

# **FUNCIONALIDADES DE LA PLANTILLA**

- ☐ El backend debe ser una API REST tal como se ha visto en clase implementando al menos los métodos GET, POST, PUT y DELETE y devolviendo un conjunto de códigos de estado adecuado.
- ☒ La API debe tener un mecanismo de autenticación.
- ☒ La API que gestione el recurso también debe ser accesible en una dirección bien versionada.
- ☒ Se debe tener una documentación de todas las operaciones de la API incluyendo las posibles peticiones y las respuestas recibidas.
- ☒ Debe tener persistencia utilizando MongoDB u otra base de datos no SQL.
- ☐ Deben validarse los datos antes de almacenarlos en la base de datos (por ejemplo, haciendo uso de mongoose).
- ☒ Debe haber definida una imagen Docker del proyecto
- ☒ Gestión del código fuente: El código debe estar subido a un repositorio de Github siguiendo Github Flow
- ☒ Integración continua: El código debe compilarse, probarse y generar la imagen de Docker automáticamente usando GitHub Actions u otro sistema de integración continua en cada commit
- ☒ Debe haber pruebas de componente implementadas en Javascript para el código del backend utilizando Jest o similar. Como norma general debe haber tests para todas las funciones del API no triviales de la aplicación. Probando tanto escenarios positivos como negativos. Las pruebas deben ser tanto in-process como out-of-process.
- ☐ Usar el patrón materialized view para mantener internamente el estado de otros microservicios.
- ☐ Implementar cachés o algún mecanismo para optimizar el acceso a datos de otros recursos.
- ☐ Consumir alguna API externa (distinta de las de los grupos de práctica) a través del backend o algún otro tipo de almacenamiento de datos en cloud como Amazon S3.
- ☐ Implementar el patrón “rate limit” al hacer uso de servicios externos.
- ☒ Implementar un mecanismo de autenticación basado en JWT o equivalente.
- ☐ Implementar el patrón “circuit breaker” en las comunicaciones con otros servicios.
- ☒ Implementar un microservicio adicional haciendo uso de una arquitectura serverless (Functions-as-a Service). *(Para hacerlo debes eliminar el archivo [db.js](#) y eliminar las partes del código donde se conecta a la base de datos. Recuerda borrar las variables de entorno relacionadas con la BD y también repetir esas acciones en los tests, ya que actualmente se puede usar una BD de prueba o de desarrollo para testear)*
- ☐ Implementar mecanismos de gestión de la capacidad como throttling o feature toggles para rendimiento.
- ☐ Cualquier otra extensión al microservicio básico acordada previamente con el profesor.

## EXTRAS:

- ☒ Plantillas de issues
- ☒ Logger
- ☒ Cobertura de pruebas
- ☒ [CONTRIBUTING.MD](#), [SECURITY.MD](#), CODE\_OF\_CONDUCT.MD, LICENSE
- ☒ estructura general
- ☒ linter automático y hooks