Tabla de Contenido

Tabla de gráficos

Diagrama de Contexto

- Modelo general de la solución planteada, en el cual se observa la iteración del usuario o Customer con los sistemas Core del banco que permite obtener la información del usuario, cuentas, balances y transferencias entre sus cuentas. Es importante comentar que, al realizar una transferencia, se debe notificar al usuario por lo cual se plantea dos mecanismos que son:
    - Envío de notificación por correo electrónico, el cual puede ser un servicio interno o externo.
    - Envío de mensajes de textos a dispositivos móvil, conocido como sms, se recomiendo utilizar el servicio de AWS Amazon SNS que permite implementar fácilmente este tipo de mensajería.

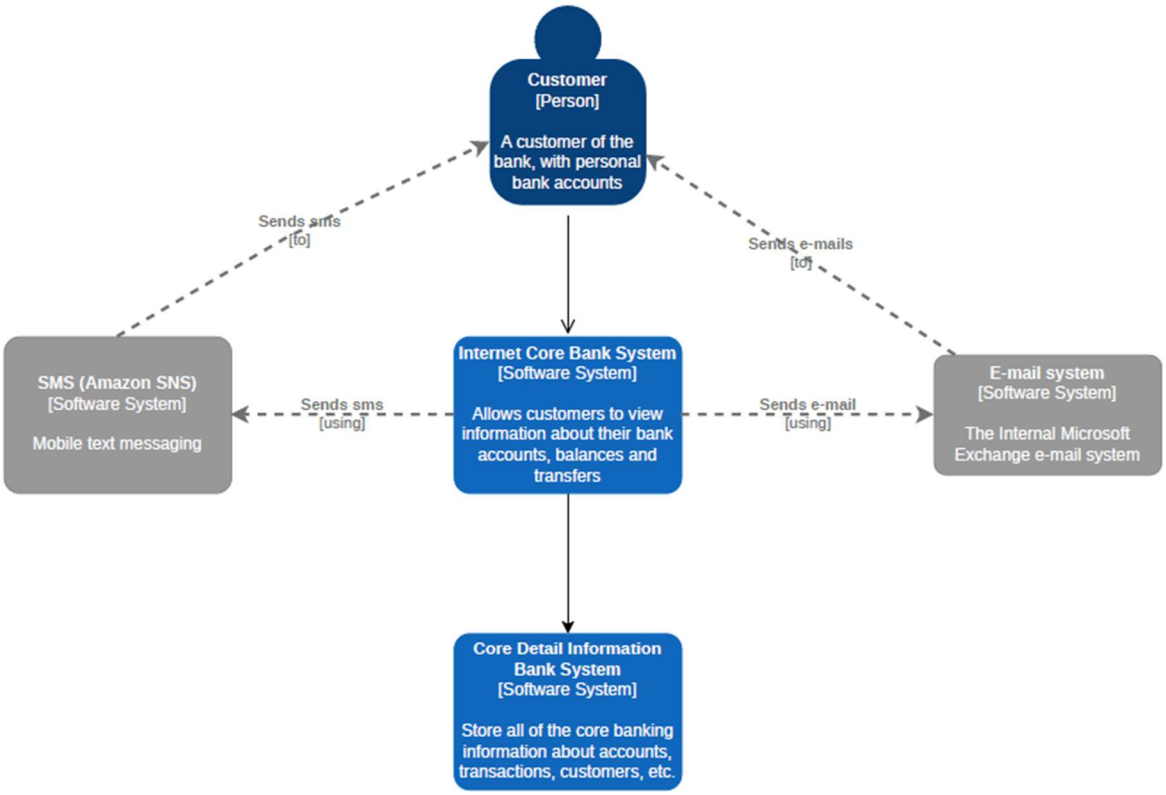

Figura 1 Diagrama de Contexto

Diagrama de Contenedores

En este diagrama se puede observar la tecnología que se plantea para la implementación de este proyecto como se detalla a continuación.

- SPA: se plantea utilizar JavaScript con React o Angular como tecnología que nos permite desarrollar un sitio web con características óptimas para que el usuario pueda utilizar la página de manera sencilla.
- Mobile App: Se recomienda utilizar React Native o Angular que son las tecnologías más utilizadas y que nos permite tener aplicaciones multiplataforma sin tener una afectación considerable en el rendimiento de la misma. Flutter una tecnología de Google también es recomendada por su sencilles en el aprendizaje y actualmente va incrementando su adopción para aplicaciones móviles.

  Se recomienda utilizar microfrontends que nos permite desagregar las funcionalidades y adicionalmente permite que la aplicación permita tener una arquitectura desacoplada lo que nos da flexibilidad en el caso de actualizar backend o front end.

- Como orquestador se puede utilizar MODYO que nos permite utilizar Widgets los cuales son componentes desarrollados en Angular o React (Reac Native para funciones específicas del dispositivo) y que se puede utilizar tanto en aplicaciones móviles, como web, lo que facilita su mantenimiento y desarrollo ágil con equipos grandes.
- A nivel de servicios se recomienda utilizar Spring Framework lo cual nos ayuda para implementar MVC en nuestra aplicación, para la creación de las apis tipo REST, o si es necesario se puede utilizar Spring Boots para crear microservicios lo cual nos facilita el mantenimiento y agilidad en el desarrollo.
- Se utiliza un ESB es decir un bus empresarial con la finalidad que se pueda integrar a cualquier Core Legacy que se pueda tener, ya que aquí podemos realizar transformaciones de mensajería que recibe como entrada un Json y que puede transformarlo a una mensajería por ejemplo XML o cualquier otro tipo de mensajería propia que utilice el Core, con lo cual se permite desacoplar los sistemas unos de otros, se plantea utilizar los servicios de AWS como Route53, Amazon Api Gateway, VPC, NLB, Fargate. Se va a utilizar una arquitectura lo mas serverless posible, no solo porque permite reducir costos de hardware y licenciamiento por ejemplo si se utilizara un IBM Datapower, sino porque se facilita el no tener que preocuparse de provisionar y operar servidores.
- Al contar con el mecanismo de autenticación con el estándar OAuth 2.0, se recomienda el flujo en el cual en primera instancia cualquier petición primero se valida en el api Gateway en el cual se valida el nivel de acceso de cada petición, una vez validado se puede utilizar Keycloack que nos permite implementar la autenticación a aplicaciones de servicios seguros con un mínimo esfuerzo, adicionalmente nos permite el manejo de la validación y generación del token Access.

  Se puede utilizar Spring security para securizar los endpoint de las Apis expuestas en la aplicación y con el cual se enviará el request Access Token para que las Apis puedan acceder a la información.
- Finalmente, si se pretende migrar a la nube se recomienda implementar OAuth 2.0 con el flujo de autorización para Amazon Cognito usando AWS lambda y Amazon DynamoDB.
- Seguridad: Para evitar ser vulnerados todos los datos de entrada sensibles deben ser validados, encriptados, la información como tarjetas de crédito, números de cuentas deben enviarse de manera encriptada al API para que en el servicio se des- encripte y se envíe a buscar la cuenta o tarjeta requerida para una transferencia.

  La comunicación entre las APIs y AWS se lo puede realizar mediante una VPN para que permita que la comunicación vaya cifrada mediante certificados TLS.

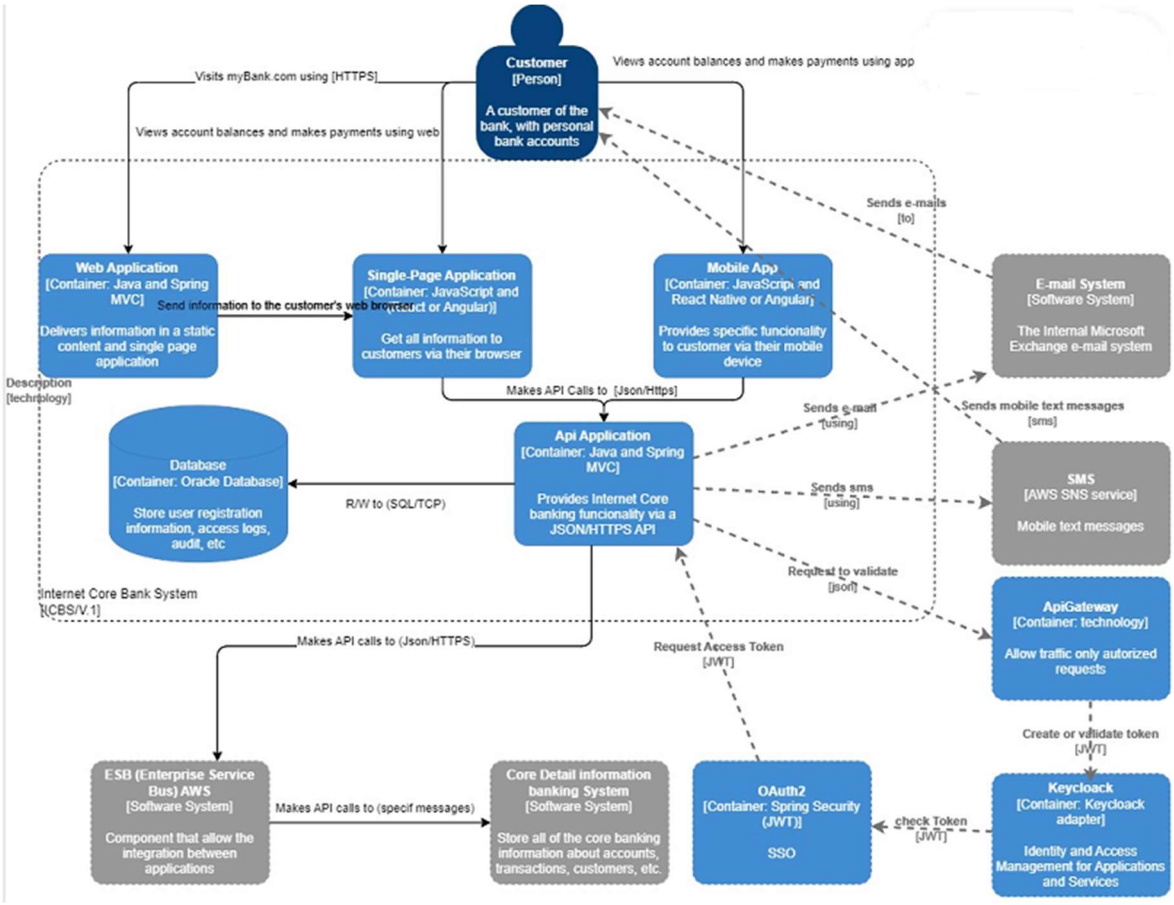

Figura 2 Diagrama de contenedores

Diagrama de Componentes (Autenticación web y móvil, Consulta de movimientos)

En este diagrama se puede observar cómo se realizaría el proceso de autenticación (SSO) y como se puede implementar en web y app (considerando el enrolamiento de un usuario y su mecanismo de autenticación que puede ser reconocimiento facial, biometría, etc).

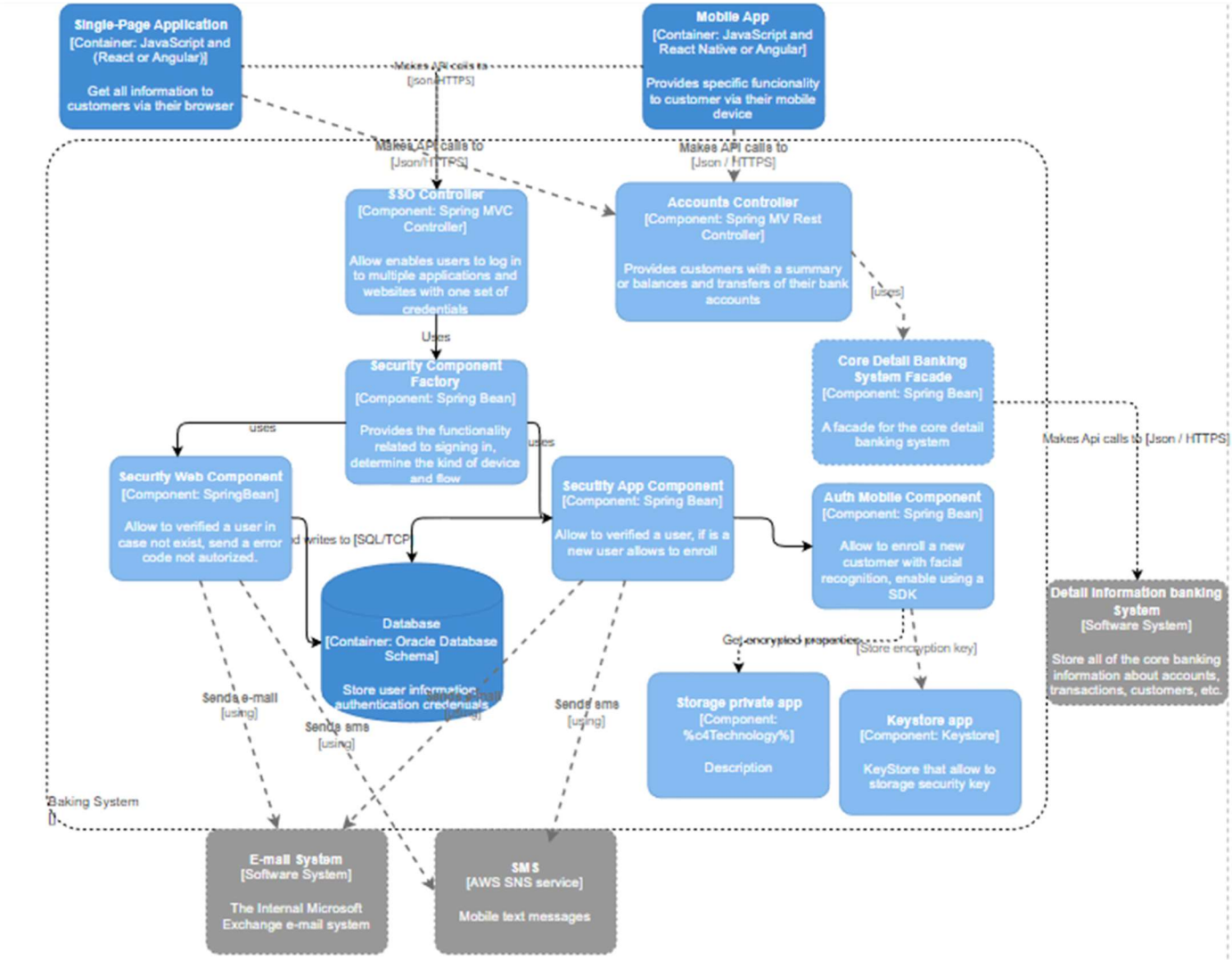

Figura 3 Diagrama de componentes Autenticación

Modelo de componentes (Transferencias)

- Adicionalmente se puede observar el modelo planteado para transferencias, pagos del cliente y como interactúa con los distintos componentes.

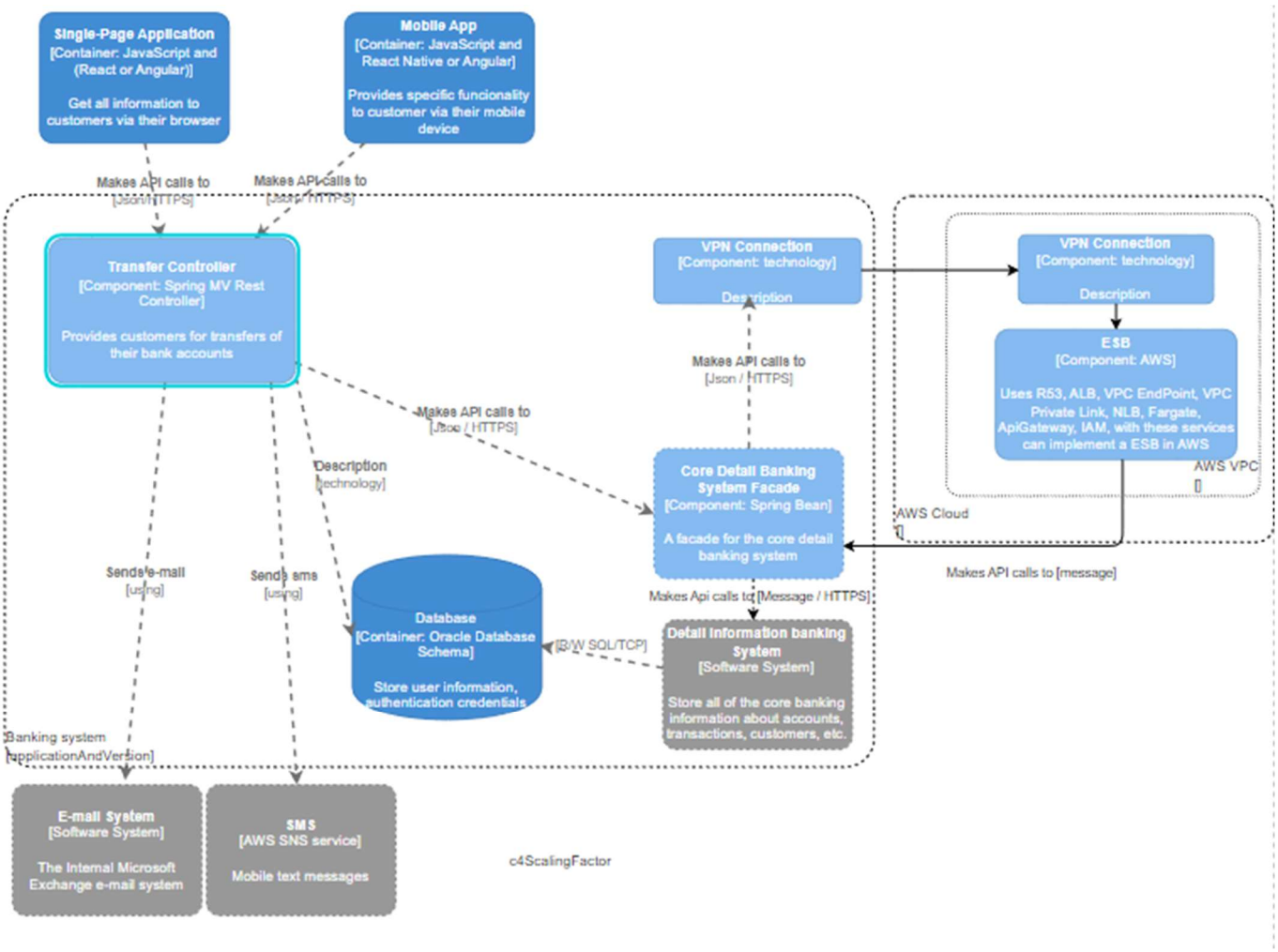

Figura 4 Diagrama de Componentes (Transferencias)

Modelo de componentes (Consulta movimientos)

- A continuación, se puede observar el modelo planteado para consulta de movimientos de cuentas, y como interactúa con los distintos componentes.

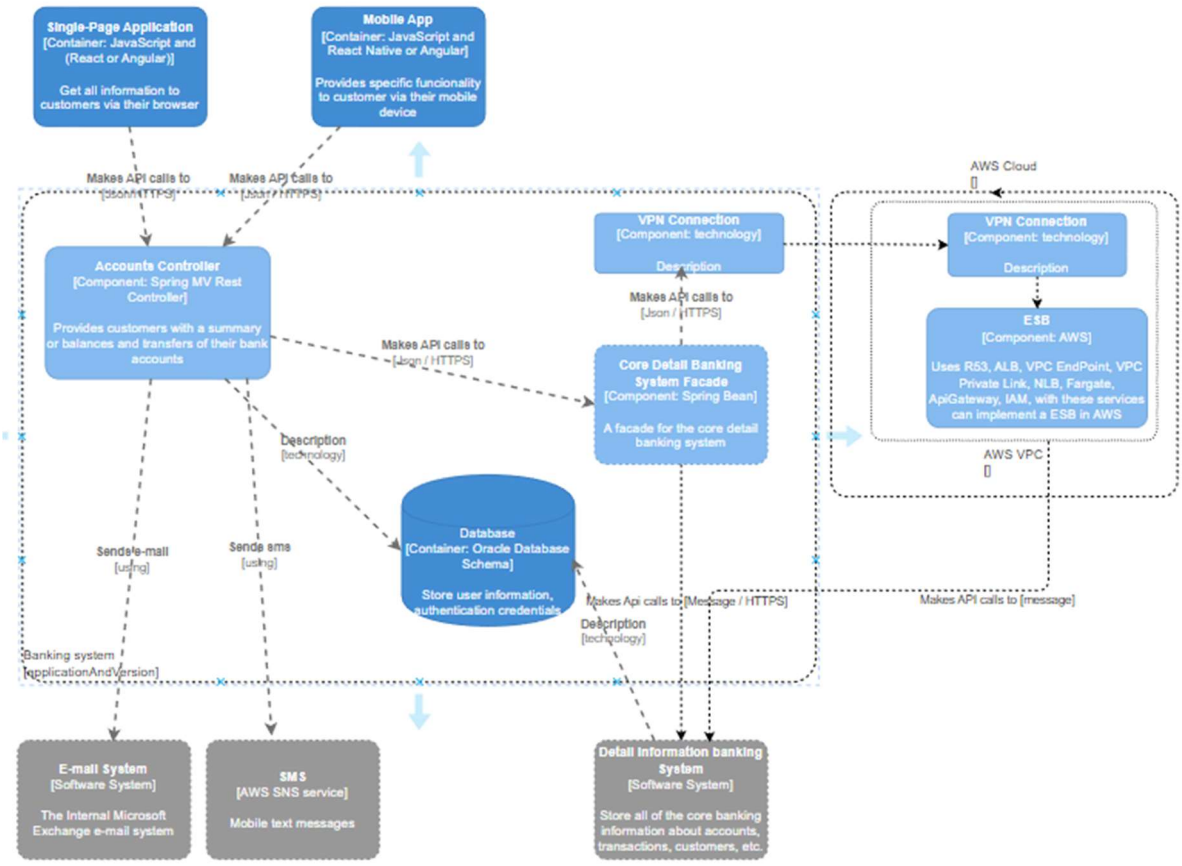

Figura 5 Diagrama de Componentes (Consulta movimientos)

Modelo de componentes (Auditoría)

- Finalmente se puede observar cómo se puede implementar la auditoría que nos permite registrar toda la actividad que realice un usuario, este registro se lo realizará mediante procesos asíncronos que evitan cargar procesamiento en especial cuando existe alta demanda. Se recomienda que en los archivos logs nunca se registre información en texto claro, la información sensible como tarjetas de crédito, números de cuenta, correo electrónico, número de celular, esta información se debe registrar enmascarada para evitar exposición innecesaria de esta información.
- El flujo planteado para el registro de auditoría es la siguiente, la API expuesta registra en el archivo log la información, Flink permite leer el archivo log y crea un mensaje al encolador ActiveMQ, el cual al procesar los mensajes va registrando en la base de datos las actividades del usuario con lo cual se puede extraer fácilmente la información.
- Base de datos, se recomienda utilizar un motor de base robusto como Oracle para registrar la información, la misma que se debe implementar mecanismos de seguridad para no registrar en tablas información sensible en texto claro, por ejemplo, los números de tarjeta de crédito deben ser tokenizadas para evitar vulnerabilidades o incorrecta manipulación de la información.

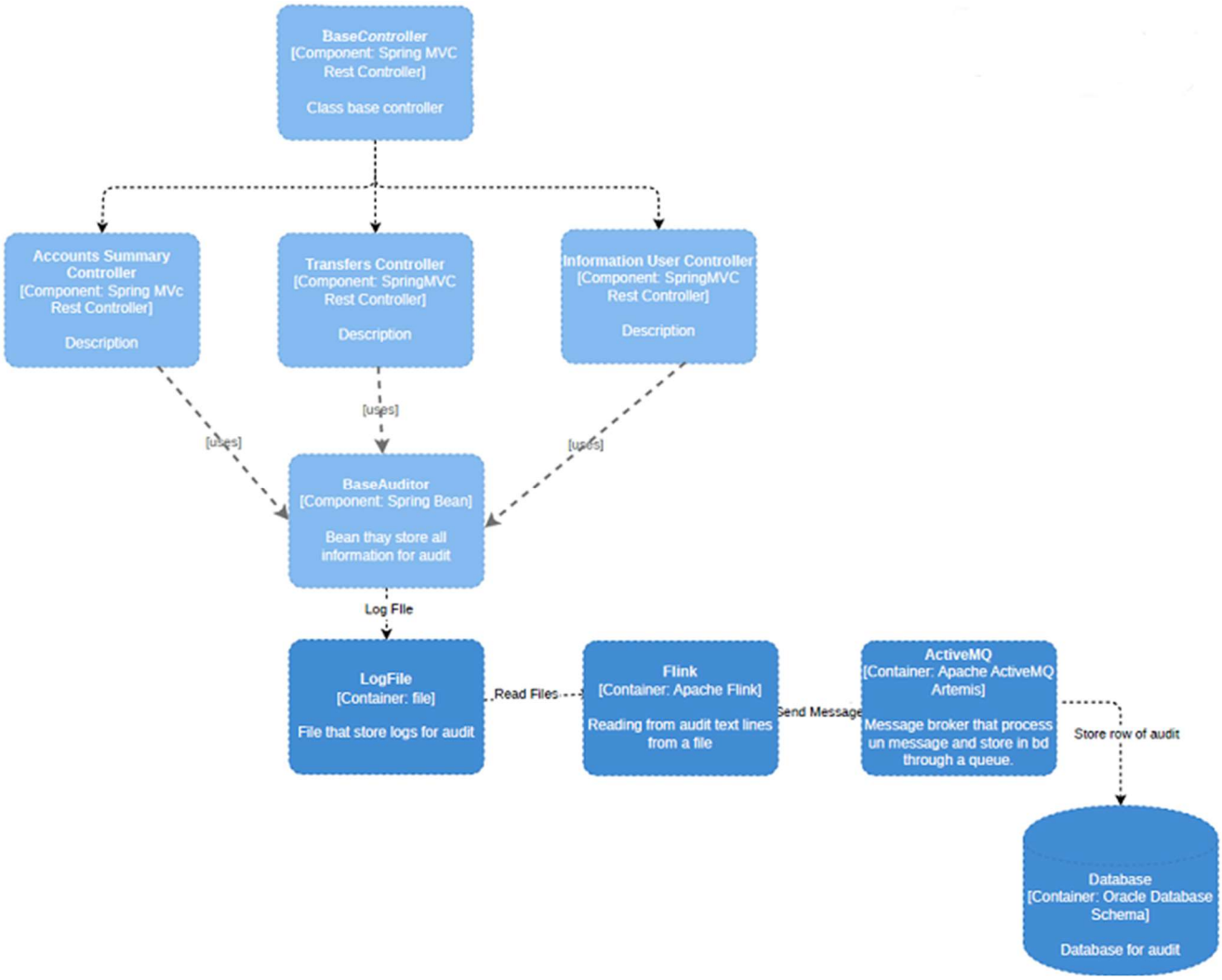

Figura 6 Diagrama de Componentes (Auditoría)

Diagrama de despliegue

- Se adiciona este diagrama en el cual se puede observar un flujo que nos permitirá implementar la integración continua y entregas continuas (CI/CD) de la aplicación con lo cual utilizando servicios de AWS podemos tener un esquema de alta disponibilidad, escalamiento automático, tolerancia a fallos y podemos monitorear fácilmente.
- Se plantea utilizar servicios de AWS como se detalla a continuación.
    - AWS CodeBuild que nos permitirá compilar el código fuente de la aplicación y generar el artefacto war que posteriormente automáticamente se generará la imagen docker que será registrada en AWS ECR.
    - AWS ECR nos permite registrar los contenedores Docker de manera sencilla y que se lo utilizará en el clúster de Kubernetes.
    - AWS CodePipeline es el que permite orquestar automáticamente si determina si se realizó algún cambio en el repositorio del código fuente y se genera el artefacto, por seguridad se recomienda niveles de aprobación para su ejecución. También nos permitirá realizar el deploy continúo lo que permite actualizaciones continuas.
    - AWS EKS, nos permite crear clúster de Kubernetes en la nube de AWS.
    - AWS CloudWatch Logs, nos permite monitorear los servicios.
    - AWS Fargate nos permite ejecutar los pods de Kubernetes y que provee bajo demanda la capacidad de los contenedores, lo que nos evita tener que provisionar o escalar grupos de máquinas virtuales, sino que lo realiza de manera automática, si es necesario se crea más instancias para que la aplicación responda correctamente.
    - Se configura CronJobs a los contenedores lo que nos permitirá ahorro de recursos ya que podemos configurar en que período de tiempo van a estar online nuestra infraestructura en la nube.

Con este esquema se pretende que la aplicación tenga alta disponibilidad ya que automáticamente al utilizar un clúster de Kubernetes se puede configurar el número de PODS mínimos y máximos que son requeridos de acuerdo con la demanda, es decir que automáticamente se crean instancias y permite que la aplicación se recupere rápidamente a fallos.

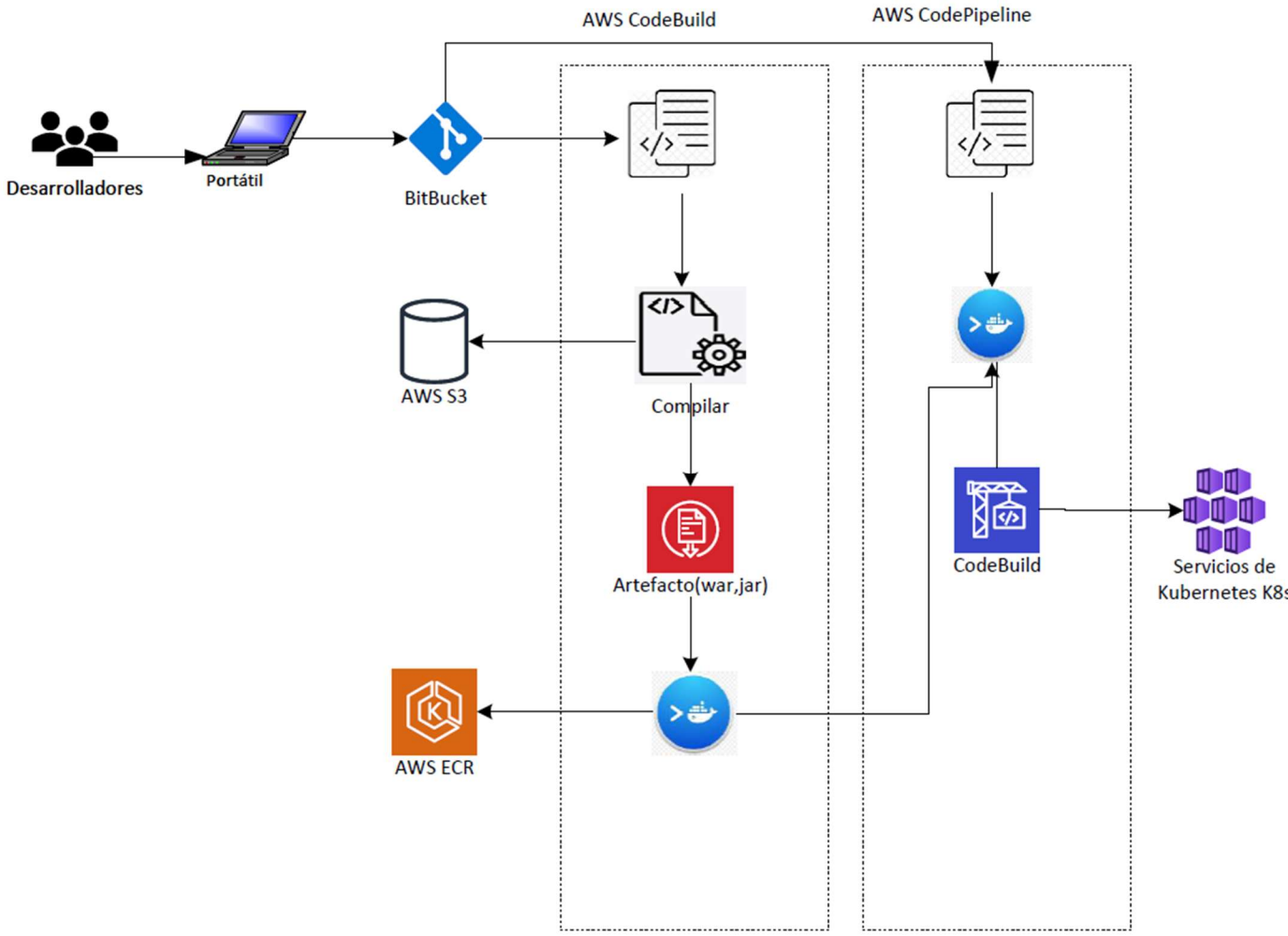

Figura 7 Diagrama de Despliegue

Manejo de costos

Al plantear la solución en la nube, para optimizar los costos se ha considerado los siguientes puntos:

- Se plantea una arquitectura serveless en lo que se pueda aplicar, con la finalidad de reducir costos tanto en hardware como licenciamiento del cliente. Un ejemplo es evitar el uso de un ESB que podría ser un IBM Datapower, un hardware costoso, y se plantea una solución en la nube optimizando recursos.
- Se configura CronJobs al clúster con la finalidad de optimizar recursos por ejemplo, se configura el horario en el cual la infraestructura va a estar disponible, en un ambiente de desarrollo se puede configirar con horario de 8 – 7pm y en la noche se apagan lo que genera ahorro.
- En un ambiente productivo se puede implementar políticas para que dependiendo del horario se baje automáticamente las instancias disponibles tanto en escalamiento vertical como horizontal dependiendo del componente que se requiera optimizar, con este tipo de medidas podemos optimizar recursos.
- La arquitectura planteada recomienda utilizar software open source como Java, Spring Framework, Spring Boots, Spring Security, Keycloak, Docker, Kubernetes, Apache Camel que nos permite reducir costos de licenciamiento del cliente y que a la vez nos permite integrarnos a la nube de AWS optimizando los servicios utilizados.