# Objetivo general

Crear una aplicación que permita la carga masiva de datos para el área de gestión y operaciones.

**Epic general**
Como encargado de operaciones quiero poder hacer una carga masiva de datos a través de un archivo xlsx que permita actualizar o crear estudiantes y padres. También la carga debe permitirnos inscribir a dichos estudiantes dentro de los cursos seleccionados.

**User stories generales**

1. “Como encargado de operaciones quiero poder crear padres y estudiantes masivamente e inscribir a dichos estudiantes en grupos para poder brindarles acceso”.
2. “Como encargado de operaciones quiero poder actualizar masivamente los datos de estudiantes que figuran dentro del archivo de carga masiva como un candidato a inscripción pero que conocemos previamente que ya existe en nuestro sistema”.
3. “Como encargado de operaciones quiero poder visualizar si al cargar los datos se produjeron errores asociados a la estructura de datos”.
4. “Como encargado de operaciones quiero poder visualizar cuales son los estudiantes cargados en el grupo seleccionado”.
5. “Como encargado de operaciones quiero poder visualizar cuales son los estudiantes cargados en el salón seleccionado”.

# ¿Qué buscamos implementar?

**Descripción objetivo**
Nuestro objetivo es poder crear una aplicación que permita procesar los datos de padres/madres/tutores, estudiantes e inscripciones para atender todos los casos de ventas B2B dentro de la empresa o poder actualizar masivamente datos de cualquiera de estos tres objetos asociados. Los cursos, grupos, salones y profesores que se van a utilizar pueden estar creados previamente a la carga masiva pero dicha creación debe hacerse siempre a través del API. Es super importante tener en cuenta que el usuario es una persona de operaciones, por lo que el manejo de errores e intuición deben ser parte fundamental para el entendimiento de los comportamientos posibles y/o esperados.

En definitiva, lo que vamos a buscar es:

- Crear padres/madres/tutores.
- Crear estudiantes.
- Crear inscripciones.
- Listar estudiantes de un grupo.
- Listar estudiantes de un salon.
- Listar grupos con sus respectivos cursos.
- Listar salones con sus respectivos grupos.
- Crear salones
- Crear grupos
- Crear profesores
- Crear cursos

Si deseamos que el estudiante y/o padre/madre/tutor deben actualizarse, debemos guiarnos por el ID introducido en una columna de nuestro archivo.

**Algo importante a considerar es que en el caso de que una creación falle, el error debe manejarse como un error en la creación de la fila completa del archivo (es decir, ningún elemento (estudiante, padre/madre/tutor, inscripción) debería ser creado.** Ejemplo: Imaginemos que una fila maneja todos los datos del estudiante, padre/madre/tutor, curso, grupo y salón. Si estamos creando el estudiante y este no puede crearse, no se debe proceder a crear el padre/madre/tutor ni tampoco el resto de la información. Esto con el fin de evitar dejar datos "huérfanos" o corromper el esquema de datos.

**Requisitos técnicos**

- Se deberá desarrollar la app utilizando **Django** o **Fast API** como frameworks.
- La base de datos debe ser relacional (***Elige la que más te guste***).
- La aplicación debe poder correr en cualquier sistema operativo y que sea de fácil portabilidad (***Docker***).
- Se deben poder validar los criterios de aceptación a través de **Unit Tests**.
- Necesitamos contar con una **Documentación** para saber como levantar el proyecto y hacerlo que funcione.
- En cuanto a librerías, puedes usar las que consideres necesarias.

**Puntos opcionales**

A continuación presentamos una lista de aspectos adicionales que sumarán muchos puntos si logras hacerlos:

- Carga masiva de cursos, grupos y salones a través de un archivo xlsx.
- Actualizar datos del curso, grupo y salón siempre y cuando se introduzca el id de los estos en el archivo.
- Eliminar cursos
- Eliminar salones
- Eliminar grupos
- Eliminar estudiantes
- Listar los estudiantes que no están inscritos en el grupo.
- Listar los profesores que tienen asignado al menos un salón.
- Listar los profesores que no tienen asignado ningún salón.
- Implementación de estrategias de optimización para hacer que tus servicios respondan lo más rápido posible.
- Clean architecture.
- Principios SOLID.

**Detalles generales**

- Un padre/madre/tutor puede tener múltiples estudiantes pero un estudiante solo pertenece a un padre/madre/tutor.
- Los estudiantes son quienes quedan inscritos en los respectivos grupos y salones.
- Un grupo puede tener múltiples salones pero un salón solo pertenece a un grupo.
- Todos los salones tienen el mismo horario del grupo.
- El email del estudiante es único.
- El email del padre/madre/tutor es único.
- Un estudiante no puede estar más de una vez en un grupo o salón.
- El tipo de documento, número de documento y país es una combinación única.
- Los atributos requeridos de cada entidad están marcados con un asterisco.

| Students | Guardians | Courses | Groups |
|---|---|---|---|
| id | id | id | id |
| first_name * | first_name * | name * | name * |
| last_name * | last_name * | description | description |
| email * | email * | | start_time * |
| guardian_id * | document_type | | end_time * |
| | document_number | | course_id * |
| | country | | |

| Rooms | Teachers | Enrollments |
|---|---|---|
| id | id | id |
| name * | first_name * | group_id * |
| group_id * | last_name * | student_id * |
| teacher_id * | email * | course_id * |
| | document_type | room_id * |
| | document_number | |
| | country | |