario, puede realizarse a traves del sitio administrador de Django. Adicionalmente, se puede incluir un Group, Auditor, que incluya ciertos permisos por defecto.

Tomando en cuenta las Historias de Usuario, algunos de los permisos de usuario podrían ser:

- Puede ver listados de auditoría.
- Puede generar reportes CSV.
- Puede generar reportes PDF.
- Puede visualizar las estadísticas.
- Puede visualizar el usuario que realizó alguna acción.
- Puede visualizar el modelo sobre el cual se realizó la acción.
- Puede inspeccionar los registros creados.
- Puede inspeccionar los registros actualizados.
- Puede inspeccionar los registros eliminados.

***Mixin* de para el control de los permisos.**

Django posee una serie de *mixins* que permiten de alguna manera, restringir las funcionalidades sobre una vista particular. El "PermissionRequiredMixin" verifica si un usuario cuenta con todos los permisos definidos en la variable "permissions_required", la cual es un iterable (lista o tupla). Si el usuario no posee alguno de estos, se envía un *response Forbbiden* (403). Si no está autenticado, redirige a la página de "iniciar sesión".

Se pueden generar dos *mixins* que hereden "PermissionRequiredMixin" para manejar los casos en los que se desean restringir los listados y las estadísticas, puesto que estas prohiben el acceso al URL.

Existe otro *mixin* de Django que proporciona parte de la lógica requerida para el resto de las funcionalidades, “AccessMixin”. Este puede ser usado para redirigir a la página de autenticación o levantar una excepción indicando que el usuario no posee los permisos si el atributo “raise_exception” es igual a True.

Para los permisos de generar reportes, se puede heredar el *mixin* anterior, colocando False el atributo “raise_exception” puesto que la intención no es enviar una repuesta *Forbidden* sino agregar en el contexto de la vista una variable que oculte el botón para descargar el reporte, según sea el caso.

El “AccessMixin” puede ser heredado para manejar el resto de los permisos. Ese *mixin* procesaría el *queryset*, eliminando la información que está restringida o evitando que ciertas variables sean agregadas al contexto de la vista que lo hereden.