Máster en Ingeniería del Software: Datos, Cloud y Gestión TI

Curso 2019/2020

Memoria justificativa del proyecto

Asignatura: Fundamentos de Ingeniería del Software para Sistemas Cloud

Equipo 3:

Agustín Núñez Arenas

Carlos Capitán Agudo

Tabla de contenido

1. Nivel de acabado solicitado

El nivel de acabado al que nos presentamos con este documento es el nivel hasta 7 puntos. Para ello, cumplimos todos los requisitos del nivel hasta 5 puntos, poseemos una aplicación basada en microservicios básica implementada y poseemos dos características de la implementación de un microservicio avanzado:

- API REST documentada con swagger.
- Implementación de un mecanismo de autenticación más completo.

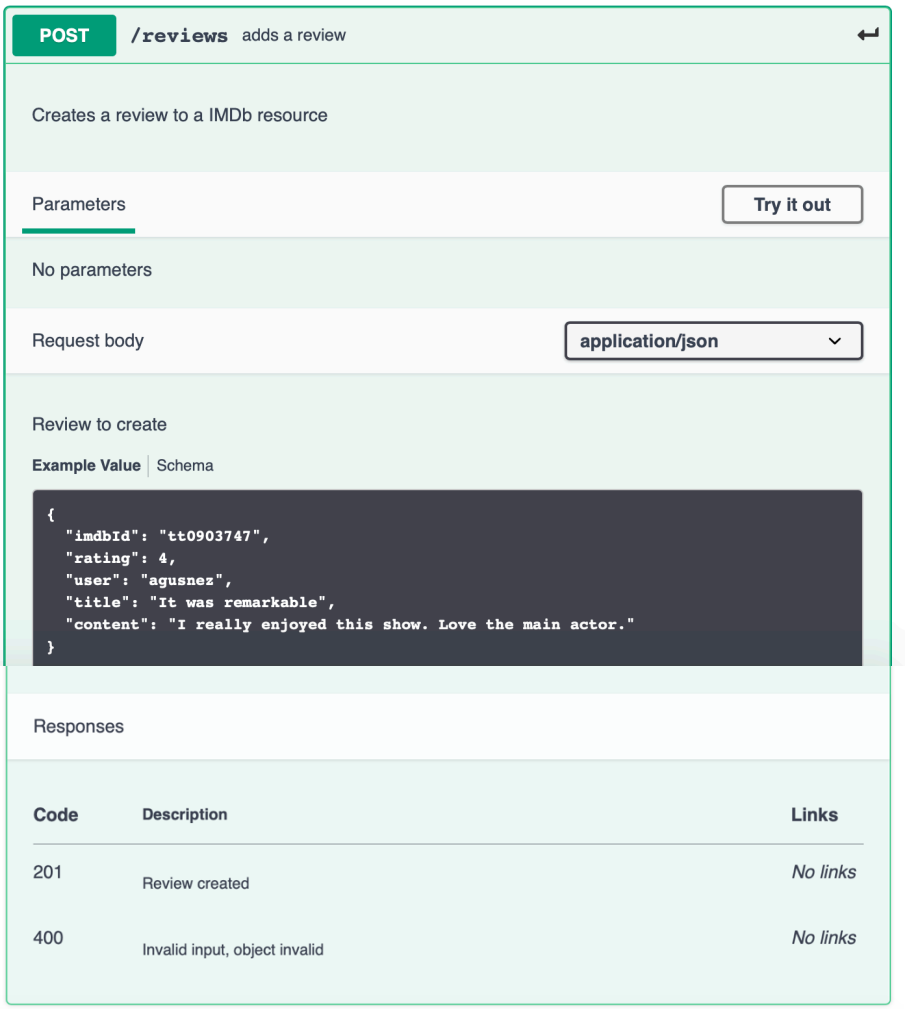
2. Justificación de los requisitos

2.1 Microservicio básico

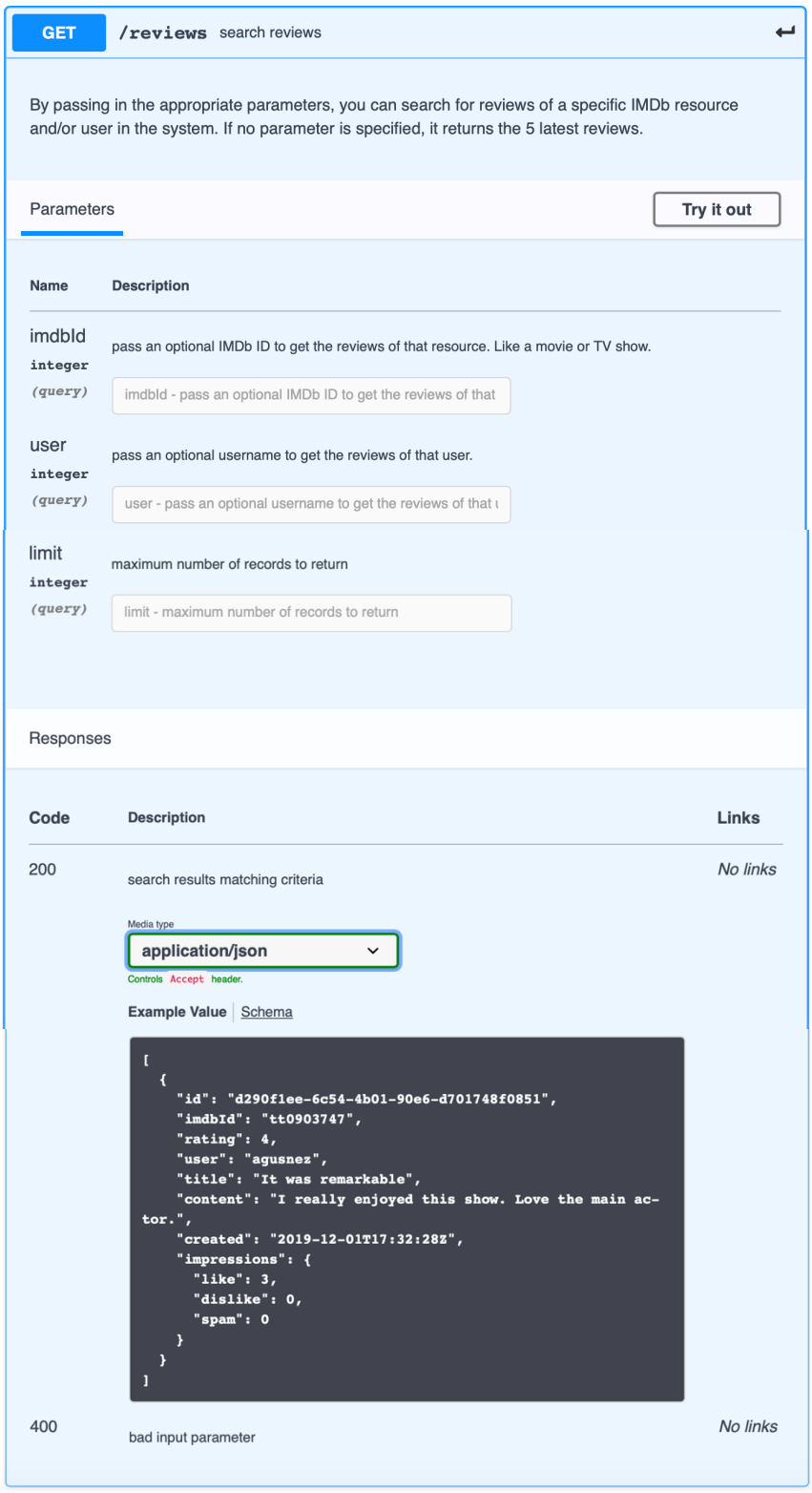
2.1.1 Implementación de una API REST con los métodos GET, POST, PUT y DELETE.

Nuestro microservicio se encarga de todo lo relacionado con las reviews de películas de una página web, lo cual incluye las siguientes operaciones descritas en swagger (todas estas operaciones están implementadas en el repositorio de reviews-api):

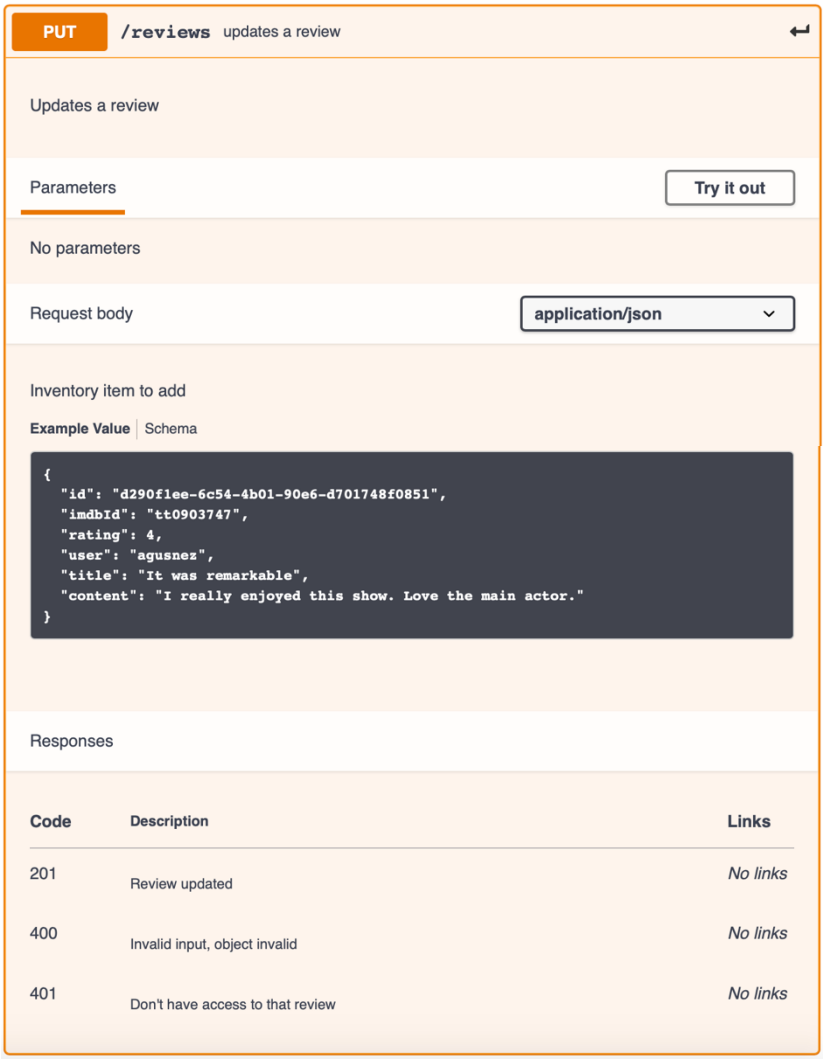
- Crear una review de una película, que sería la operación POST.



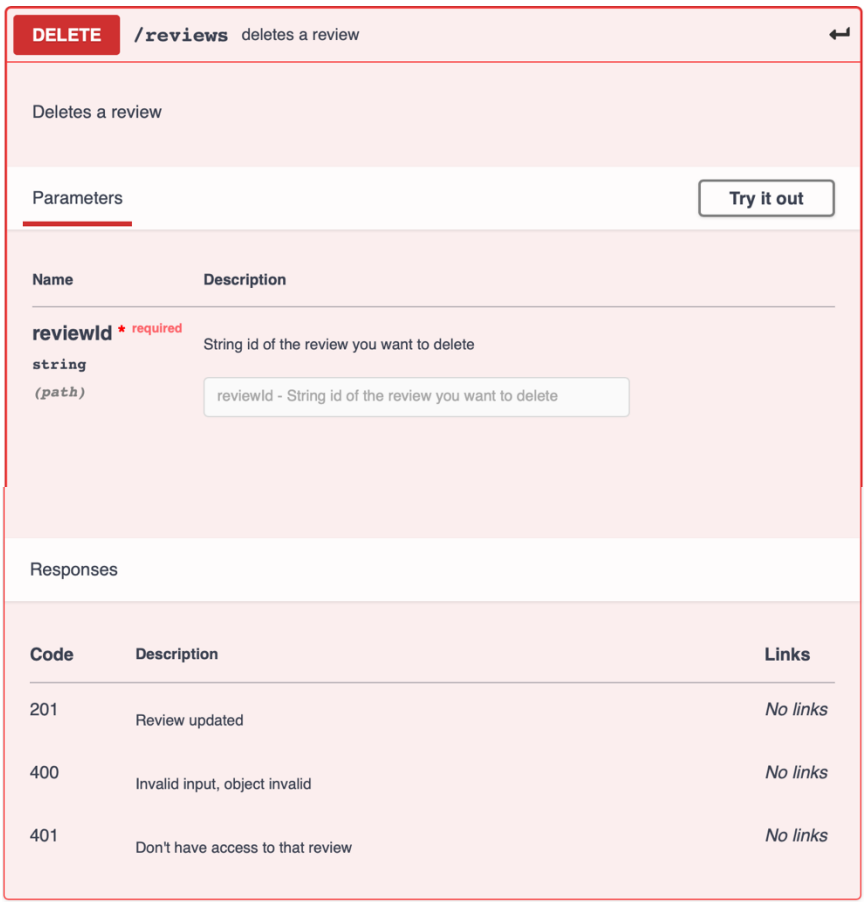
- Obtener las reviews asociadas a una película, que sería la operación GET.



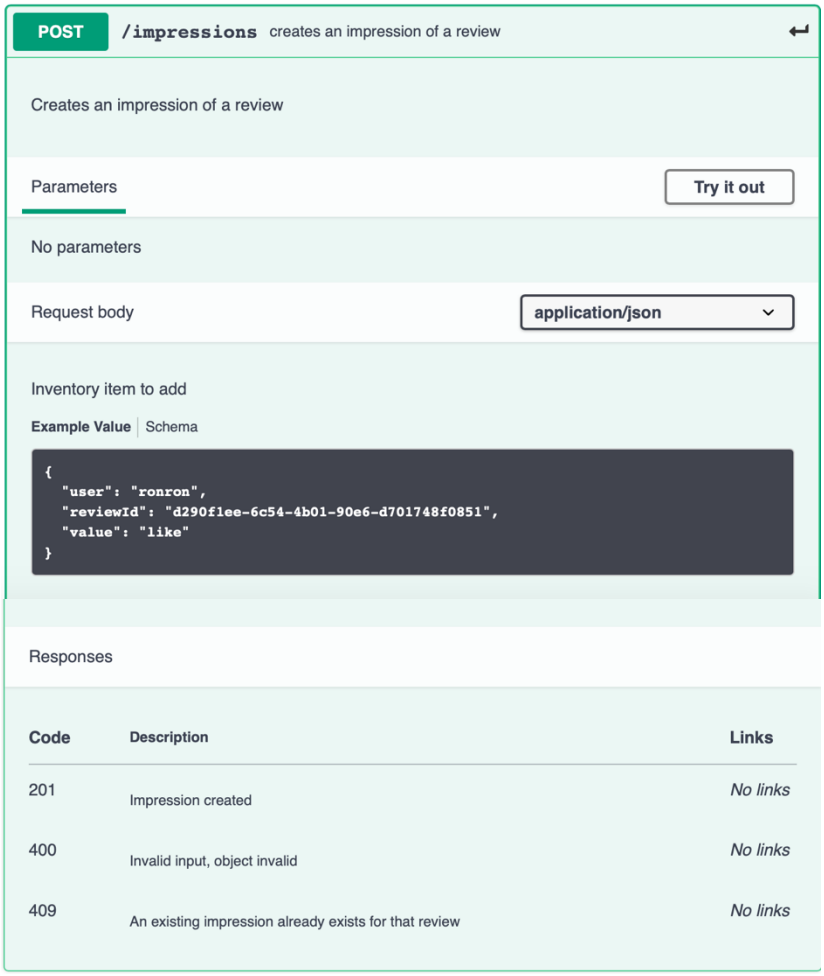
- Modificar la review asociada a una película, que sería la operación PUT.



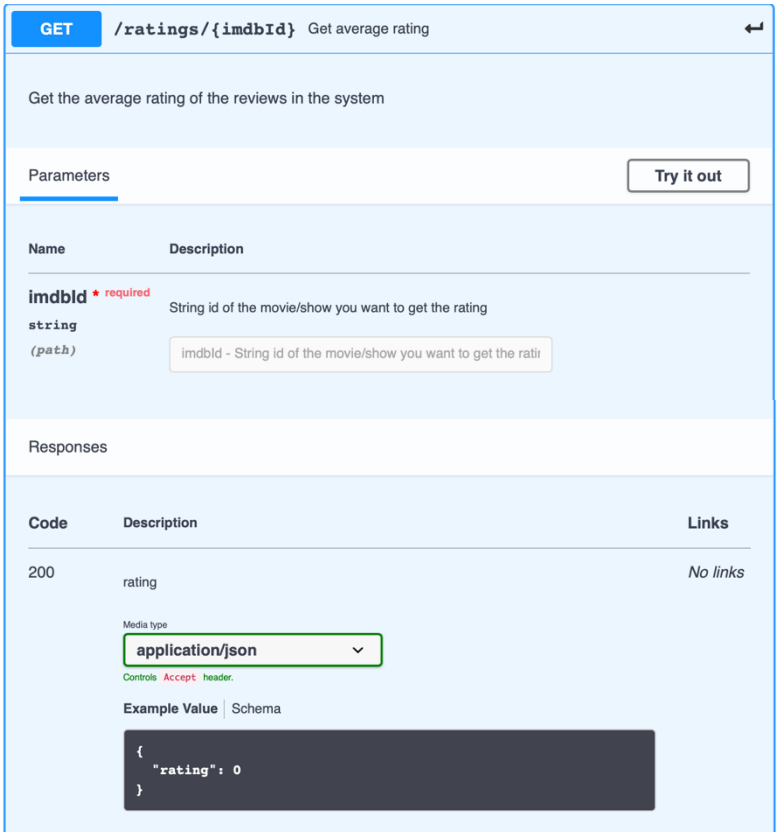
- Borrar la review asociada a una película, que sería la operación DELETE.



- Crear los avisos de ‘me gusta’, ‘no me gusta’ y ‘spam’ asociados a una review, que sería otra operación POST.



- Obtener la valoración media de una película a partir de las valoraciones de las reviews, que sería otro método GET.



2.1.2 Implementación de un frontend de todas las operaciones de la API

Se puede encontrar el frontend común en **https://fis-frontend.herokuapp.com**

2.1.3 Despliegue en la nube y accesibilidad por una URL

El microservicio está desplegado en **Heroku** y se puede acceder de forma pública en esta URL:

https://reviews-api.herokuapp.com

2.1.4 Accesibilidad a la API gestora de recursos

Para acceder a nuestro servicio hemos creado un enrutado que permite que nuestra API sea fácilmente accesible.

2.1.5 Conjunto de ejemplos de uso del API en Postman de todas las operaciones

En el repositorio de reviews-api está un archivo nombrado **postman-collection.json** que contiene ejemplos para todas las operaciones del API.

2.1.6 Persistencia mediante MongoDB

Hemos optado por la alternativa de consumir la base de datos como servicio que ofrece **MongoDB Atlas.** El servicio venía configurado por defecto en modo clúster con tres servidores, en este mismo clúster hemos configurado tres bases de datos: de desarrollo, de integración y de producción.

Las ventajas que nos ha traído esta decisión son las siguientes:

- **Seguir buenas prácticas de desarrollo.** Por tener la base de datos de desarrollo y de producción con la configuración casi idéntica.
- **Simplificar la infraestructura.** No alojamos nosotros la base de datos ni tocamos la configuración avanzada.
- **Desarrollo ágil y cómodo.** Los integrantes del grupo compartimos la misma base de datos, es decir, en desarrollo vemos los mismos datos y la base de datos está siempre disponible sin necesidad de desplegarla.

2.1.7 Gestión del código fuente y mecanismos de integración continua

La gestión del código fuente del microservicio desarrollado se ha hecho con Git y el repositorio está subido en el siguiente repositorio de **GitHub**: https://github.com/Agusnez/reviews-api. Se han seguido las recomendaciones recogidas en *GitHub flow*. Toda nueva implementación se desarrolla en una rama y una vez apta para producción se producen revisiones de código en cada *pull request*.

Con respecto a la integración continua se ha usado **Travis CI** que ofrece una solución sencilla con repositorios de GitHub.

2.1.8 Definición de una imagen Docker del proyecto

En el repositorio de reviews-api hay una imagen de Docker definida en donde se especifica que se copien los archivos “package.json” y “package-lock.json”, los archivos que representan el servidor (server.js) y la base de datos (db.js), modelos utilizados en MongoDB, las funciones enrutadas y las funciones auxiliares.

2.1.9 Realización de pruebas unitarias en Javascript para el código del backend utilizando Jest.

En el repositorio de github (https://github.com/Agusnez/reviews-api) existe un archivo llamado server.test.js dentro de la carpeta “tests”, en el cual se encuentran implementadas pruebas para todos los casos del backend tanto los ideales (aquellos en los que se devuelve un código de respuesta 200 o 201) como los no ideales. En ambos casos se ha utilizado Jest para realizar las pruebas.

2.1.10 Realización de pruebas de integración con la base de datos.

En el repositorio de github (https://github.com/Agusnez/reviews-api) existe un archivo llamado integration.test.js dentro de la carpeta “tests”, en el cual se comprueba que están escribiendo los datos en la base de datos.

2.1.11 Mecanismo de autenticación en la API.

El mecanismo de autenticación que se ha usado es el resultado de una integración con nuestros compañeros de grupo encargados de hacer el microservicio de autenticación. Lo consideramos un desarrollo avanzado y lo explicamos en detalle en la sección 2.2.2.

2.2 Microservicio avanzado

2.2.1 API REST documentada con swagger

Nuestra API REST se encuentra documentada en el siguiente enlace y pudo apreciarse en el apartado 2.1.1:

https://app.swaggerhub.com/apis-docs/Agusnez/reviews/1.0.0

2.2.2 Implementación de un mecanismo de autenticación más completo.

Como adelantamos en la sección 2.1.11, el mecanismo de autenticación que se ha desarrollado es fruto de una integración con nuestros compañeros encargados de desarrollar el microservicio de autenticación. A modo resumen, esta autenticación consiste en un *token* que el cliente consumidor de la API puede obtener iniciando sesión en el microservicio de autenticación y usarlo en nuestra API configurando la variable `Authorization` en la cabecera de cada una de las peticiones que requieran autenticación. Concretamente las llamadas que necesitan de autenticación son:

- **POST /reviews**
- **PUT /reviews**
- **POST /impressions**

En los diagrama de secuencia de a continuación, se puede observar un ejemplo de autenticación positivo y negativo.

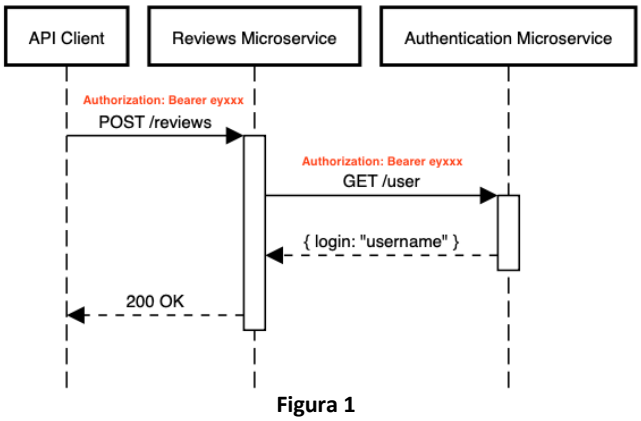


Figura 1

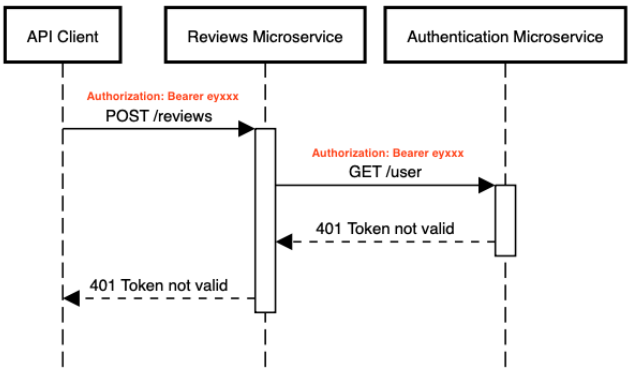


Figura 2

3. Análisis de esfuerzos

En la siguiente tabla vienen recogidos nuestros esfuerzos en desarrollar el proyecto:

Tarea	Responsable	Número de horas
Preparación del proyecto en heroku y en GitHub con CI/CD (travis, dockerfile…)	Agustín Núñez Arenas	5h
Documentación de la Api en Swagger	Agustín Núñez Arenas	4h
Implementación del método GET de reviews, realización de sus pruebas unitarias y de ejemplos en Postman	Agustín Núñez Arenas	10h
Implementación del método POST de reviews, realización de sus pruebas unitarias y de ejemplos en Postman	Agustín Núñez Arenas	10h
Implementación del método POST de impresiones, realización de sus pruebas unitarias y de ejemplos en Postman	Agustín Núñez Arenas	10h
Investigación sobre la implementación del método DELETE en MongoDB	Carlos Capitán Agudo	5h
Implementación del método DELETE de reviews, realización de sus pruebas unitarias y de ejemplos en Postman	Carlos Capitán Agudo	10h
Investigación de como implementar el método put con mongoDB	Carlos Capitán Agudo	5h
Implementación del método PUT de reviews, realización	Carlos Capitán Agudo	10h

de sus pruebas unitarias y de ejemplos en Postman		
Implementación del método getAverageRating, realización de sus pruebas unitarias y de ejemplos en Postman	Carlos Capitán Agudo, Agustín Núñez Arenas	2h, 3h
Realización de pruebas de integración con la base de datos	Agustín Núñez Arenas	2h
Implementación de la paginación del Front-End	Carlos Capitán Agudo, Agustín Núñez Arenas	10h, 8h
Investigación sobre la implementación de ventanas modales y pop up en react	Carlos Capitán Agudo	6h
Implementación de las ventanas modales de los distintos métodos en el Front-End	Carlos Capitán Agudo	15h
Implementación del estilo de las reviews en el Front-End	Agustín Núñez Arenas	10h
Implementación de la unión del Backend con el Front-End	Agustín Núñez Arenas	3h
Redacción del documento justificativo	Carlos Capitán Agudo, Agustín Núñez Arenas	1h, 2h
Realización de la presentación del proyecto	Carlos Capitán Agudo	2h