

ISS-08 — Client con ModelViewSet

Publicación del CRUD completo de clientes con ModelViewSet, SimpleRouter y Baja Lógica en acceso OPEN

StoreLab REST API — Arquitectura de Software Backend

¿Qué construiremos en el ISS-08?

Objetivo Central del ISS-08:

- **CRUD Homogéneo para Recursos:** Exponer las operaciones REST completas (list, create, retrieve, update, partial_update, destroy) para la entidad Client usando ModelViewSet.
- **Acceso Abierto (OPEN - Fase I):** Configurar permission_classes = [AllowAny] antes de introducir la autenticación JWT y RBAC en el ISS-22.
- **Baja Lógica Personalizada:** Sobrescribir perform_destroy() para evitar la eliminación física de la base de datos y cambiar status a RecordStatus.INACTIVE.
- **Enrutamiento Limpio con SimpleRouter:** Usar SimpleRouter(use_regex_path=False) registrando 'clients' con basename='client' para generar URLs estándar semánticas.
- **Pruebas Automatizadas de Endpoint:** Implementar ClientApiTests con APITestCase comprobando respuestas 201, 200, 204 (baja lógica) y 400 (validaciones).

Punto de partida y Requisitos

Estado Previo (ISS-07 Superado)

- Modelo Client definido con campos name, email, phone, status e id.
- ClientSerializer implementado con validación de e-mail y conversión de cadenas vacías a None.
- Configuración global de DRF con `djangorestframework-camel-case` y `CorsMiddleware`.

Requisitos de la Fase I (OPEN)

- No requiere tokens JWT ni cabecera Authorization (acceso libre).
- `permission_classes = [AllowAny]`.
- No se importan ni usan clases de `apps.security`.
- En el ISS-22 este mismo endpoint será protegido con JWT + RBAC sin modificar la lógica interna del `ModelViewSet`.

Construcción: ClientViewSet en apps/client/views.py

```
from rest_framework import viewsets
from rest_framework.permissions import AllowAny

from apps.client.models import Client
from apps.client.serializers import ClientSerializer
from apps.common.status import RecordStatus

class ClientViewSet(viewsets.ModelViewSet):
    queryset = Client.objects.all()
    serializer_class = ClientSerializer
    permission_classes = [AllowAny]
    lookup_value_converter = "int"

    def get_queryset(self):
        queryset = super().get_queryset()
        status_value = self.request.query_params.get("status")
        if status_value:
            queryset = queryset.filter(status=status_value)
        return queryset

    def perform_destroy(self, instance):
        instance.status = RecordStatus.INACTIVE
        instance.save(update_fields=["status", "updated at"])
```

Enrutamiento: SimpleRouter y URLs globales

apps/client/urls.py

```
from rest_framework.routers import SimpleRouter
from apps.client.views import ClientViewSet

router = SimpleRouter(use_regex_path=False)
router.register("clients", ClientViewSet,
               basename="client")

urlpatterns = router.urls
```

config/urls.py & Rutas Generadas

Inclusión en config/urls.py:
`path("api/", include("apps.client.urls"))`

Rutas expuestas en el API:

- GET /api/clients/ (list)
- POST /api/clients/ (create)
- GET /api/clients/<int:pk>/ (retrieve)
- PUT /api/clients/<int:pk>/ (update)
- PATCH /api/clients/<int:pk>/ (partial_update)
- DELETE /api/clients/<int:pk>/ (destroy -> baja lógica)

Verificación: Pruebas en apps/client/tests.py

Casos de Prueba en ClientApiTests (APITestCase OPEN):

- **test_create_defaults_and_persists:** POST /api/clients/ -> Status 201 CREATED. Comprueba status=inactive por defecto, normalización de email a minúsculas, presencia de 'createdAt' en camelCase JSON y persistencia en ORM.
- **test_invalid_name_and_duplicate_email:** POST con nombre de 1 carácter -> Status 400. POST con email duplicado -> Status 400 BAD REQUEST.
- **test_soft_delete_keeps_the_row:** DELETE /api/clients/<id>/ -> Status 204 NO CONTENT. Verifica que Client.objects.count() sigue siendo 1 y status cambia a 'inactive'.
- **test_list_retrieve_and_update:** GET /api/clients/ (200 OK), GET /api/clients/<id>/ (200 OK), PATCH /api/clients/<id>/ (200 OK) cambiando teléfono y estado.

Conceptos Django REST Framework e Integridad

ModelViewSet

Clase que provee la implementación estándar de las 6 acciones CRUD sobre un Model y Serializer.

perform_destroy

Punto de extensión idiomático de DRF para personalizar el borrado. Evita sobrescribir delete() en la vista.

SimpleRouter vs DefaultRouter

Se prefiere SimpleRouter para evitar la creación no deseada de la raíz global de la API que causaría colisiones entre apps.

use_regex_path=False

Instruye al router para generar convertidores nativos de path() (<int:pk>) en lugar de expresiones regulares avanzadas.

Criterios de Aceptación, Evidencias y GATE

Matriz GATE — ISS-08

[AC-08-01] POST válido a /api/clients/ responde HTTP 201 CREATED. (Verif: ClientApiTests | Evidencia: EVI-08-01 -> **PASS**)

[AC-08-02] El cliente creado persiste en la base de datos con los valores correctos. (Verif: Client.objects.get | Evidencia: EVI-08-01 -> **PASS**)

[AC-08-03] DELETE /api/clients/<id>/ responde HTTP 204 y conserva la fila con status=inactive. (Verif: perform_destroy | Evidencia: EVI-08-01 -> **PASS**)

[AC-08-04] Un cuerpo JSON inválido (ej. nombre corto de 1 carácter) responde HTTP 400 BAD REQUEST. (Verif: ClientSerializer | Evidencia: EVI-08-01 -> **PASS**)

Preguntas de Defensa Oral — ISS-08

P: ¿Por qué se utiliza `ModelViewSet` en lugar de `APIView` o `GenericAPIView` para el recurso `Client`?

R: Porque `Client` es un recurso CRUD totalmente homogéneo sin lógica de negocio personalizada compleja en la vista. `ModelViewSet` estandariza las 6 acciones REST sin duplicar código.

P: ¿Por qué se sobrescribe `perform_destroy()` en lugar de `delete()` en el `ViewSet`?

R: `perform_destroy` es el hook idiomático provisto por `mixins.DestroyModelMixin` en DRF para intervenir la operación de borrado antes o durante la persistencia, permitiendo implementar la baja lógica de forma limpia.

P: ¿Por qué se utiliza `SimpleRouter` y no `DefaultRouter`?

R: `DefaultRouter` genera una vista raíz de API (API Root). Si múltiples subaplicaciones registran routers en la misma ruta base (`/api/`), `DefaultRouter` genera colisiones en la vista raíz. `SimpleRouter` solo genera los endpoints explícitos del recurso.

P: ¿Qué efecto tiene `query_params.get('status')` en `get_queryset()`?

R: Permite al cliente filtrar los registros devueltos por su estado (ej. `/api/clients/?status=active`) sin romper la paginación ni requerir un endpoint adicional.