# Examen técnico

Debes desarrollar una API, que a través de sus endpoints se obtenga la información de las películas más populares y poder guardarlas y valorarlas en una base de datos propia. Para ello, la información debe obtenerse desde la API pública de TheMovieDB para ser consumida desde esta API.

La documentación de la API la podés obtener en este enlace [Popular (themoviedb.org)](https://developer.themoviedb.org/reference/movie-popular-list)

Debes crearte una cuenta en TheMovieDB para obtener una API Key desde [Mi configuración de API — The Movie Database (TMDB) (themoviedb.org)](https://www.themoviedb.org/settings/api).

Debés crear la ruta de los endpoints siguiendo las buenas prácticas de REST.

## Desafío

Para el desafío, es válido que la base de datos sea Volátil o en una base de datos utilizando ORM.

Recordá que es requisito excluyente los tests de cobertura de este examen.

Debés disponibilizar los siguientes endpoints:

* Un endpoint que me permita ver las películas populares en TheMovieDB
* Endpoints que me permitan agregar y quitar la película a la lista por ID a una lista de películas favoritas, teniendo en cuenta el ID del usuario que las crea.
* Endpoints que me permita valorar y corregir la validación de una película con un índice de 0 a 5 estrellas.
* Un endpoint que me permita consultar mis películas favoritas ordenadas por fecha de estreno de la película y mi valoración. Este endpoint debe devolver todos los detalles de la película, no solamente mi valoración.
* Un endpoint de administración para borrar todos los favoritos de un usuario, basado en su ID.

## Requisitos

1. Lenguaje y Framework:
   * Elegimos Flask para este examen. Esperamos que te apoyes en las librerías sugeridas de Flask para realizar este examen. Al ser una API, este examen debe incluir las protecciones básicas requeridas de una API para ser consumida desde un microservicio frontend.
2. Arquitectura:
   * Sugerimos seguir una arquitectura hexagonal (servicio y adaptadores) para separar las responsabilidades y facilitar el testeo de componentes individuales.
3. Autenticación:
   * Protegé las rutas de la API con un token Bearer. Si escogés una base de datos volátil, hardcodeá los valores ya que no es el foco del examen el sistema de autenticación, sino demostrar la correcta implementación de perfilado (Ver tabla):

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Usuario | Token | Permiso |
| 1 | admin | abcdef1234567890 | ADMIN |
| 2 | consumer | 1234567890 | USER |

1. Caché:
   * Implementá caché usando Redis (Ver punto 6). Si un endpoint GET recibe el mismo llamado en menos de 30 segundos, debe responder con la respuesta cacheada en lugar de realizar la llamada externa nuevamente. La duración del caché debe configurarse mediante una variable de entorno (considerando que el servicio se ejecutará en un pod de Kubernetes).
2. Políticas de Retry y Backoff:
   * Implementá políticas de Retry y Backoff usando la librería `requests`. Si una solicitud a una API externa falla, debe intentar nuevamente según la política definida. En caso de fallo, responde con la información cacheada en Redis, pero debe mostrar el error en consola (stderr).
3. Endpoints:
   * GET: Los endpoints GET que hagan llamados externos deben ser resilientes. Maneja las excepciones de IOError de manera adecuada, devolviendo un JSON con el error siguiendo buenas prácticas RESTful.
   * POST, PUT, PATCH: Los endpoints POST deben tener un token Bearer (hardcodeado) que validen el usuario que esté realizando la operación.
4. Testing:
   * Es fundamental que escribas tests unitarios usando pytest para cubrir todas las rutas de la API, servicios y adaptadores. Asegurate de testear tanto los casos exitosos como los de fallo y de borde, especialmente para la autenticación y el manejo de respuestas y caché.

## Entrega

* El proyecto debe entregarse en un repositorio Git en Github, incluyendo un README con instrucciones claras sobre cómo ejecutar la aplicación y sus tests; y un contrato OpenAPI (dentro del repositorio).
* Te pedimos si es posible que lo publiques en tu cuenta Github/Gitlab/Bitbucket, y la URL del proyecto sea pública. En caso que no puedas, solicitamos que lo comuniques a tu recruiter para coordinar la entrega de este challenge. ¡Exitos!