Ecosistema de soporte a proyectos Big Data

Práctica 3 - Arquitectura en entornos Big Data: caso Spainflix

José Manuel Bustos Muñoz

Índice

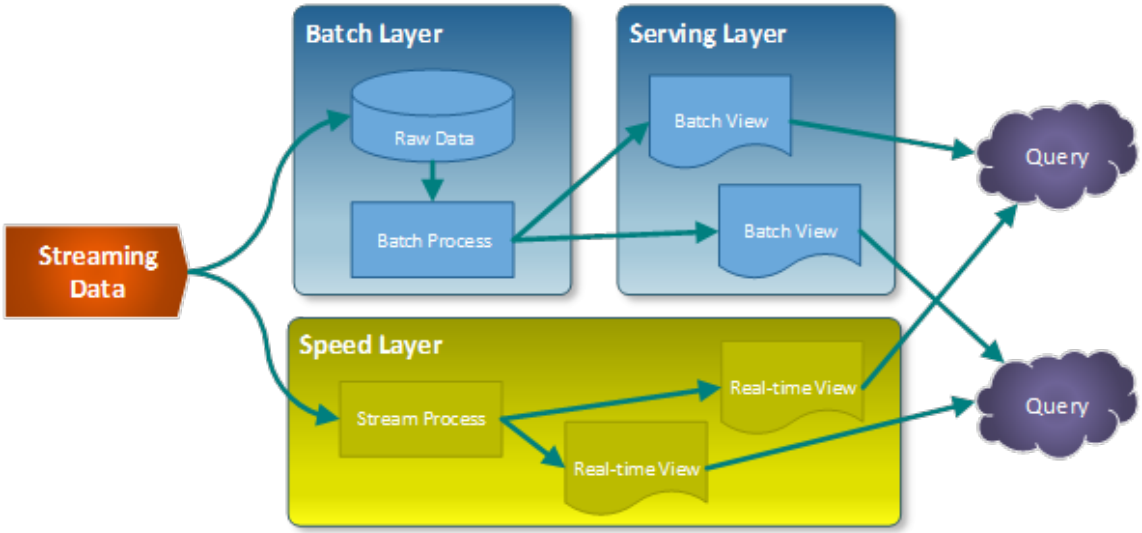
1. Descripción general de la solución.

Se ha elegido el caso de *SpainFlix*, que tiene como objetivo el implantar una solución de recomendación de visualización.

Se recomienda la separación de parte Batch y streaming, y dadas las características de la arquitectura Lambda decidimos utilizar este tipo de arquitectura que separa ambas capas. Se utilizaría la capa Batch para crear y entrenar los modelos mejorando así las predicciones, y la capa streaming para aplicarlos con los datos y hacer valoraciones en tiempo real.

Además otras características que proporciona una arquitectura Lambda es otorgar un sistema robusto tolerante a fallos, que permite realizar escrituras y lecturas con baja latencia, y además que es linealmente escalable. Todas estas características encajan en el objetivo que se busca.

Imagen de una arquitectura Lambda:

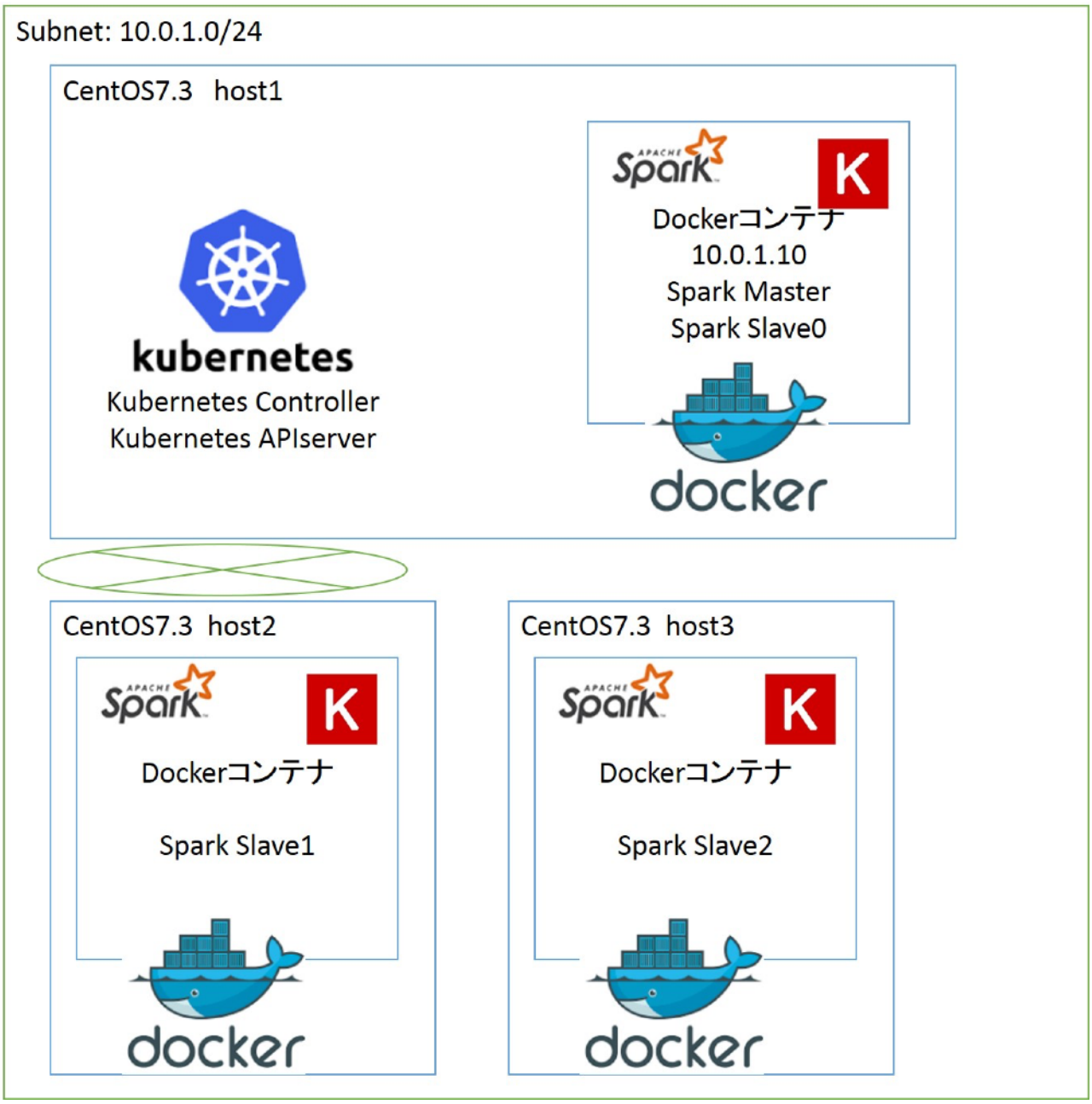


Uno de los requisitos que se indican en el documento sería que se utilice la infraestructura actual, y también se recomienda el uso de clusters basados en *dockers* y gestionados por *Kubernetes*. Asumimos que este sistema es utilizado en otras aplicaciones de la empresa, y por tanto se implantará de este modo la solución: usando contenedores gestionados por *Kubernetes*, en cloud.

Con este mecanismo se crean dinámicamente los contenedores necesarios para ejecutar los jobs, y se tiene un gran ahorro de costes haciendo uso de los clusters *Kubernetes* con autoescalado. Que la solución sea lo más escalable posible es una de las preocupaciones o peticiones iniciales.

Como se quiere que la solución sea Cloud para no aumentar en el inmovilizado, podría usarse una solución en la nube *Azure* de *Microsoft*, que tiene un entorno de *Kubernetes*. Además para la capa de procesamiento y análisis podría utilizarse *Spark*, que tiene integración nativa con clústeres *Kubernetes* y nos proporcionaría toda la potencia necesaria y todas las herramientas para realizar los modelos de recomendación que se buscan con el uso de Spark ML, y para la capa streaming utilizar Spark Streaming.

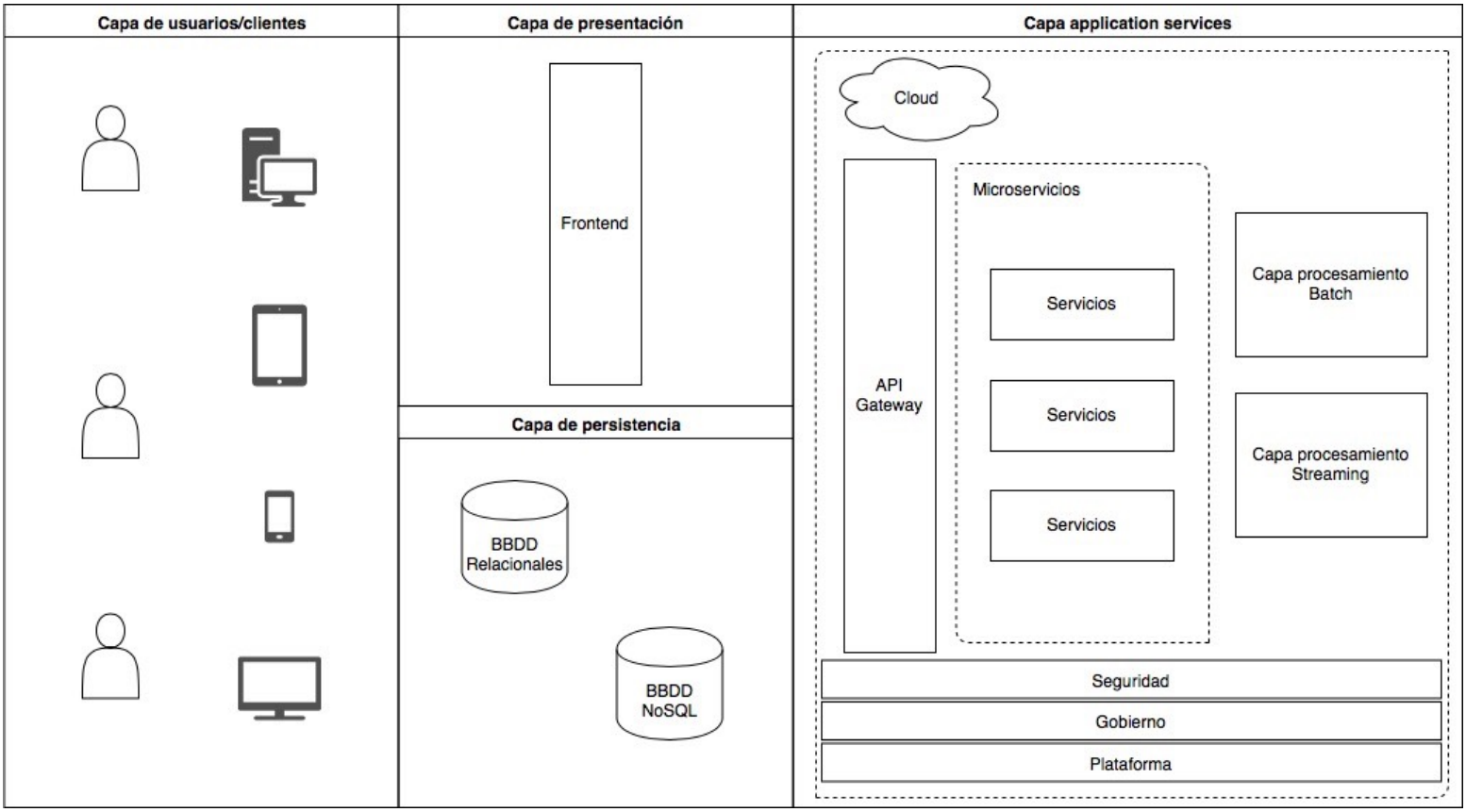
Imagen ejemplo de la integración de Kubernetes, Docker y Spark:



Para la persistencia de la información se siguen utilizando los sistemas que actualmente utilizan en el cliente, por un lado BBDD relacionales y por otro lado BBDD NoSQL.

Por último, se requiere que los usuarios puedan conectarse mediante microservicios (que serán contenedores docker con alguna funcionalidad).

2. Diagrama de visión general de la arquitectura.



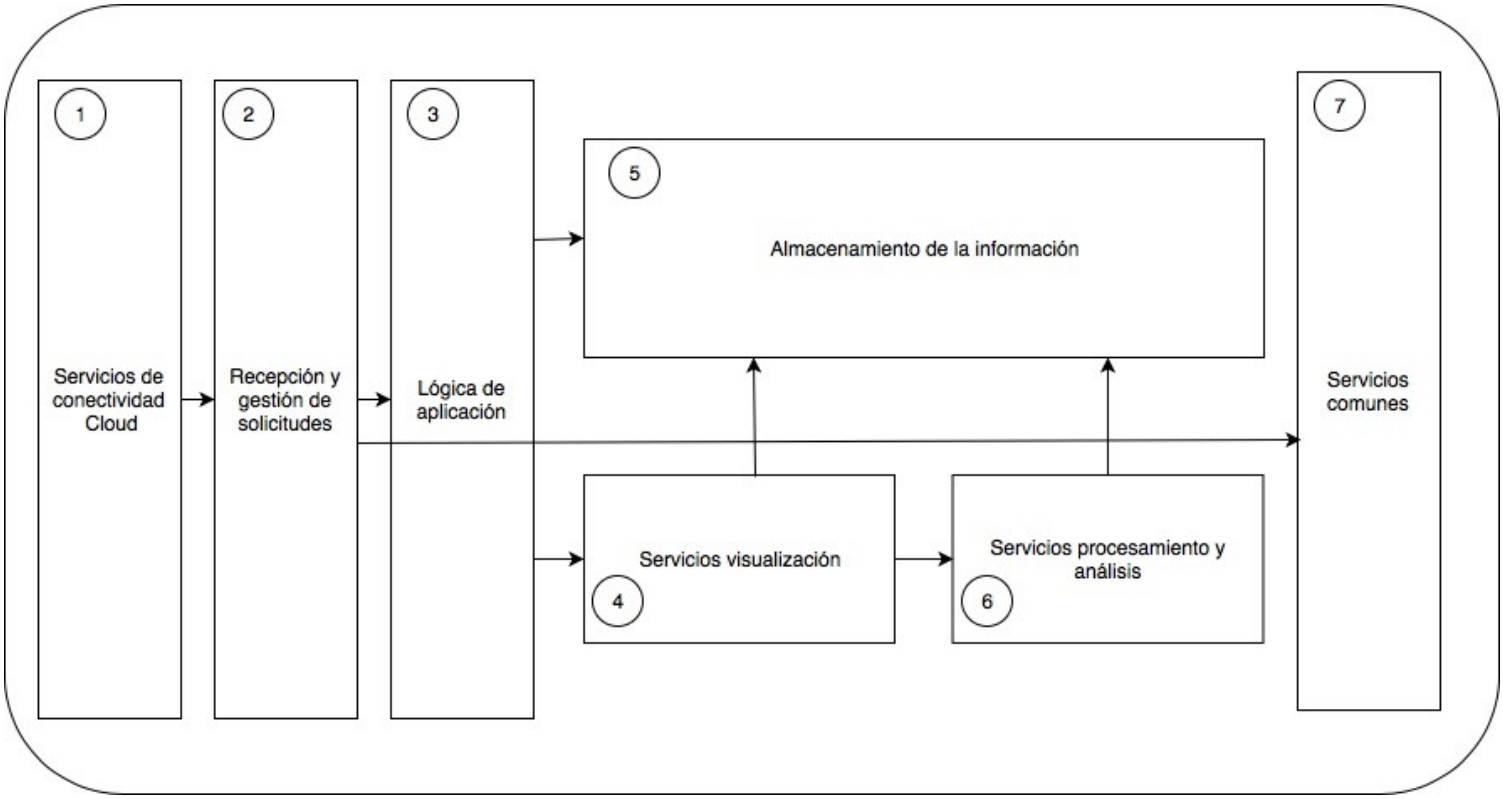
3. Diagrama de contexto de la solución.

4. Recopilación de requerimientos.

Los requerimientos funcionales y no funcionales se recogen en la excel “Requerimientos_Funcionales_NoFuncionales_josemanuelbustos”. A modo resumen serían los siguientes:

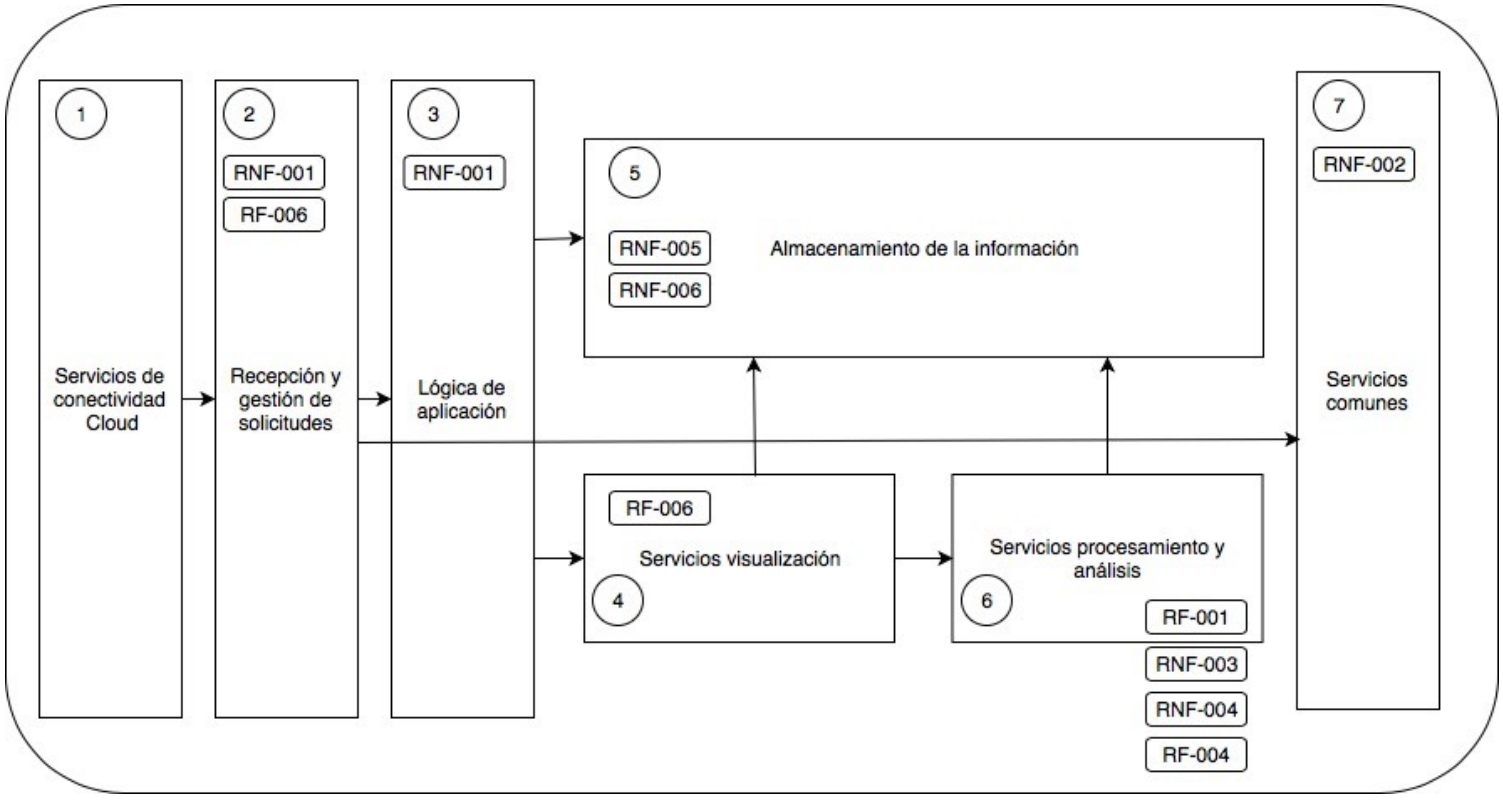
Tipo	Requerimiento
NF	Incluir un motor de recomendación que pueda ser invocado desde un punto como servicio web.
F	Debe ser capaz de responder a estos tipos de recomendaciones: por perfil de usuario, histórico de visualizaciones, recomendación de la empresa y por lo que han visto otros como tú.
NF	Reutilizar la infraestructura actual de la instalación.
F	Las recomendaciones deben ser sensibles a las últimas visualizaciones realizadas.
F	Realizar un piloto de tipo Friend & Family con clasificación por edad de los usuarios.
NF	La solución debe tener desacoplados los procesos batch de los de tiempo real.
NF	En ambas líneas de procesamiento batch y streaming debe ser modular para tener flexibilidad.
NF	Persistencia y trato con distinta información y almacenada en distintos tipos de sistemas.
F	El contenido debe ajustarse según el perfil de consumidor.
NF	Se debe tener cuidado con la información que se guarda por temas de leyes y protección de datos.
NF	La solución debe ser cloud para no tener más inmovilizado.
NF	Alta escalabilidad.
F	Se debe tener en cuenta como línea de negocio el suplemento económico por visualizar algunos contenidos.
F	Necesario un frontend para invocar un modelo.
NF	Se recomienda el uso de clusters basados en dockers y gestionados por Kubernetes u otro gestor de dockers.

5. Modelo de componentes de la solución.



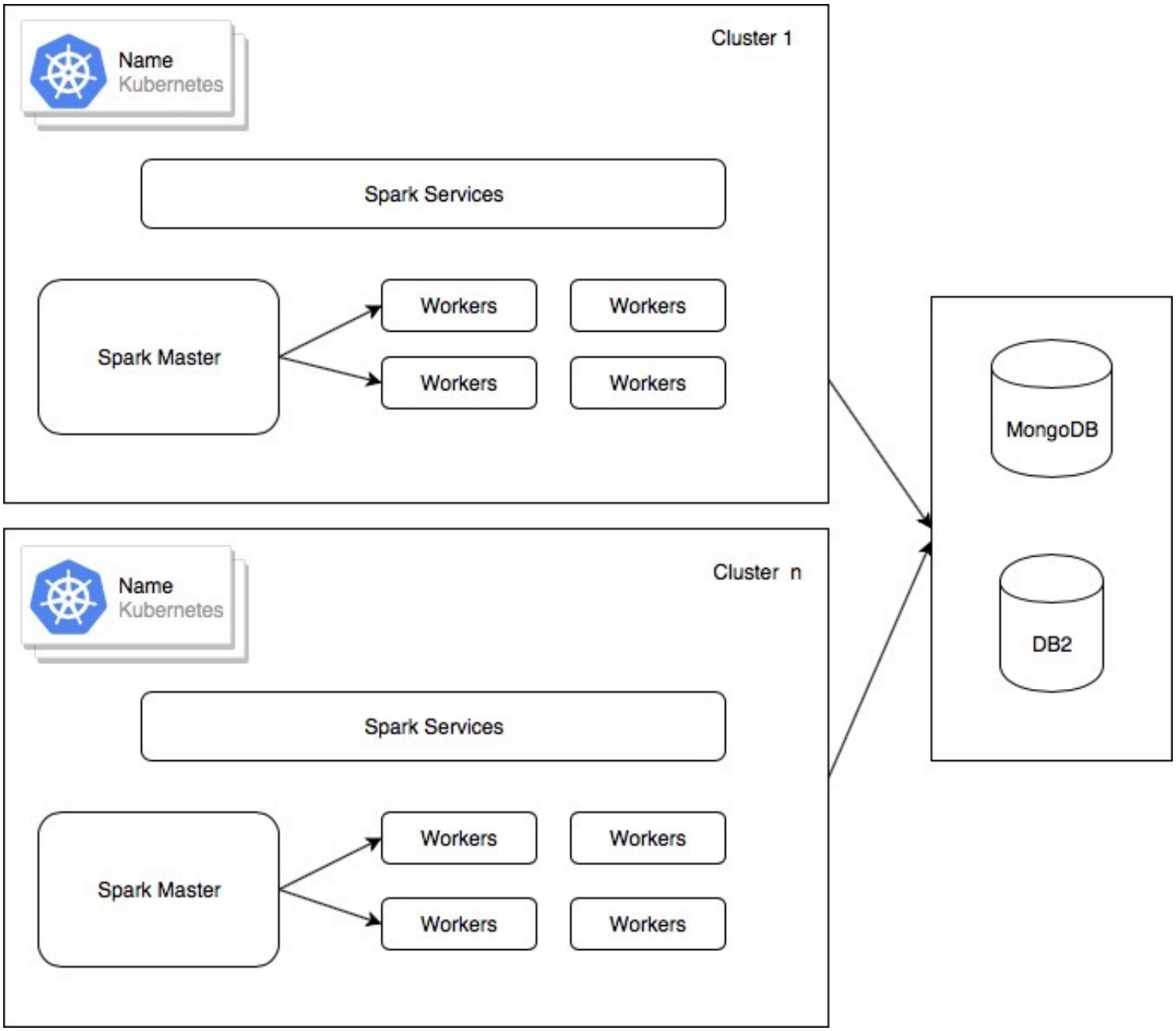
1. Servicios de conectividad al servicio cloud.
2. Servicios de invocación a los micro servicios ofrecidos.
3. Responsable de la lógica a utilizar cuando se reciben solicitudes de acceso a los servicios.
4. Servicio de visualización de la aplicación vía frontend.
5. Almacenamiento de toda la información necesaria, tanto en tiempo real como no, tanto información estructurada como semi estructura o no estructurada.
6. Todo lo relacionado con el procesamiento y análisis de la información.
7. Servicios comunes como podrían ser los relacionados con seguridad o gobierno del dato.

Ahora colocamos sobre los componentes los requerimientos funcionales o no funcionales.



6. Modelo operacional físico (cloud).

Modelo operacional físico:



7. Documento de decisiones arquitecturales.

Las decisiones arquitecturales serían las siguientes:

Decisión arquitectural DA-001: Uso de arquitectura Lambda con capa Batch y Streaming.

Subject Area	Diseño	**Topic**	Lambda
Architectural Decision	Utilización de arquitectura de referencia Lambda para separar capa Batch y capa Streaming.	**ID**	DA-001
Issue or Problem Statement	Decidir el tipo de arquitectura.		
Assumptions	-		
Motivation	Decidir el tipo de arquitectura, y conseguir una separación Batch y streaming, modularizando y haciendo lo más flexible posible la solución.		
Alternatives	Como arquitecturas de referencia destaca además de la Lambda la arquitectura Kappa donde sólo hay flujo streaming, ya que el Batch sería igual o sin importancia.		
Decision	Se decide utilizar la arquitectura Lambda, separando el flujo Batch del flujo streaming.		
Justification	Además de que vía cliente ha venido como recomendación, se piensa que es la mejor arquitectura para un caso de sistema/modelo de recomendación ya que mientras en la capa streaming se van usando los modelos con los datos recogidos, en la capa Batch se van entrenando y mejorando con todos los datos recogidos.		
Implications	-		
Derived requirements	RNF-003, RNF-004		
Related Decisions	-		

Decisión arquitectural DA-002: Uso de Kubernetes en infraestructura Cloud.

Subject Area	Infraestructura	**Topic**	Kubernetes, Cloud.
Architectural Decision	Utilización de clusters de contenedores gestionados por Kubernetes en Cloud.	**ID**	DA-002
Issue or Problem Statement	Decidir la infraestructura y tipo de plataforma para la solución.		
Assumptions	Asumimos que sería una solución consensuada con el entorno actual y que suele utilizar la empresa en otras soluciones, según lo hablado con cliente.		
Motivation	No generar más inmovilizado para la empresa, y conseguir buena escalabilidad.		
Alternatives	En lugar de Cloud se podría montar una solución física, y no usar una solución con contenedores.		
Decision	Utilización de solución Cloud, y con uso de clusters dockers gestionados por Kubernetes.		

Justification	Con el uso de Cloud nos ahorramos el tener más inmovilizado, se ahorra en costes y en gestión ya que la solución Cloud transparente en cierto modo algunos aspectos como la seguridad. Además la solución Cloud unida al uso de contenedores gestionados por Kubernetes nos proporciona una gran capacidad y facilidad de escalado, lo cual es básico en una aplicación de este tipo donde no conocemos de inicio los clientes y puede ir subiendo de forma incremental.
Implications	-
Derived requirements	RNF-007, RNF-009
Related Decisions	-

Decisión arquitectural DA-003: Uso de Spark para el análisis y modelos de recomendación.

Subject Area	Procesamiento	**Topic**	Spark
Architectural Decision	Utilización de Spark para los motores de recomendación, procesamiento y análisis.	**ID**	DA-003
Issue or Problem Statement	Tecnología a utilizar para la construcción de modelos y análisis de la información generada en la plataforma.		
Assumptions	-		
Motivation	Conseguir la mejor manera para procesar la información y realizar los modelos, en términos de eficiencia, velocidad y capacidad.		
Alternatives	Se podría utilizar una solución más basada en Hadoop, con uso de herramientas como Storm, Hive o HBase.		
Decision	Utilizar Spark como tecnología de procesamiento y análisis.		
Justification	Consideramos que Spark es la solución adecuada, que nos proporciona tanto para Batch como para Streaming lo que necesitamos, con el uso por ejemplo de Spark ML y Spark Streaming, además que tiene una integración nativa con Kubernetes y encaja a la perfección en el sistema propuesto.		
Implications	-		
Derived requirements	-		
Related Decisions	DA-002		

8. Conclusiones.

Como se ha podido ver existen en el mundo relacionado al Big Data multitud de tecnologías y herramientas que pueden darte aquello que necesitas, o en gran medida, por lo tanto nunca hay una solución arquitectural única y hay que intentar ver aquello que mejor puede funcionar en conjunto para la solución o aquello que haga hincapié en los puntos más importantes de los requerimientos de la aplicación.

En este caso se dispone de información sobre la empresa cliente y su entorno, y su infraestructura, por lo que ayuda a tomar ciertas decisiones y tener en cuenta ciertas herramientas para aprovechar la infraestructura y soluciones actuales.

No se ha hecho hincapié o profundizado en ciertos aspectos más técnicos como los algoritmos a utilizar para los modelos y las distintas recomendaciones, o como habría que usar el caching para gestionar la última búsqueda realizada. Podría profundizarse bastante más pero la idea era presentar la idea general a nivel más alto.