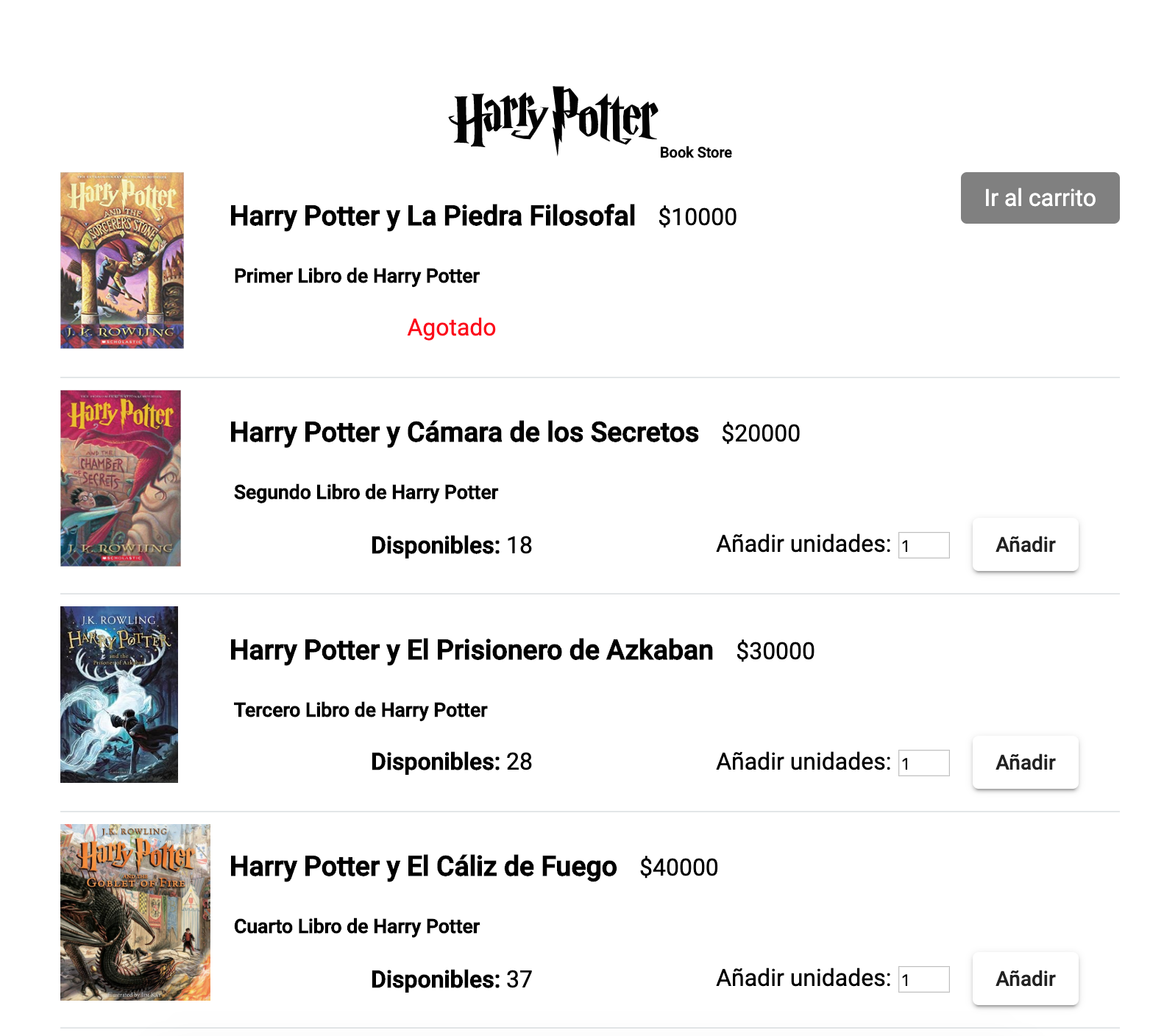
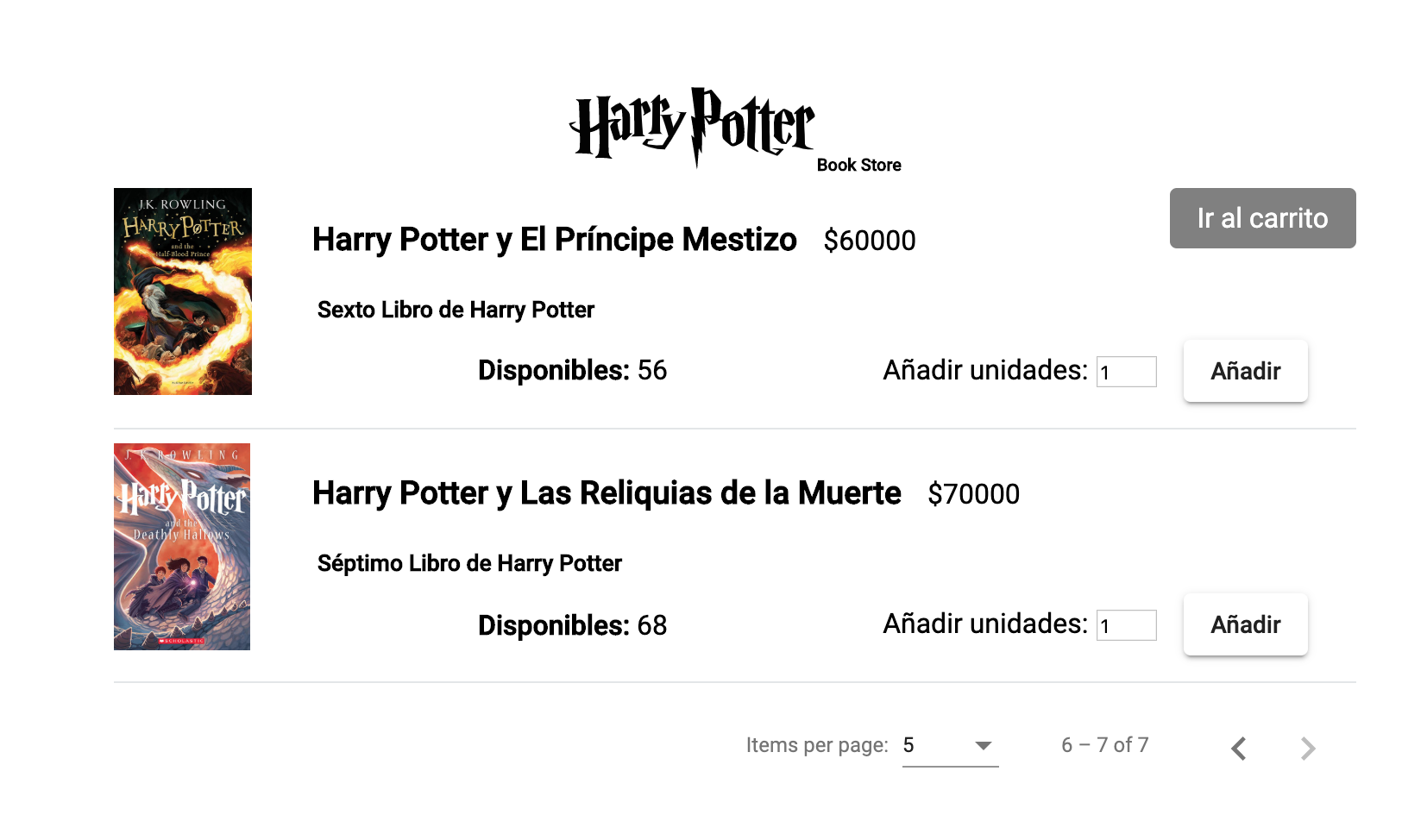
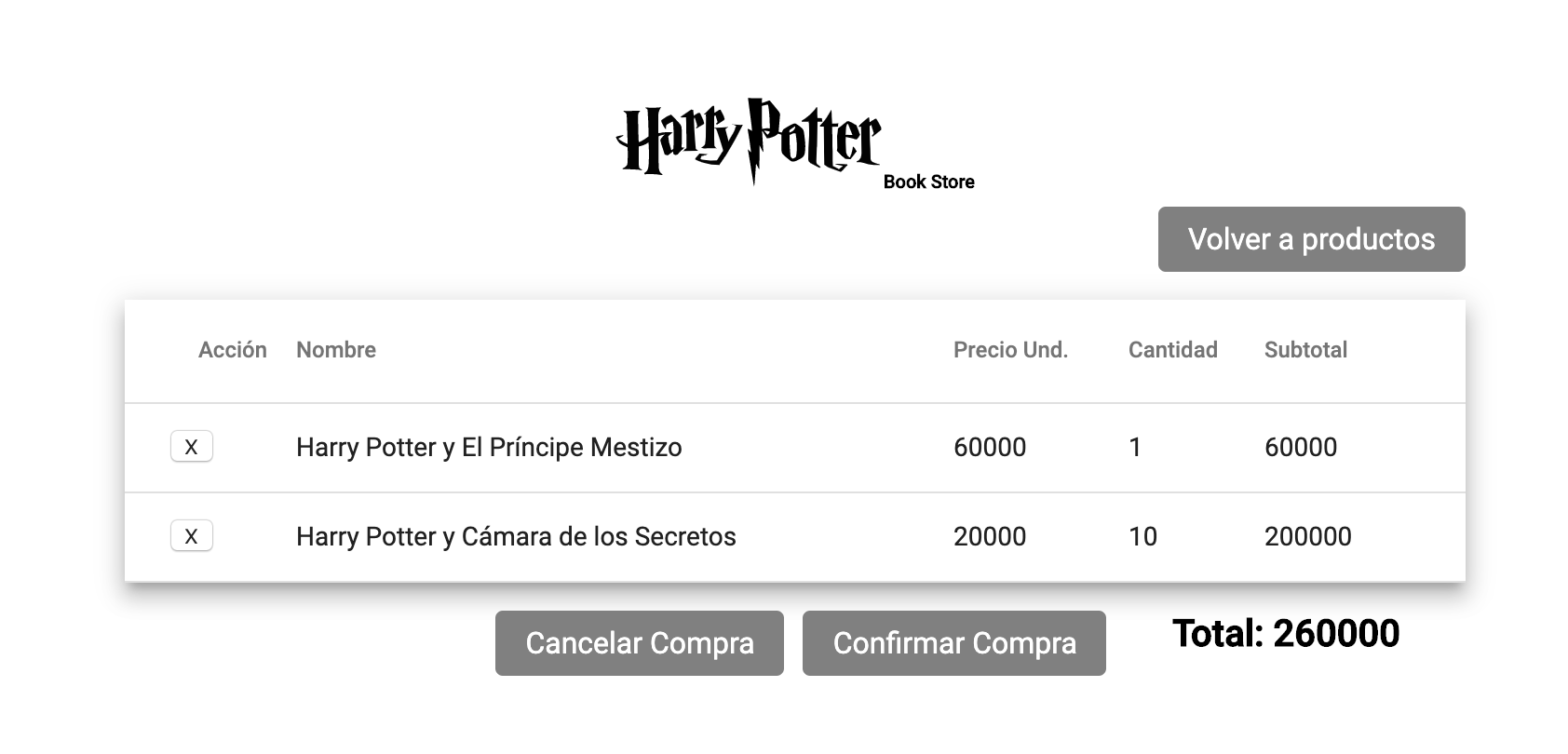
**Screenshots de la App:**

****

****

****

**Respuesta a Retos Complementarios – Intermedios**

1. Patrones arquitectónicos usados:
   * Single Page Application soportado con Angular 8
   * Microservicios soportando una API REST en el Backend
   * Patron Layer básico al interior del MS **book-store-back**

Principios de diseño soportados por el diseño:

* + IoC (Inversion of Control) a través de Spring/Spring Boot

1. Asumiendo un despliegue en AWS (No es por ser Vendor-Locking, pero es con el proveedor de Nube con el que soy mas familiar)



**Web Application Firewall** (WAF): Me interesa bastante las características de Rate Limiting, para evitar posibles ataques de DoS

**Cloudfront**: para tener rendimiento en la entrega de contenido (CDN). Se usa S3 para alojar tanto los fuentes como los assets de la aplicación web.

**API Gateway**: Disponiblizar los recursos de nuestros MS a través de ello. Posiblemente se requiera hacer una configuración de Swagger en nuestros MS para poder configurar los endpoints y payloads de forma dinámica (cuando cambien)

**ECS**: Despliegue y escalado de instancias basadas en imágenes Docker. Fargate para enfocarnos en la construcción de las instancias en clústeres autoadministrados, y auto-escalado automático para aprovisionar instancias a través del Task Definition.

1. Empezaría por mejorar el delivery en desarrollo, usando estrategias de CI/CD. Se podría configurar un Jenkins que tome como base el repositorio de nuestros componentes, realice validación del mismo (coverage, linter, etc.) y arme una Imagen Docker con base en el artefacto desplegable, y lo almacene en el Container Registry. Esto se puede usar tanto en rama de desarrollo como en release.

Mientras vaya creciendo la aplicación y sus flujos de negocio, iría identificando cuales de ellos no tienen una concurrencia alta, y sean cantidatos para mover a Serverless. Con eso los costos en infraestructura y dimensionamiento se reducen.

Me quedaría con NoSQL, quizás no soportado con MongoDB sino también pensar en DynamoDB. Pensando no solo en libros sino en productos, cada producto puede tener características diferentes. Esa variedad de posibles combinaciones de campos que representan cada una de las características, en una SQL debería ser soportada por cada una de las filas de una hipotética tabla. En NoSQL al no existir una estructura definida, cada documento es libre de definir su estructura.

Esperaría a ver que otras características se desean incluir para proponer tecnologías y sean bien aprovechadas.

1. Diagrama de componentes:



Flujo de autenticación de usuario



Flujo General de Consumo API



1. Es posible usar esta solución para cualquier otro tipo de producto, no necesariamente libros. Se alimenta para cada producto las características propias del mismo; reforzando lo que se planteó anteriormente con el uso de NoSQL, en donde se poblaba cada tipo de producto en el documento con la información que se deseara.

Authentication MS se propone como una solución transversal de Autenticación/Autorización. Con el filtro de autenticación propuesto, toda petición requiere que el token asociado a ella sea validado, y permita o no el paso de dicha petición; de lo contrario: 403 Forbidden.

1. Si bien no está propuesto en esta solución, usar AWS Parameter Store para alojar las propiedades en todos los ambientes (DEV, STG, PRO), y Spring Cloud Config Server para hacer hot swapping de estas; es posible mantener las propiedades actualizadas sin indisponibilizar la infraestructura así sea en periodos cortos. De igual forma con ECS + Fargate el swapping de instancias es progresivo, así que no es una preocupación mayor.
2. A mi parecer, como dueño del producto me enfocaría mas en habilitar capacidades para que los usuarios encuentren los productos que quieren.

* Modelo de sugerencias, usando la secuencia de libros de Harry Potter en este caso. No soy un experto en Análitica/Big Data, pero sugeriría crear un Stream de eventos que se alimente de cada compra, y con base en esto crear las tendencias de cada usuario o grupo de usuarios. Se podría crear un MS nuevo que asocie la información recibida. Mas allá de los costos relacionados con el procesamiento a usar, no veo impacto mayor en la infraestructura de la aplicación ya que funcionaría en un contexto asíncrono a los flujos de negocio de la aplicación.
* Buscador de productos en Full Text, con predictivo. Para este caso, vería necesario la implementación de ElasticSearch, para indexar la información con base en el contenido que haría parte de la búsqueda. El impacto en el performance de esta carácteristica va de cara a la necesidad de tener que replicar la información de Mongo Hacia ElasticSearch para que sobre ella se realice la búsqueda.