Reto Técnico Carlos Yanez - Devsu

Nombre: Carlos Alberto Yanez Rubio

A continuación se detalla cada uno de los diferentes diagramas pertenecientes a C4 Model:

Diagrama de Contexto

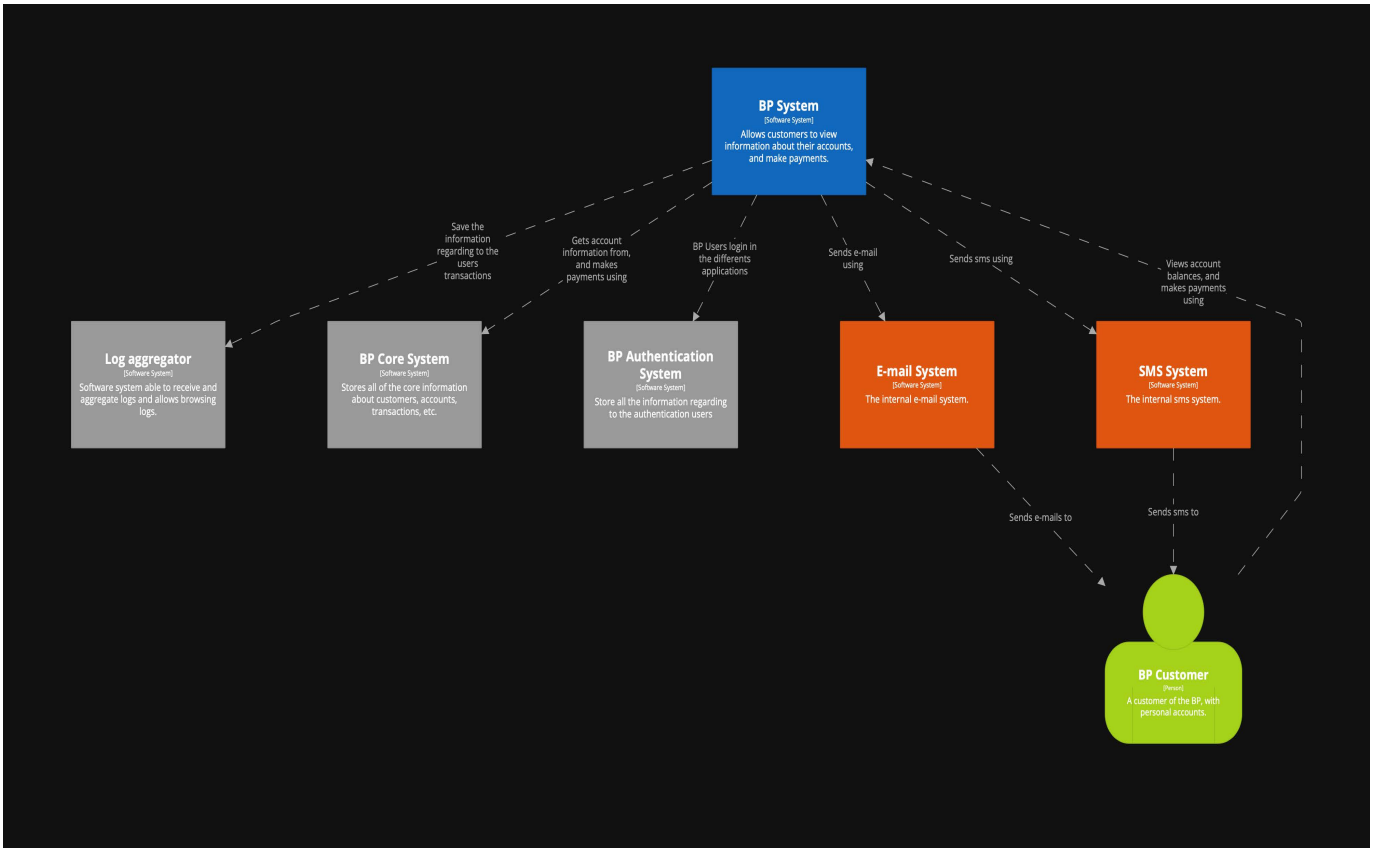


Diagrama de Contenedores

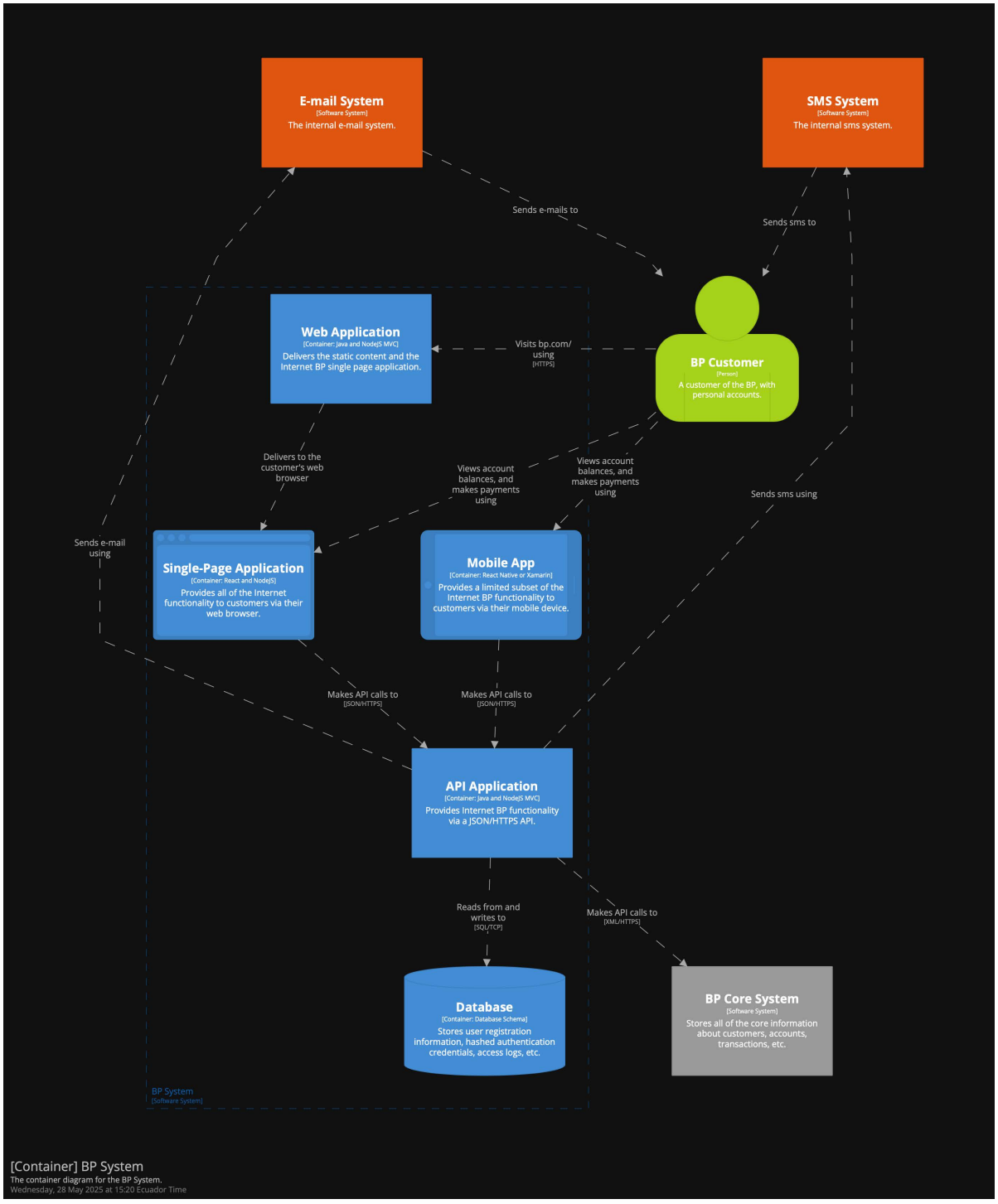
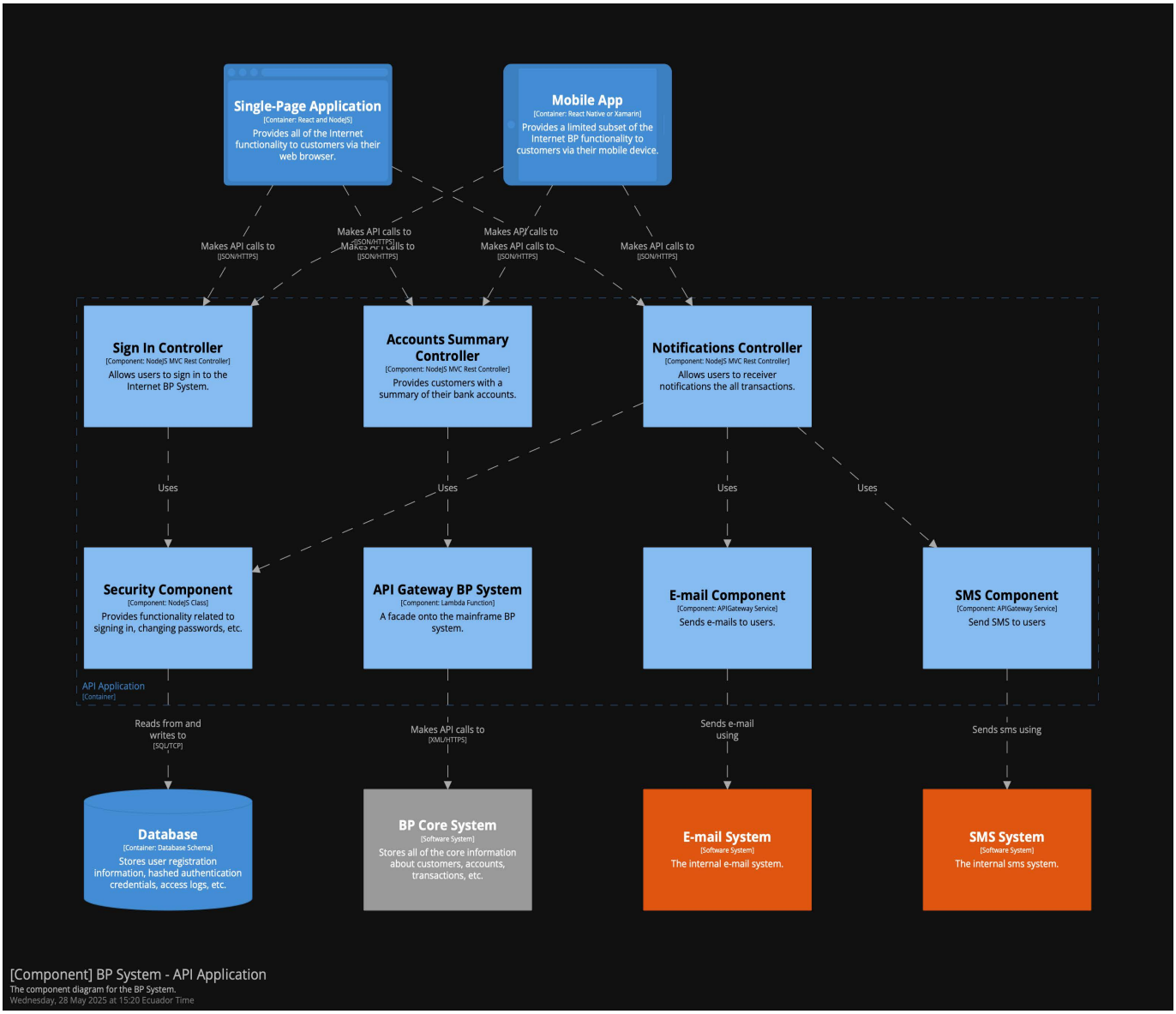
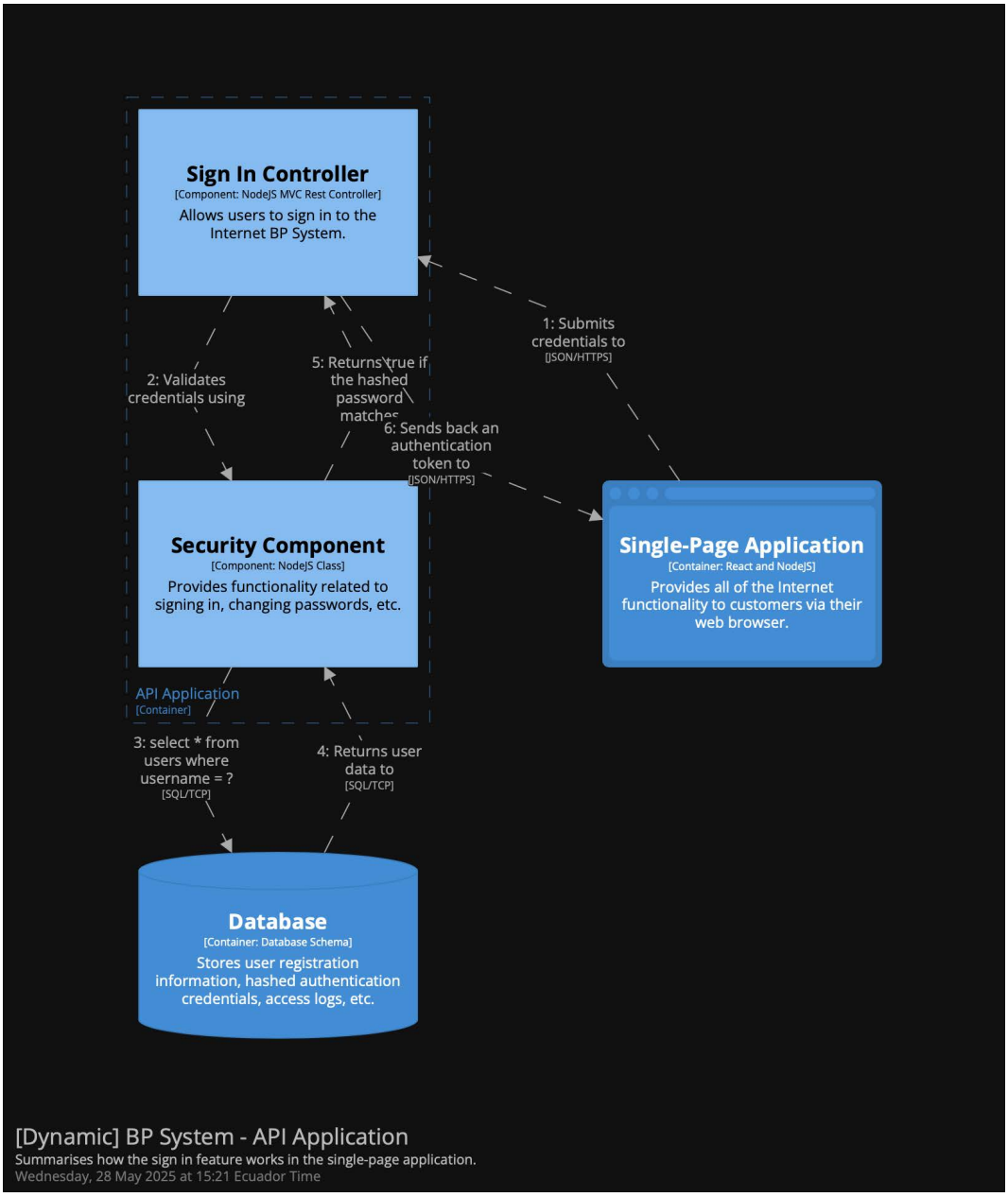


Diagrama de Componentes



SignIn Controller



Design Patterns for System

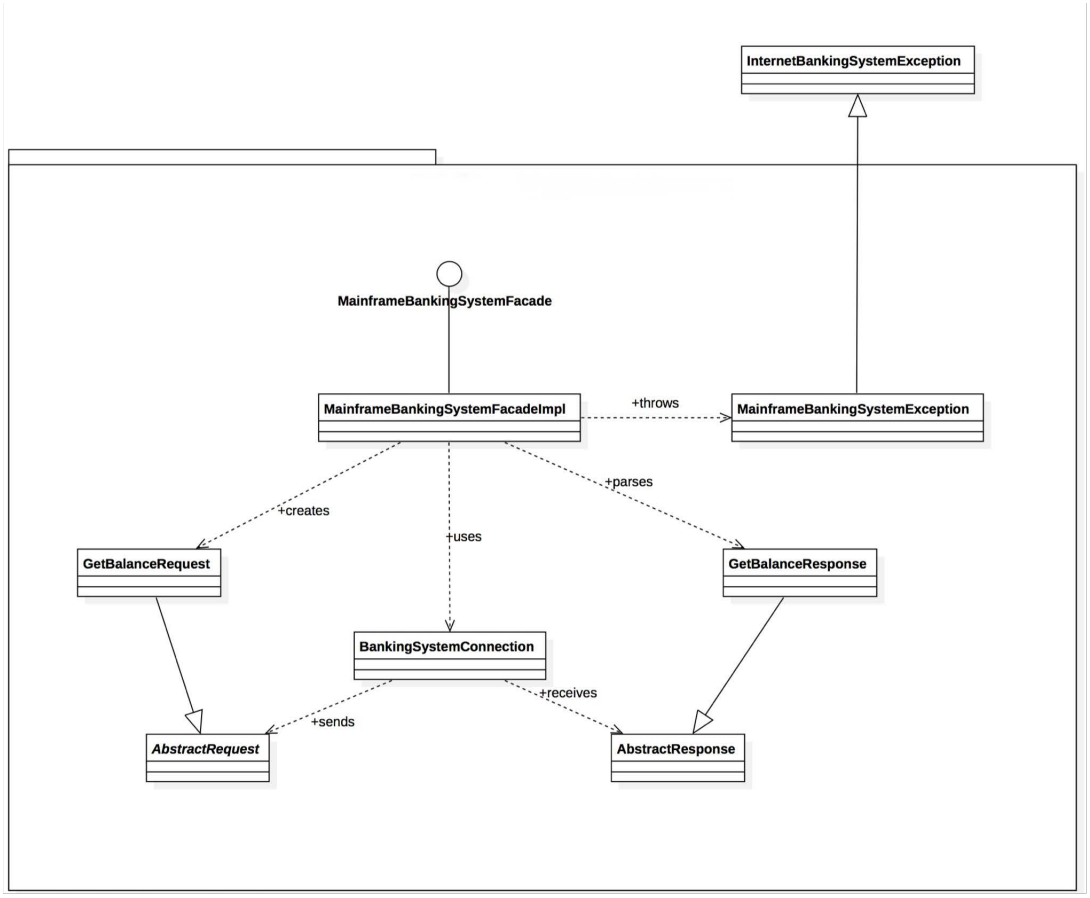
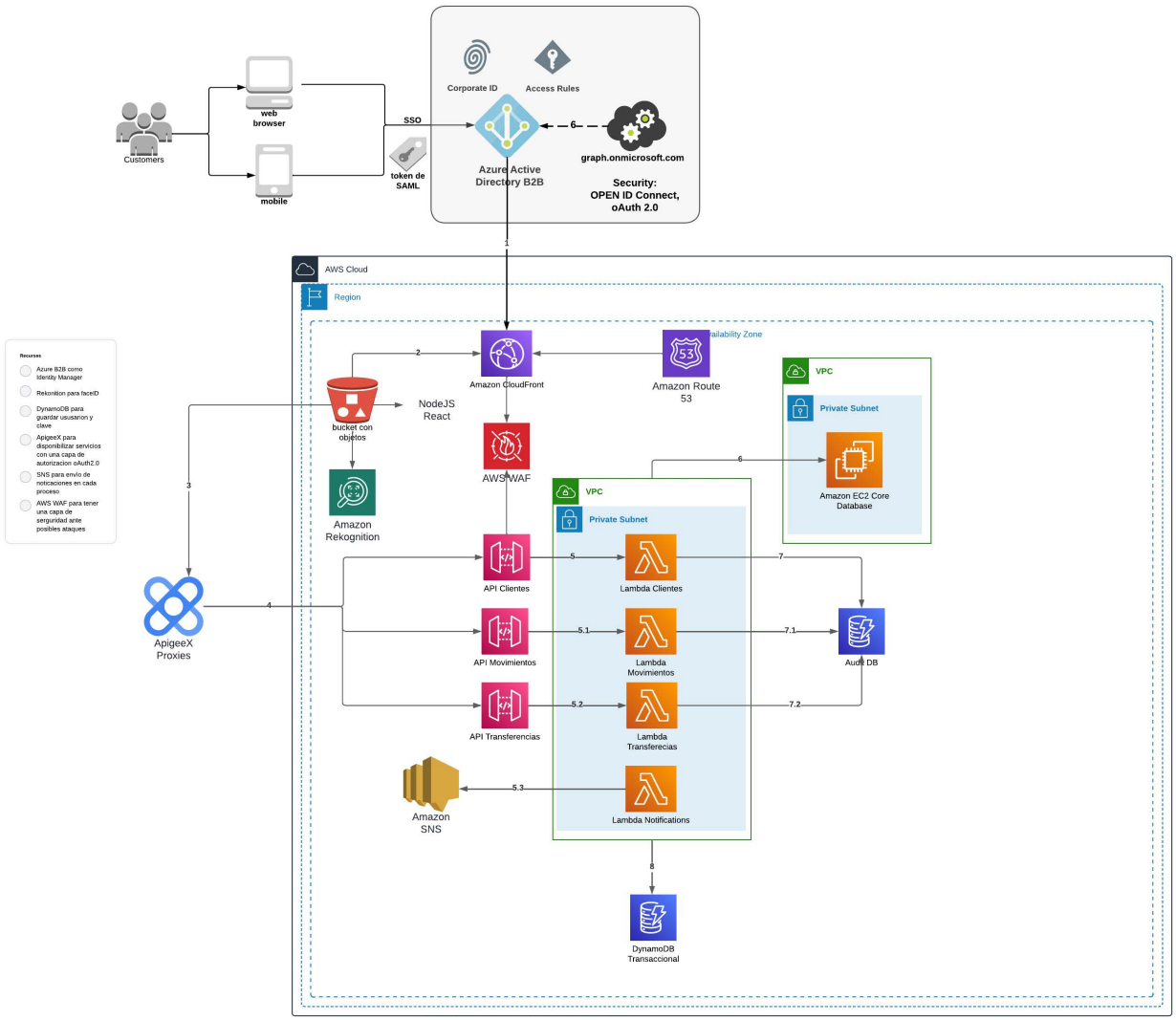


Diagrama de Arquitectura AWS



Consideraciones:

- El diseño de la arquitectura esta basada en Arquitectura serverless lo cual permite tener alta disponibilidad de los servicios nativos de las nubes públicas que estamos usando para el presente reto técnico.
- Hemos usado nubes públicas tanto de AWS como Azure, la nube principal donde se despliegan los recursos tanto de infraestructura, cómputo, storage es la nube de Amazon Web Services, adicionalmente hemos usado otra nube pública para manejo de identidades de usuario esta es Microsoft Azure.
- La arquitectura implementada es una arquitectura totalmente desacoplada por lo cual no necesitamos darle mantenimiento y a su vez se puede incluir varios servicios nativos de la nube principal.

- El flujo de autenticación es a través de los servicios que nos provee la nube pública de Microsoft Azure lo cual se puede federar las identidades a través de un componente de AWS que se llama AWS Cognito.
- AWS Cognito también nos permite darle una capa de seguridad adicional a nuestros APIS que exponemos para el consumo de nuestros servicios a través de un token de tipo bearer (oAuth 2.0).
- Los patrones de diseño que hemos implementado en esta arquitectura son API First, Domain Design Driven (DDD).
- La seguridad de esta arquitectura planteada está basada en el OWASP Top 10, para dicho fin se ha incorporado dentro de la arquitectura un recurso de seguridad que se llama AWS WAF lo cual permite tener una capa de protección adicional tanto para Aplicaciones WEB como para APIS de AWS API Gateway.
- El monitoreo es constante a través del servicio de AWS CloudWatch adicionalmente se puede incorporar correlacionadores de eventos como Splunk para poder tener una mayor visión de todas las transacciones.
- Para el envío de mensajes se ha utilizado un recurso de la nube pública de Amazon que se llama AWS Simple Notification Service (AWS SNS), en el cual se puede parametrizar las diferentes cuotas para envíos de correos electrónicos y SMS.
- Para el manejo de los mensajes y que estos nunca se pierdan hemos diseñado dentro de la arquitectura un servicio de AWS SQS (Amazon Simple Queue Service), El cual por defecto nos permite generar colas de tipo FIFO para que los mensajes no se pierdan y puedan haber reintentos en los envios de las notificaciones.
- Dentro de la solución propuesta hemos incluido un componentes de base de datos NO/SQL que es DynamoDB, en esta base de datos no relacional se va a guardar todo todas las plantillas correspondientes para el envío de correo electrónico y así evitar los fallos. En esta fuente de datos se va a guardar todas las plantillas asociándoles con un identificador único para que se pueda consumir desde un micro servicio.
- Al ser una Arquitectura Serverless por defecto las nubes públicas nos dan alta disponibilidad de todos los servicios nativos por lo cual no debemos preocuparnos porque un servicio vaya a estar indisponible, pero como alternativa se puede desplegar en otra región.
- Para recolectar los datos del usuario se ha diseñado una arquitectura la cual nos permite obtener esta información a través de una aplicación web o aplicaciones nativas para aplicativos móviles esta información Se va a guardar en un repositorio y a su vez se va a replicar esta información a través de Microsoft Graph Services para alojarla en el Active Directory de Azure B2C.
- Para el diseño de las aplicaciones nativas móviles se puede utilizar Microsoft Xamarin o React Native.