Arquitectura de Sistema de Banca por Internet

Proyecto de Diseño de Solución

[BP]

Fecha de elaboración: 2024-07-27

Autor: Rodrigo Francisco Mendoza Melgar

Version: 1.0

Indice

1. Diagrama de contexto

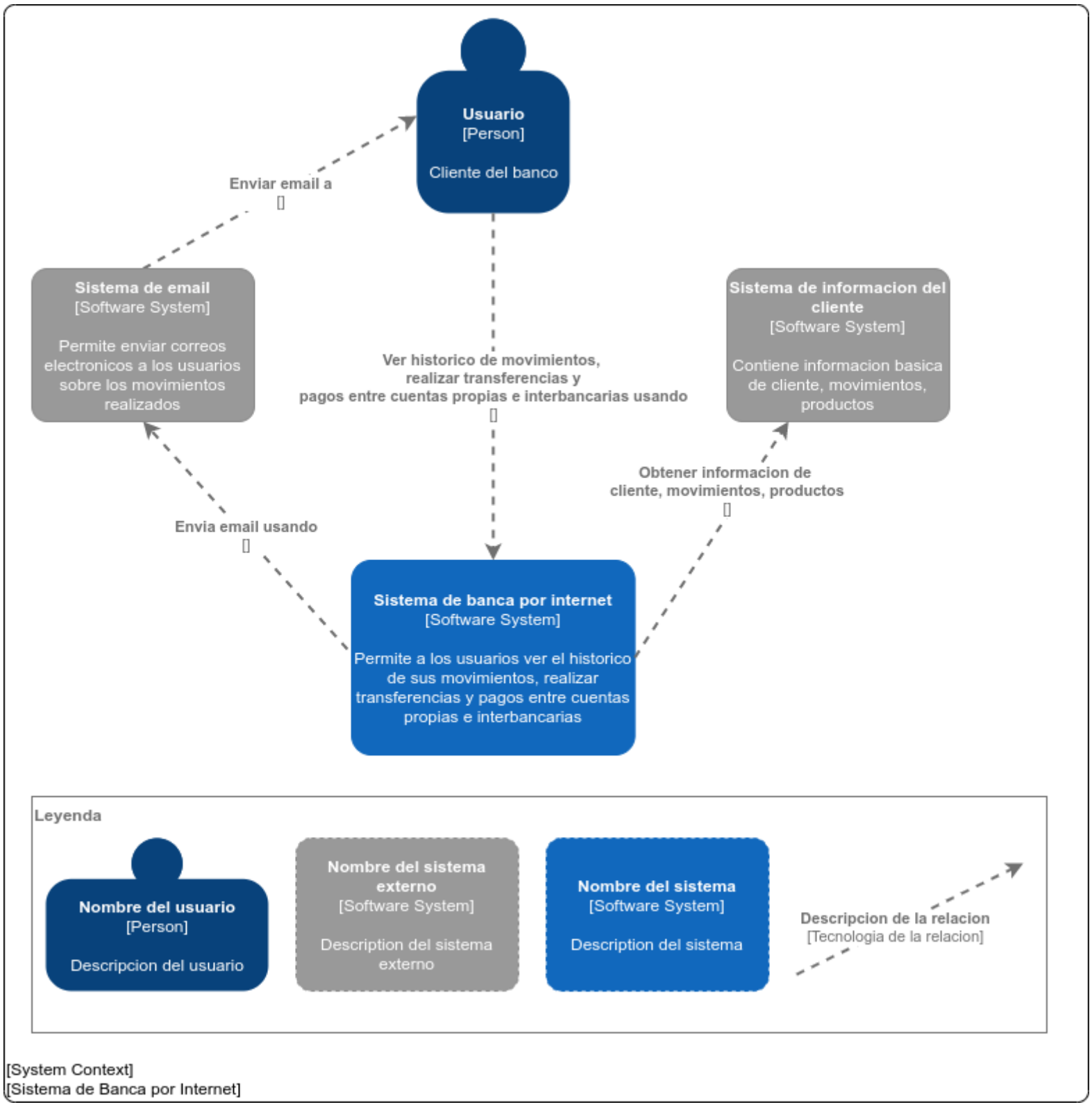

Figura 1: Diagrama de contexto

El siguiente diagrama de contexto representa el sistema de banca por internet y muestra las siguientes relaciones entre sus componentes:

1. Usuario : Representa al cliente del banco que interactúa con el sistema.
2. Sistema de banca por Internet: Es el componente central que permite a los usuarios ver su historial de movimientos, realizar transferencias y pagos entre cuentas propias e interbancarias.
3. Sistema de información del cliente: Contiene la información del cliente, movimientos y productos.
4. Sistema de email: Permite enviar correos electrónicos a los usuarios sobre los movimientos realizados.

Las interacciones principales son:

- El **usuario** accede al **sistema de banca por Internet** para ver su historial y realizar operaciones.
- El **sistema de banca** obtiene la información necesaria del **sistema de información del cliente**.
- El **sistema de banca** utiliza el **sistema de email** para notificar a los usuarios sobre sus movimientos.

2. Diagrama de contenedores

El diagrama de contenedores se presenta de 4 formas para tratar de reducir la complejidad visual por la cantidad de elementos y relaciones

1. Diagrama de contenedores [Figura 2], muestra todos los elementos y relaciones en el diagrama

2. Diagrama de contenedores agrupando microservicios en un kubernets [Figura 3], agrupa todos los microservicios en un cluster kubernetes, con el objetivo de visualizar los otros componentes como: bases de datos, message brokers, sistemas externos

3. Diagrama de contenedores de los microservicios clientes y productos [Figura 4], muestra solo las relaciones y los elementos que interactuan con los microservicios clientes y productos

4. Diagrama de contenedores de los microservicios cuentas, transferencias, movimientos y notificaciones [Figura 5], muestra solo las relaciones y los elementos que interactuan con los microservicios cuentas, transferencias, movimientos y notificaciones

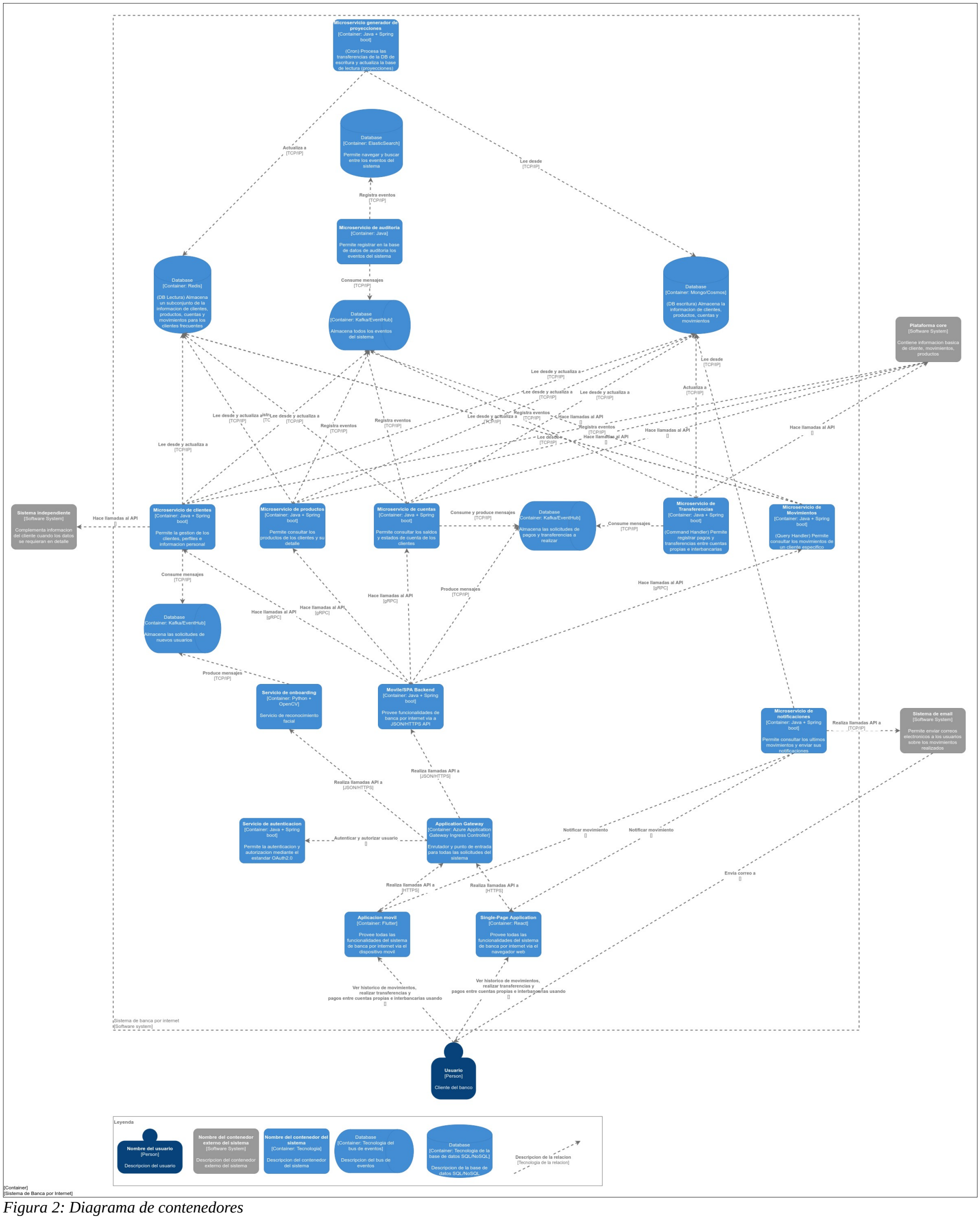

Figura 2: Diagrama de contenedores

El diagrama muestra las interacciones entre estos componentes, indicando cómo se comunican entre sí para proporcionar las funcionalidades del sistema bancario en línea.

Se emplearon los patrones

- CQRS: Para el registro y consulta de movimientos, con el objetivo de escalar ya sea la base de datos de escritura o lectura, teniendo asi un mejor rendimiento, asociado a eso, podemos poblar multiples bases de datos de lectura (para analitica), y poder tener las bases de datos en diferentes tecnologias según las necesidades.
    - Una base de datos de escritura: Para persistir solo identificadores y datos no redundantes en otras fuentes (colecciones, tablas, …), es decir esta base de datos puede estar normalizada hasta la tercera o cuarta forma. Recordar que esta DB puede ser opcional porque ya almacenamos los eventos en otra base de datos, y podemos reconstruir la db de lectura a partir de esos eventos, pero con esta base de datos podemos hacer ciertas consultas que tardarian mas en una base de datos de solo eventos, con ello tenemos un mejor rendimiento ya que mantiene el ultimo estado del sistema.
    - Una o mas base de datos de lectura: Para persistir informacion redundante, con el objetivo de no realizar “joins” y/o “vistas” en base de datos, ya que esto puede aumentar la latencia o saturar la base de datos. Esta base de datos puede estar desnormalizada.
- EventSourcing: Para el registro de los eventos del sistema, apartir de esta base de datos podemos reconstruir la base de datos de lectura o escritura. Manejamos una auditoria total del sistema, ya que registramos todos los eventos de los usuarios. Esta base de datos es la fuente de la verdad.
- BackEnd for FrontEnd: Agregamos tambien este patron para poder controlar mas la escalabilidad e la independencia en la implementacion de las caracteristicas de los diferentes frontend.

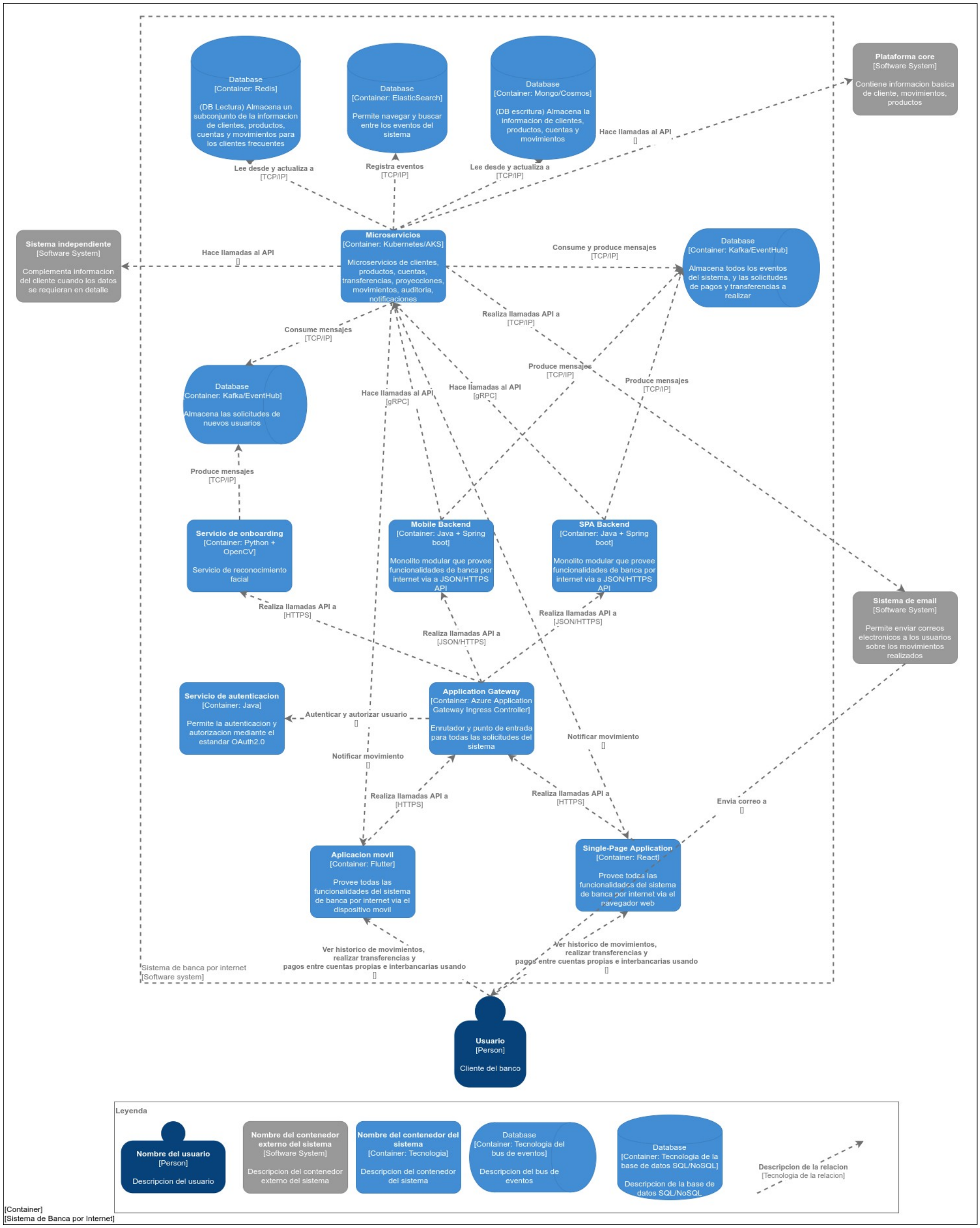

Figura 3: Diagrama de contenedores agrupando microservicios en un kubernets

Se especificaron las tecnlogias

- Base de datos de lectura en redis: Una base de datos en memoria que permite persistir la informacion en diversas estructuras de datos. Tiene uno de los mejores rendimientos.
- Base de datos de escritura en mongo: Una base de datos NoSQL orientada a documentos y clave valor, escala mejor que una base de datos relacional

- Base de datos de auditoria en elasticsearch: Para poder realizar busquedas entre los eventos del sistema. Tambien es una base de datos orientada a documentos, que permite versionar la data y cambiar la estructura.
- Message broker en kafka: Por el particionado, tiempo de vida de los mensajes, la escalabilidad, los commits manuales, los topicos, grupos de consumidores, entre otras caracteristicas.
- Application gateway: Actua como ingress controller para nuestro AKS, podemos tener tambien multiples application gateway para diferentes necesidades como WAF, API Management, entre otros.

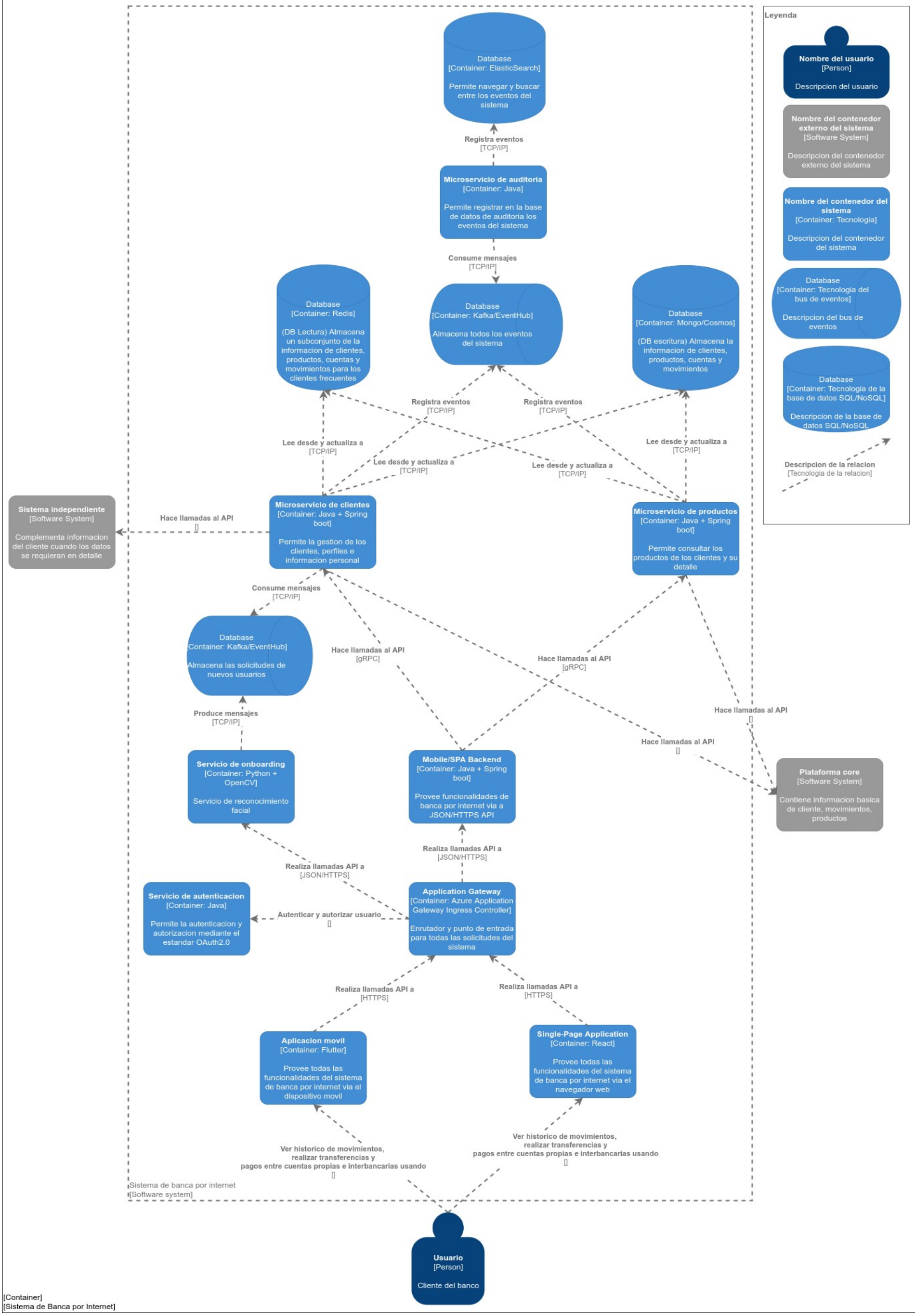

Figura 4: Diagrama de contenedores de los microservicios clientes y productos

Para el cliente nuevo realizamos el onboarding y su registro en nuestro sistema y la plataforma core, le agregamos un message broker por si en cierto momento se genera una alta demanda del microservicio de clientes y no queremos perder los nuevos registros. Tambien enviamos el nuevo cliente a la plataforma core.

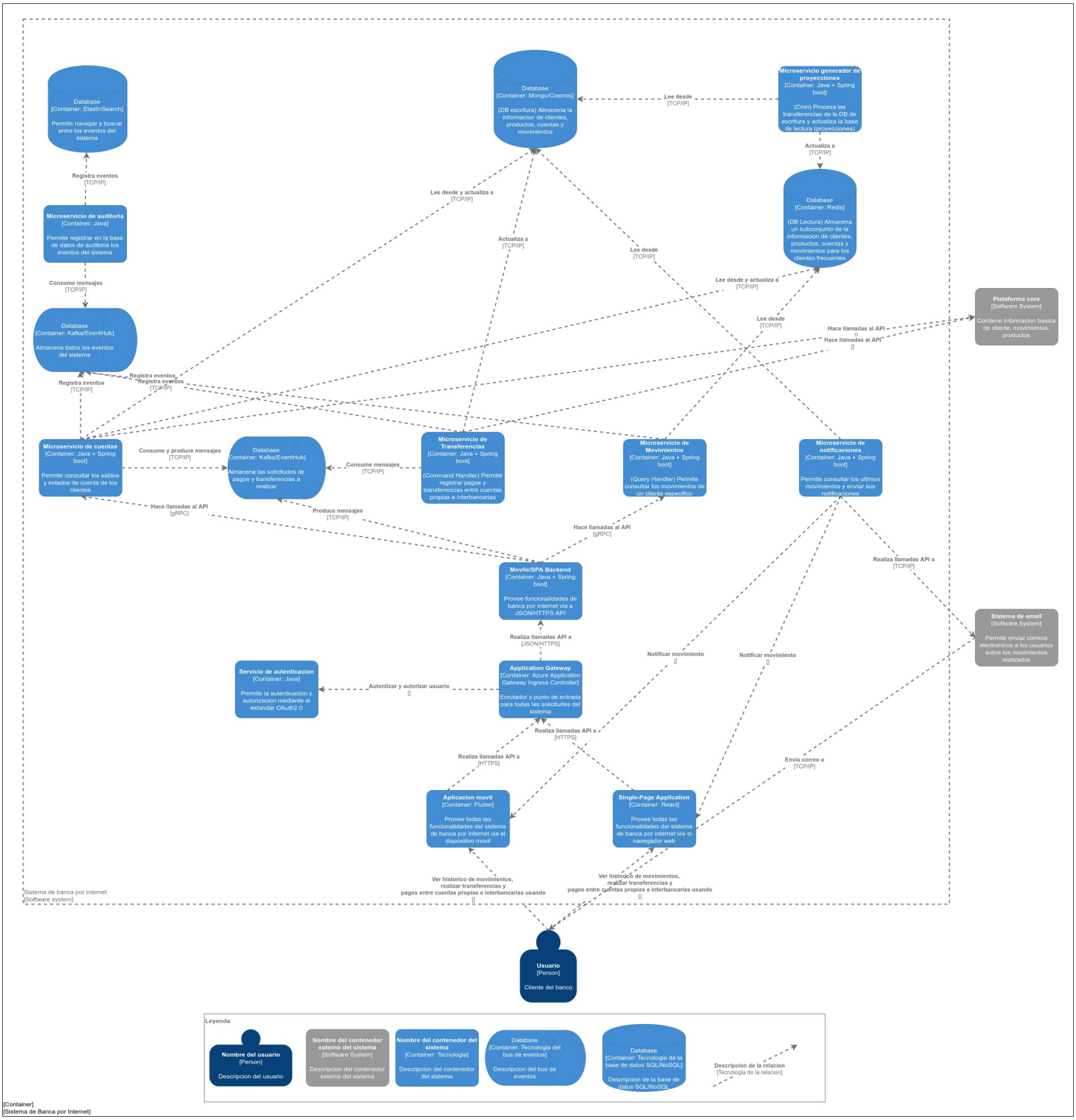

Figura 5: Diagrama de contenedores de los microservicios cuentas, transferencias, movimientos y notificaciones

- Registro de movimientos
  - El backend movil o SPA, registra solicitudes de pagos o transferencias en el message broker
  - Los microservicio de cuentas consumen estas solicitudes y valida si el cliente dispone de la cuenta y el saldo. Si todo es conforme vuelve a registrar en otro topico/canal la solicitud de pago con una verificacion y si es necesario le agrega mayor informacion al mensaje
  - Los microservicios de transferencias consumen estas solicitudes verificadas, las envia a la plataforma core y si todo esta correcto las registra en su base de datos, luego registro el evento realizado en el message broker de eventos
  - Los microservicios de auditoria escuchan los mensajes del broker de eventos y los registra en su base de datos de auditoria
  - Los microservicios generador de proyecciones se ejecutan cada minuto y consulta los ultimos movimientos de ese minuto, luego del conjunto encontrado genera sus proyecciones (consulta los otros microservicio si es necesario para agregar mayor informacion) y las almacena en la base de datos de lectura

    - Los microservicios de notificaciones al igual que los generadores de proyeccion se ejecutan cada cierto tiempo y revisan los ultimos movimientos de un periodo de tiempo y envian la notificacion al SPA y al movil, asu vez lo envian al sistema de emails
- Consulta de movimientos
    - El backend movil o SPA, consulta los movimientos en algun microservicio de movimientos, este consulta a la base de datos de lectura y retorna los movimientos encontrados para ese cliente especifico

3. Diagrama de componentes

En los posteriores diagramas se usa la arquitectura limpia para definir los componentes de cada microservicio, servicio o frontend, con el objetivo de cambiar rapidamente de tecnologia (librería, framework, …)

Ademas en el caso de los microservicios se definen los circuit breakers en los microservicios que registran datos, ya sea en la base de datos o en la plataforma core.

Para cada capa se maneja un modelo de datos y mappers para mover datos entre los otros modelos.

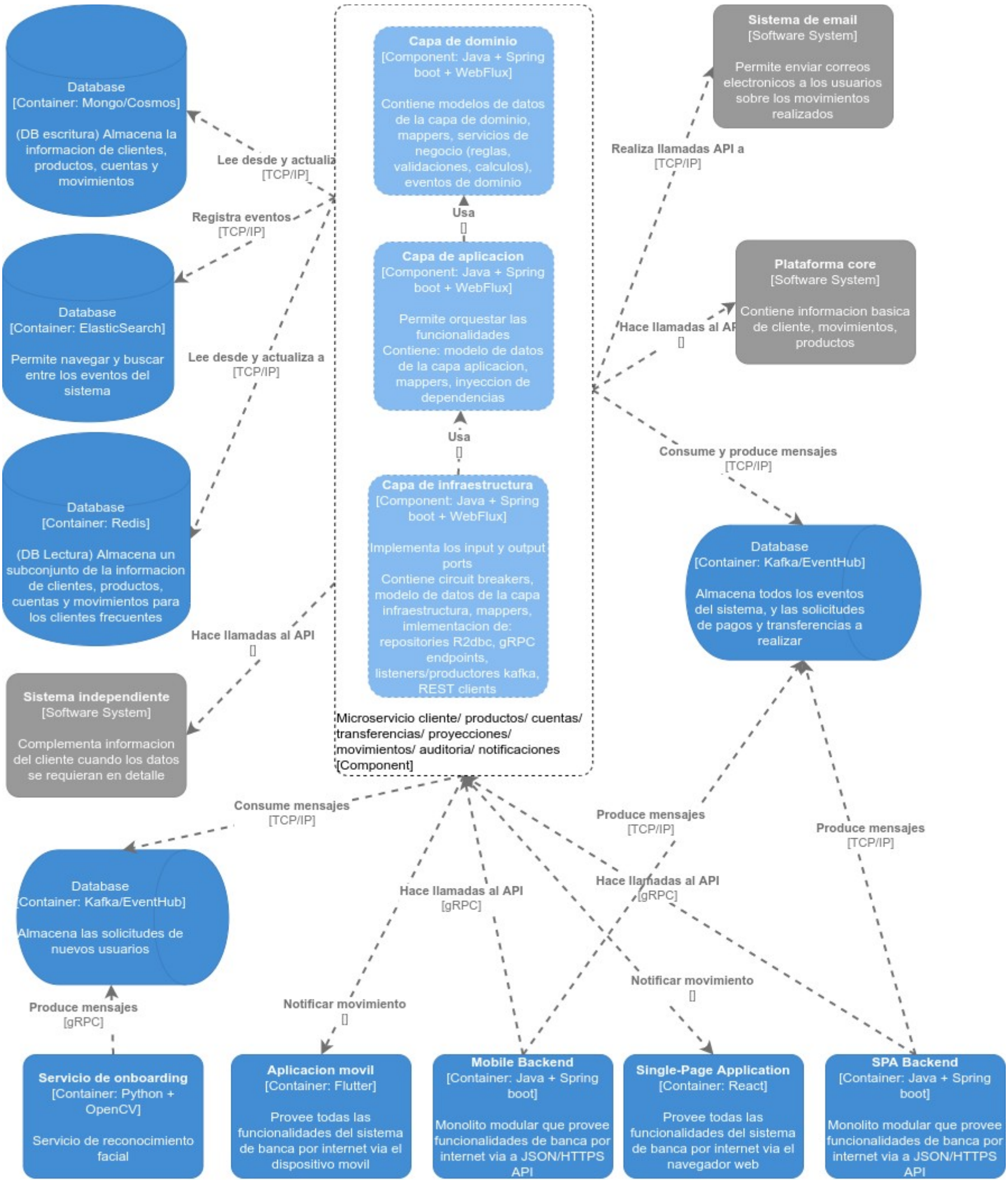

Figura 6: Componentes de los microservicios

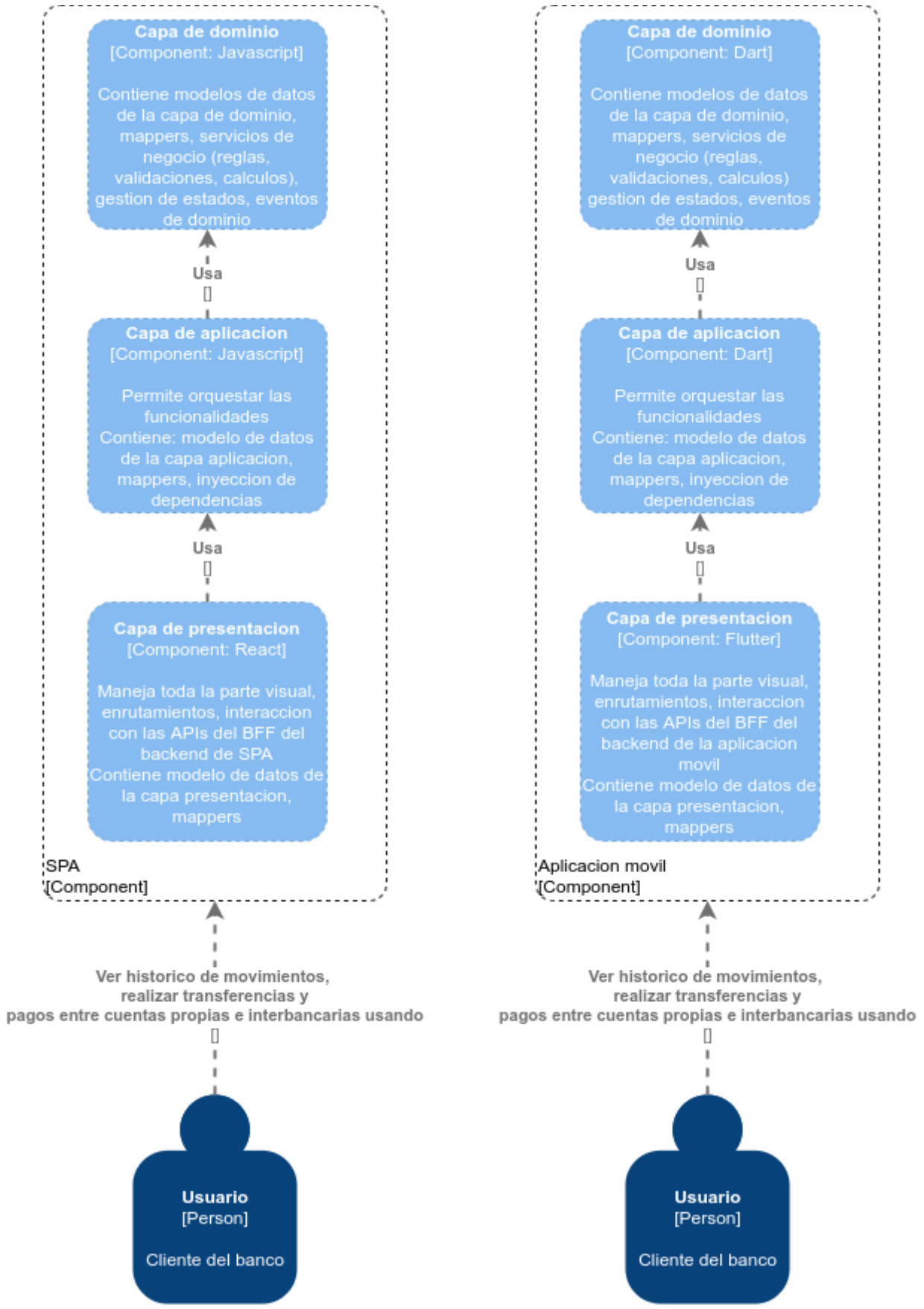

Figura 7: Componentes del SPA y de la aplicacion movil

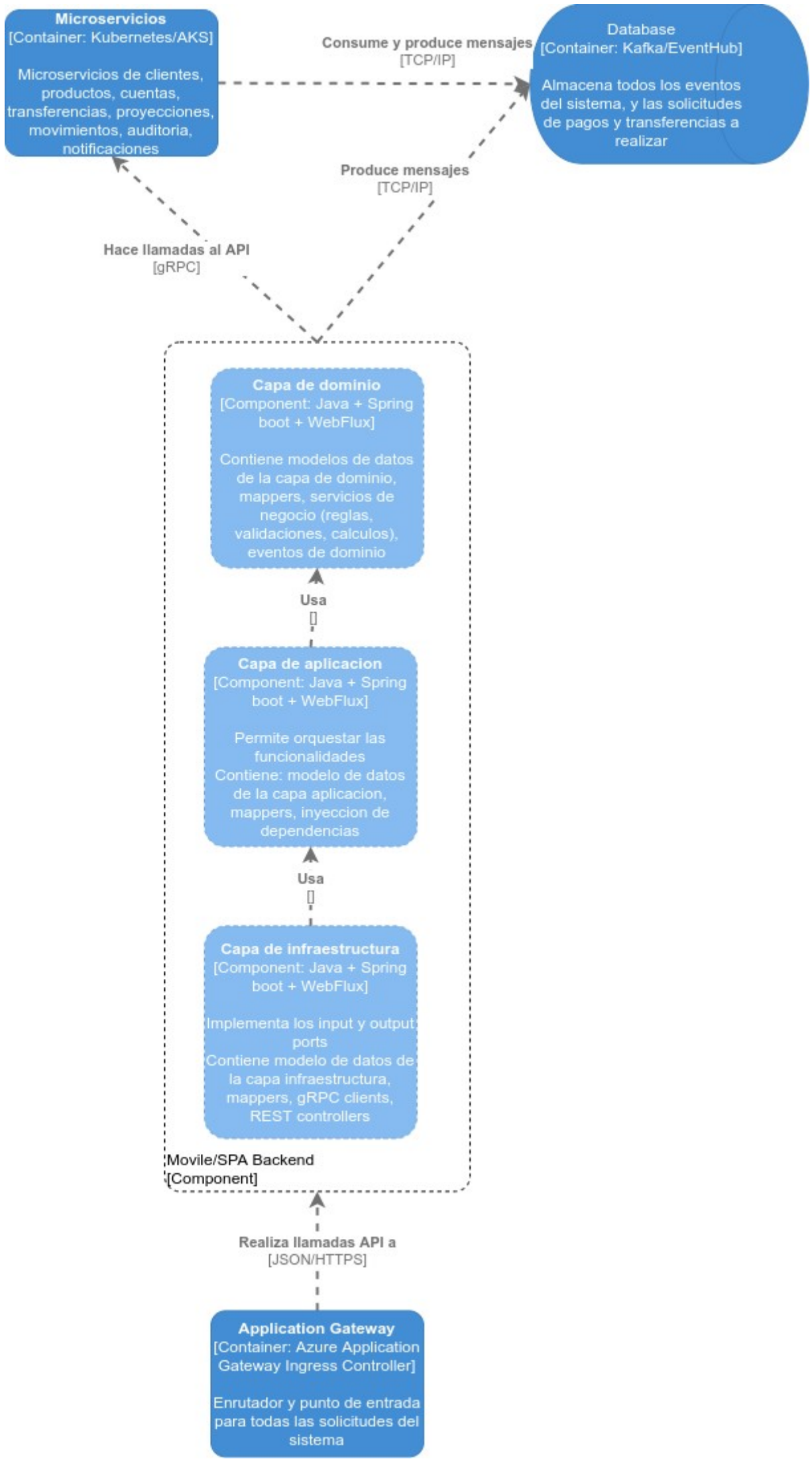

Figura 8: Componentes del movil o SPA Backend

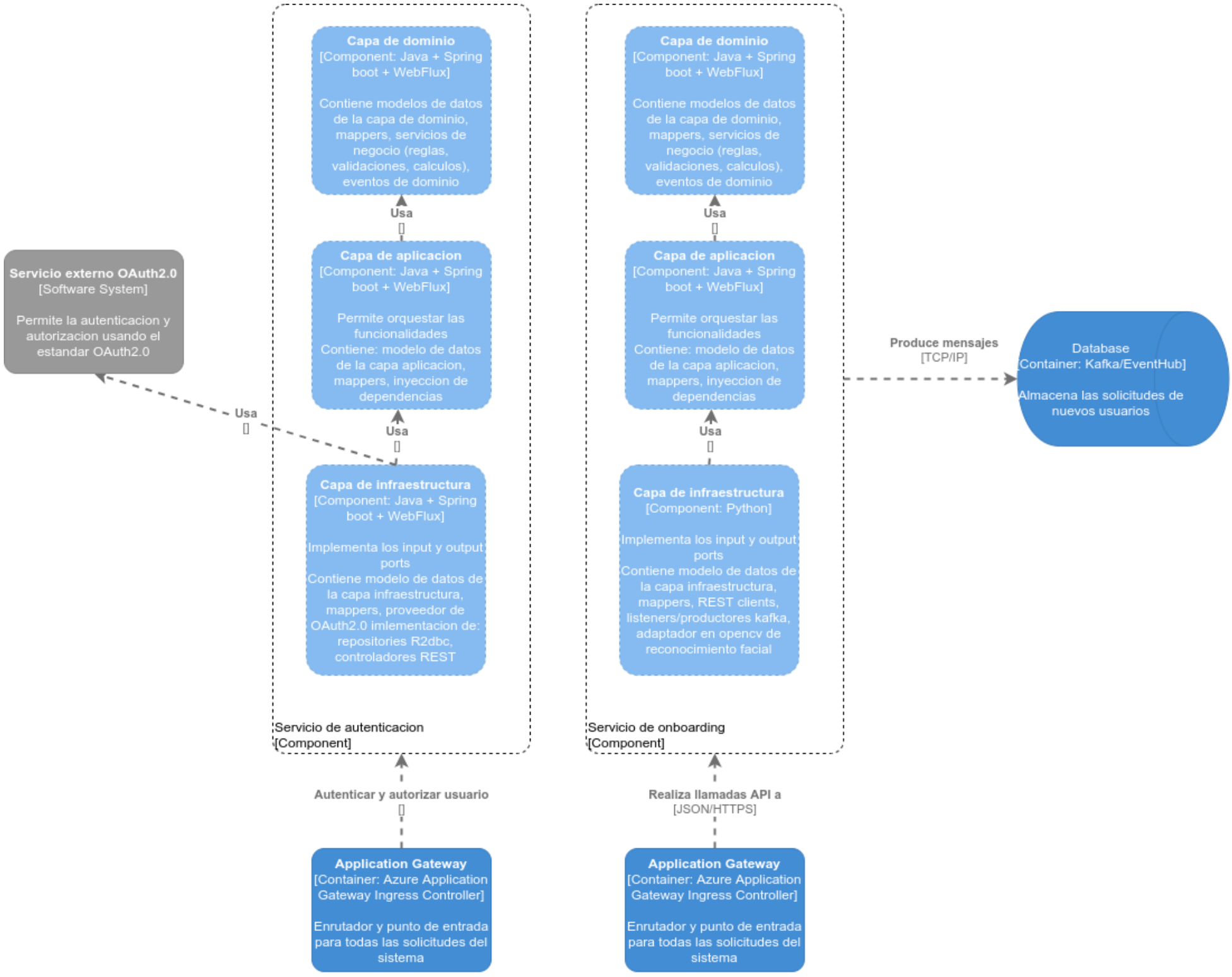

Figura 9: Servicio de autenticacion y onboarding