**Total Report**

Abril de 2023

**APROBACIÓN DEL DOCUMENTO**

**CONTROL DE VERSIONES**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Versión** | **Fecha** | **Autor** | **Revisor** | **Descripción el cambio** |
| 2.0 | 28/04/23 | Juan David Velásquez Bedoya |  | Emisión |
| 2.1 | 04/05/23 | Ricardo López |  | Se agregaron preguntas enviadas por Jhon William |
| 2.2 | 25/05/23 | Ricardo López |  | Se agregan comentarios de ultimos correos |

**CONTENIDO**

1. **CONTEXTO DE NEGOCIO** [**4**](https://docs.google.com/document/d/1e1KhNqgGMo4ynPb3EHM9b__FGVaAATqV/edit#heading=h.gjdgxs)
   1. Propósito del sistema [4](https://docs.google.com/document/d/1e1KhNqgGMo4ynPb3EHM9b__FGVaAATqV/edit#heading=h.30j0zll)
2. Definiciones [4](https://docs.google.com/document/d/1e1KhNqgGMo4ynPb3EHM9b__FGVaAATqV/edit#heading=h.1fob9te)
3. Objetivos de Negocio [5](https://docs.google.com/document/d/1e1KhNqgGMo4ynPb3EHM9b__FGVaAATqV/edit#heading=h.3znysh7)
4. Identificación de Stakeholders [5](https://docs.google.com/document/d/1e1KhNqgGMo4ynPb3EHM9b__FGVaAATqV/edit#heading=h.2et92p0)
5. Requerimientos significativos de arquitectura [6](https://docs.google.com/document/d/1e1KhNqgGMo4ynPb3EHM9b__FGVaAATqV/edit#heading=h.tyjcwt)
6. Restricciones del sistema [6](https://docs.google.com/document/d/1e1KhNqgGMo4ynPb3EHM9b__FGVaAATqV/edit#heading=h.3dy6vkm)
7. **ANÁLISIS DE ESCENARIOS DE ATRIBUTOS DE CALIDAD** [**8**](https://docs.google.com/document/d/1e1KhNqgGMo4ynPb3EHM9b__FGVaAATqV/edit#heading=h.1t3h5sf)
   1. Razonamiento [8](https://docs.google.com/document/d/1e1KhNqgGMo4ynPb3EHM9b__FGVaAATqV/edit#heading=h.4d34og8)
   2. Descripción de escenarios [9](https://docs.google.com/document/d/1e1KhNqgGMo4ynPb3EHM9b__FGVaAATqV/edit#heading=h.2s8eyo1)
   3. Tácticas de resolución de arquitectura [11](https://docs.google.com/document/d/1e1KhNqgGMo4ynPb3EHM9b__FGVaAATqV/edit#heading=h.17dp8vu)
8. **DECISIONES DE ARQUITECTURA** [**14**](https://docs.google.com/document/d/1e1KhNqgGMo4ynPb3EHM9b__FGVaAATqV/edit#heading=h.3rdcrjn)
   1. División en Microservicios con Brokers [14](https://docs.google.com/document/d/1e1KhNqgGMo4ynPb3EHM9b__FGVaAATqV/edit#heading=h.26in1rg)

3.2 Contenedores y funciones Serverless [15](https://docs.google.com/document/d/1e1KhNqgGMo4ynPb3EHM9b__FGVaAATqV/edit#heading=h.lnxbz9)

1. **VISTAS DE ARQUITECTURA** [**17**](https://docs.google.com/document/d/1e1KhNqgGMo4ynPb3EHM9b__FGVaAATqV/edit#heading=h.35nkun2)
   1. Vista de Contexto [17](https://docs.google.com/document/d/1e1KhNqgGMo4ynPb3EHM9b__FGVaAATqV/edit#heading=h.1ksv4uv)
   2. Vista general de la solución [18](https://docs.google.com/document/d/1e1KhNqgGMo4ynPb3EHM9b__FGVaAATqV/edit#heading=h.44sinio)
   3. Vista Funcional [19](https://docs.google.com/document/d/1e1KhNqgGMo4ynPb3EHM9b__FGVaAATqV/edit#heading=h.2jxsxqh)
      1. Blueprint de servicios [19](https://docs.google.com/document/d/1e1KhNqgGMo4ynPb3EHM9b__FGVaAATqV/edit#heading=h.z337ya)
      2. Flujo funcional general de la generación de reportes por los usuarios [20](https://docs.google.com/document/d/1e1KhNqgGMo4ynPb3EHM9b__FGVaAATqV/edit#heading=h.3j2qqm3)
      3. Flujo funcional general de configuración de documentos por administradores [20](https://docs.google.com/document/d/1e1KhNqgGMo4ynPb3EHM9b__FGVaAATqV/edit#heading=h.1y810tw)
   4. Vista de Desarrollo [21](https://docs.google.com/document/d/1e1KhNqgGMo4ynPb3EHM9b__FGVaAATqV/edit#heading=h.4i7ojhp)
      1. Estructura estática [22](https://docs.google.com/document/d/1e1KhNqgGMo4ynPb3EHM9b__FGVaAATqV/edit#heading=h.2xcytpi)
         1. Estructura de Microservicios base [22](https://docs.google.com/document/d/1e1KhNqgGMo4ynPb3EHM9b__FGVaAATqV/edit#heading=h.1ci93xb)
2. Estructura de las aplicaciones Front-end [23](https://docs.google.com/document/d/1e1KhNqgGMo4ynPb3EHM9b__FGVaAATqV/edit#heading=h.3whwml4)
3. Estructura de las aplicaciones del Back-end [23](https://docs.google.com/document/d/1e1KhNqgGMo4ynPb3EHM9b__FGVaAATqV/edit#heading=h.2bn6wsx)
4. Definición del tablero de control en el Home de Total Report [24](https://docs.google.com/document/d/1e1KhNqgGMo4ynPb3EHM9b__FGVaAATqV/edit#heading=h.qsh70q)
5. Definición dinámica del sistema [25](https://docs.google.com/document/d/1e1KhNqgGMo4ynPb3EHM9b__FGVaAATqV/edit#heading=h.3as4poj)
6. Detalle del flujo de configuración y carga de fuentes realizada por un usuario [25](https://docs.google.com/document/d/1e1KhNqgGMo4ynPb3EHM9b__FGVaAATqV/edit#heading=h.1pxezwc)
7. Detalle del proceso de formulación con cálculo interactivo y en lote [27](https://docs.google.com/document/d/1e1KhNqgGMo4ynPb3EHM9b__FGVaAATqV/edit#heading=h.49x2ik5)
8. Detalle del proceso de validación de cálculos [28](https://docs.google.com/document/d/1e1KhNqgGMo4ynPb3EHM9b__FGVaAATqV/edit#heading=h.2p2csry)
9. Detalle del proceso de transmisión [28](https://docs.google.com/document/d/1e1KhNqgGMo4ynPb3EHM9b__FGVaAATqV/edit#heading=h.147n2zr)

Para la definición de la anterior solución se consideró la restricción RST-SIS-07 llegando a la conclusión de que no era conveniente transicionarla por los siguientes motivos: [29](https://docs.google.com/document/d/1e1KhNqgGMo4ynPb3EHM9b__FGVaAATqV/edit#heading=h.3o7alnk)

1. Detalle del proceso de definición de reglas en documento técnico y formato [29](https://docs.google.com/document/d/1e1KhNqgGMo4ynPb3EHM9b__FGVaAATqV/edit#heading=h.23ckvvd)
2. Definición del proceso de generación de reportes por fecha [30](https://docs.google.com/document/d/1e1KhNqgGMo4ynPb3EHM9b__FGVaAATqV/edit#heading=h.ihv636)
3. Concurrencia y estrategia de escalamiento [31](https://docs.google.com/document/d/1e1KhNqgGMo4ynPb3EHM9b__FGVaAATqV/edit#heading=h.32hioqz)
4. Vista de Información [33](https://docs.google.com/document/d/1e1KhNqgGMo4ynPb3EHM9b__FGVaAATqV/edit#heading=h.1hmsyys)
5. Vista de Despliegue [35](https://docs.google.com/document/d/1e1KhNqgGMo4ynPb3EHM9b__FGVaAATqV/edit#heading=h.41mghml)
   1. Infraestructura de la etapa 1: Migración en su estado actual. [35](https://docs.google.com/document/d/1e1KhNqgGMo4ynPb3EHM9b__FGVaAATqV/edit#heading=h.2grqrue)
   2. Infraestructura de la etapa 2: migración en etapa intermedia [36](https://docs.google.com/document/d/1e1KhNqgGMo4ynPb3EHM9b__FGVaAATqV/edit#heading=h.vx1227)
   3. Infraestructura para la etapa 3 y selección tecnológicasd [37](https://docs.google.com/document/d/1e1KhNqgGMo4ynPb3EHM9b__FGVaAATqV/edit#heading=h.3fwokq0)
   4. Proveedor Cloud y compatibilidad con otros proveedores [37](https://docs.google.com/document/d/1e1KhNqgGMo4ynPb3EHM9b__FGVaAATqV/edit#heading=h.1v1yuxt)
   5. Modelo de responsabilidad y gestión de recursos Cloud [38](https://docs.google.com/document/d/1e1KhNqgGMo4ynPb3EHM9b__FGVaAATqV/edit#heading=h.4f1mdlm)
6. Vista de Integración y Seguridad [39](https://docs.google.com/document/d/1e1KhNqgGMo4ynPb3EHM9b__FGVaAATqV/edit#heading=h.2u6wntf)
7. **MODELO DE ENTREGA** [**42**](https://docs.google.com/document/d/1e1KhNqgGMo4ynPb3EHM9b__FGVaAATqV/edit#heading=h.19c6y18)
   1. Estrategia de migración [42](https://docs.google.com/document/d/1e1KhNqgGMo4ynPb3EHM9b__FGVaAATqV/edit#heading=h.3tbugp1)
   2. Pasos para la ejecución de la migración [42](https://docs.google.com/document/d/1e1KhNqgGMo4ynPb3EHM9b__FGVaAATqV/edit#heading=h.28h4qwu)
   3. Plan de acción [44](https://docs.google.com/document/d/1e1KhNqgGMo4ynPb3EHM9b__FGVaAATqV/edit#heading=h.nmf14n)
   4. Pasos para la migración de Total Tax [47](https://docs.google.com/document/d/1e1KhNqgGMo4ynPb3EHM9b__FGVaAATqV/edit#heading=h.37m2jsg)
8. **PRUEBAS DE CONCEPTO Y ANEXOS** [**49**](https://docs.google.com/document/d/1e1KhNqgGMo4ynPb3EHM9b__FGVaAATqV/edit#heading=h.1mrcu09)
   1. Prueba 1. Verificación de CodeCatalyst como solución DevOps [49](https://docs.google.com/document/d/1e1KhNqgGMo4ynPb3EHM9b__FGVaAATqV/edit#heading=h.46r0co2)
   2. Prueba 2. Zenduty como solución de llamadas de soporte. [51](https://docs.google.com/document/d/1e1KhNqgGMo4ynPb3EHM9b__FGVaAATqV/edit#heading=h.2lwamvv)
   3. ANEXO 1. Taller de Atributos de Calidad [52](https://docs.google.com/document/d/1e1KhNqgGMo4ynPb3EHM9b__FGVaAATqV/edit#heading=h.111kx3o)
   4. ANEXO 1. Tecnologías y tópicos relevantes [53](https://docs.google.com/document/d/1e1KhNqgGMo4ynPb3EHM9b__FGVaAATqV/edit#heading=h.3l18frh)
   5. ANEXO 1. Estimación aproximada del desempeño [54](https://docs.google.com/document/d/1e1KhNqgGMo4ynPb3EHM9b__FGVaAATqV/edit#heading=h.206ipza)

1. **CONTEXTO DE NEGOCIO**
   1. **Propósito del sistema**

Tótal Report es una aplicación diseñada para solucionar las necesidades de generación de reportes regulatorios colombianos.

Mediante Tótal Report las empresas cuentan con una herramienta para gestionar la información a reportar y los reportes generados, actualmente este aplicativo puede operar on-premise o en la nube, facilitando el control y seguridad de la información, especialmente en la modalidad on-premise.

1. **Definiciones**

|  |  |
| --- | --- |
| **Término** | **Definición** |
| **Formato** | Documento regulatorio exigido por alguna autoridad de control a entidades vigiladas y que se rige por una norma técnica. |
| **Migración** | Transferencia de uno o varios reportes generados por Tótal Report al ente de control. |
| **Fuentes** | Información en formato digital cuyo contenido tiene el propósito de generar el reporte. |
| **Reporte** | Documento final generado por la plataforma y basado en las fuentes de información del usuario. |
| **- TotalReport.Worker**  **- Servicio.Migración**  **- Servicio.Fuentes** | Aplicación interna tipo consola, creada para trabajar en paralelo en el procesamiento de formatos, se pueden lanzar desde capas superiores o inferiores. |
| **TotalReport.WebApp** |  |
| **Conglomerados.API** | Servicio web |
| **TotalReport.AgenteTransmisión** | Componente encargado de transmitir a entes de control, actualmente derogado. |
| **TotalReport.XBRL** | Componente a cargo de la generación de XBRL |
|  |  |

1. **Objetivos de Negocio**

|  |  |
| --- | --- |
| **ID** | **Descripción del objetivo de negocio** |
| **OBJ-NEG-01** | Cumplir con las expectativas técnicas de capacidad de carga de trabajo al migrar a la nube, derivados de la relación comercial con la  empresa Nequi. |
| **OBJ-NEG-02** | Expandir la capacidad operativa de Tótal Report para trabajar con grandes volúmenes de información simultáneamente. |
| **OBJ-NEG-03** | Mejorar la usabilidad de la aplicación. |
| **OBJ-NEG-04** | Mejorar la seguridad de la aplicación para migrar con tranquilidad a la nube. |
| **OBJ-NEG-05** | Modificar la aplicación para que facilite la reutilización de capacidades al incorporar nuevos reportes. |

1. **Identificación de *Stakeholders***

|  |  |
| --- | --- |
| **Stakeholder** | **Intereses** |
| **John William Parra** | Director ejecutivo corporativo, desea consolidar negociaciones con Nequi y mejorar la plataforma preparándola para clientes similares. |
| **Andrey Ramirez** | Transformar o actualizar la plataforma a fin de eliminar los cuellos de botella identificados en el procesamiento de grandes cantidades de información. |
| **Ricardo López** | Líder de fábrica Project House, desea actualizar la plataforma para el manejo de grandes cantidades de información y recibir apoyo en el desarrollo derivado de las propuestas. |
| **John Edier Vargas** | Líder de implementaciones en cliente, su interés se centra en mejorar la competitividad de la solución a fin de garantizar la continuidad y crecimiento de Total Report comercialmente. |
| **Darwin Terraza** | Líder técnico de Total Report en Project House, interesado en la mejora técnica para facilitar el desarrollo comercial. |
| **Daniela Jimenez** | Líder de equipo e implementación. |

1. **Requerimientos significativos de arquitectura**

|  |  |
| --- | --- |
| **ID** | **Descripción de la restricción** |
| **TSR-SIS-01** | La nueva solución deberá basarse en la mayor medida posible en la aplicación actual construida con tecnologías Microsoft y buscar darle continuidad, evitando tener que iniciar desde cero. |
| **TSR-SIS-02** | La nueva plataforma deberá operar en la nube de AWS. |
| **TSR-SIS-03** | Se requiere poder manejar archivos fuentes con hasta 10.000 millones de registros. |
| **TSR-SIS-04** | Se espera rediseñar la aplicación para mejorar la usabilidad |
| **TSR-SIS-05** | Se debe utilizar tecnologías de comunicación seguras para garantizar la protección de la información. |
| **TSR-SIS-06** | Se espera contar con la flexibilidad y libertad técnica como la que ofrece la arquitectura de microservicios. |
| **TSR-SIS-07** | La operación normal de la plataforma no se debe ver afectada negativamente por la modernización de la misma. |
| **TSR-SIS-08** | Existe una vertiente de la aplicación llamada Total Tax para reportes a la DIAN la cual debe ser integradada en la solución |

1. **Restricciones del sistema**

|  |  |
| --- | --- |
| **ID** | **Descripción de la restricción** |
| **RST-SIS-01** | Componentes legados dificultan la actualización de la versión del framework utilizado el cual es .NET Framework MVC 4.0 con Razor v2.0, entre estos se encuentran:   Infragistics: Utilizado en un 40% del front.  JQuery v.1.8:  SpreadJS v.3: Utilizado en un 90% del front y depende de JQuery v1.8  DevExpress v.20.1: Utilizado en un 30%, utilizado para exportar PDFs gracias a su conexión con formatos XBRL. |
| **RST-SIS-02** | El sistema tiene dependencia de SQL Integration Services con múltiples ETL necesarias para procesar fuentes grandes y procesarlas, esto dificulta el cambio de base de datos. |
| **RST-SIS-03** | El sistema utiliza servicios REST, SOAT, gRPC o GraphQL de forma general y homogénea para la comunicación entre componentes, por ahora solo la representación o renderización para el front-end y los llamados locales puesto qué trabaja como una única unidad de ejecución. |
| **RST-SIS-04** | El sistema no tiene un módulo administrativo independiente para gestionar usuarios, roles, permisos y parametrizaciones, en su lugar la misma vista es alterada para mostrar menús adicionales en caso de que el rol del usuario sea administrador. |
| **RST-SIS-05** | El componente SpreadJS de GrapeCity requiere ser actualizado o cambiado para pasar a la nube, la licencia necesaria es una licencia de desarrollo y despliegue a perpetuidad, dicho valor puede superar los 12.000 USD, se recomienda la alternativa OpenSource JSpreadSheet CE de https://github.com/jspreadsheet/ce |
| **RST-SIS-06** | La carga de información en el sistema de almacenamiento de la aplicación (bases de datos y sistemas de archivos) dependerán de la velocidad de lectura y entrega de información de los sistemas de los clientes, es decir, que la velocidad máxima alcanzable para Total Report en carga de fuentes es la velocidad máxima de la fuente de datos localizada en los sistemas computacionales y de red de los usuarios. |
| **RST-SIS-07** | La aplicación cuenta con un módulo especializado en transmisiones o consolidación de reportes formulados en archivo plano, el cual fue desarrollado como aplicación aparte en lenguaje Python y no es estrictamente necesario para el funcionamiento habitual de la aplicación, se ha solicitado estudiar su arquitectura y validar su transición y mejor integración en la presente solución. |

1. **ANÁLISIS DE ESCENARIOS DE ATRIBUTOS DE CALIDAD**
   1. **Razonamiento**

|  |  |  |
| --- | --- | --- |
| **Atributo** | **Razonamiento** | **Prioridad** |
| **Desempeño** | La aplicación requiere leer y procesar millones de registros en tiempos razonables, dado que actualmente le toma 1 hora para fuentes de datos de con 23 millones de registros, los nuevos clientes de la compañía requieren mayor volumen de procesamiento. | 1 |
| **Modificabilidad** | Actualmente la aplicación tiene distintos formatos para la generación de reportes regulatorios, sin embargo, no siempre coinciden con los que los clientes necesitan, para lograr esta coincidencia se puede requerir modificar el software, por lo anterior, surge la necesidad de mejorar la modificabilidad de la aplicación de forma óptima. | 2 |
| **Usabilidad** | La aplicación no presenta un flujo simple y claro para ejecutarse o una guía que permita a los usuarios llevar a cabo la generación de reportes, por este motivo el cliente ha evidenciado la necesidad de mejorar la usabilidad general. | 3 |
| **Seguridad** | Considerando la sensibilidad de la información que maneja la aplicación, así como los requerimientos regulatorios de manejo de cifrado en las comunicaciones, la seguridad se destaca dentro de los atributos. | 4 |
| **Escalabilidad** | La falta de escalabilidad es el primer aspecto por el cual la plataforma no es apta para gestionar grandes cargas de trabajo, al solo ejecutarse actualmente en un solo conjunto de servidores, su capacidad está limitada a la potencia de hardware de cualquiera de sus servidores. | 5 |

1. **Descripción de escenarios**

|  |  |
| --- | --- |
| **Identificador** | **ESC\_001** |
| **Atributo de calidad relevante** | Desempeño |
| **Objetivo de negocio correspondiente** | OBJ-NEG-02 |
| **Fuente** | Usuario |
| **Estímulo** | Cargar más de 100 millones de registros de una fuente local. |
| **Artefacto** | Aplicación web |
| **Ambiente** | Alta carga de trabajo. |
| **Respuesta** | Carga de fuente exitosa. |
| **Medida de la respuesta** | 5 minutos o 333.334 registros por segundo por servidor o instancia. |

|  |  |
| --- | --- |
| **Identificador** | **ESC\_002** |
| **Atributo de calidad relevante** | Modificabilidad (Especificación: reusabilidad) |
| **Objetivo de negocio correspondiente** | OBJ-NEG-01, OBJ-NEG-05 |
| **Fuente** | Administrador de plataforma |
| **Estímulo** | Habilitar un nuevo tipo de reporte para los clientes en la aplicación con el menor esfuerzo de desarrollo. |
| **Artefacto** | Aplicación web |
| **Ambiente** | Operación normal |
| **Respuesta** | Configuración completada |
| **Medida de la respuesta** | Reducción de etapas de desarrollo de código actualmente necesarias, convertidas a horas por desarrollador. |

|  |  |
| --- | --- |
| **Identificador** | **ESC\_003** |
| **Atributo de calidad relevante** | Usabilidad |
| **Objetivo de negocio correspondiente** | OBJ-NEG-03 |
| **Fuente** | Usuario |
| **Estímulo** | Generar uno o varios reportes en archivos planos con información preconfigurada para obtenerse de forma automática, considerando la fecha de corte o con una acción simple. |
| **Artefacto** | Aplicación web |
| **Ambiente** | Operación normal |
| **Respuesta** | Migración terminada o archivos planos generados y listos para presentación |
| **Medida de la respuesta** | Tiempo en minutos, requerido por el usuario en interacción para obtener un reporte luego de proporcionar las fuentes y configuraciones necesarias y fecha de autogeneración. |

|  |  |
| --- | --- |
| **Identificador** | **ESC\_004** |
| **Atributo de calidad relevante** | Usabilidad (Especificación: administrabilidad) |
| **Objetivo de negocio correspondiente** | OBJ-NEG-03 |
| **Fuente** | Usuario final |
| **Estímulo** | Generación de reporte |
| **Artefacto** | Aplicación web |
| **Ambiente** | Operación normal |
| **Respuesta** | Después de ingresar se debe observar un tablero de control con distintos detalles relacionados a los formatos a presentar. |
| **Medida de la respuesta** | Cantidad de detalles que requieren atención y automáticamente identificados por tipo de reporte o por cliente. |

|  |  |
| --- | --- |
| **Identificador** | **ESC\_005** |
| **Atributo de calidad relevante** | Modificabilidad |
| **Objetivo de negocio correspondiente** | OBJ-NEG-05 |
| **Fuente** | Especialista en regulación o administrador |
| **Estímulo** | Creación o gestión de validaciones |
| **Artefacto** | Aplicación web |
| **Ambiente** | Operación normal |
| **Respuesta** | Después de ingresar se debe simplificar la forma de implementar o modificar reglas de negocio aplicables a los formatos. |
| **Medida de la respuesta** | Capacidad de modificar reglas de negocio sin requerir cambios de código. |

1. **Tácticas de resolución de arquitectura**

|  |  |
| --- | --- |
| Nombre del estilo o táctica | Administración de recursos (Especificación: Incremento de recursos) |
| Escenarios Analizados | ESC\_001 |
| Estilos, tácticas y raciocinio | Las demandas de procesamiento en los picos de carga de trabajo son extremadamente altos, este uso no es continuo en la aplicación sino esporádico, es decir, en función de la necesidad de un cliente, por lo cual no es conveniente mantener un grupo grande de servidores siempre disponibles e incurrir en altos costos, para solucionarlo, se debe proporcionar mayor capacidad de cómputo bajo demanda y de forma automática al igual decrecer en dicha capacidad cuando el pico de trabajo se supere. |
| Riesgos | La velocidad del incremento de recursos computacionales podría no ser tan rápida como la demanda.  La capacidad de la aplicación por la lógica actual impedir la gestión eficiente de la demanda. |
| Trade-off | (+) Garantiza que la aplicación tendrá capacidad computacional suficiente para cualquier volumen de información.  (-) Complejiza la gestión de usuarios, autenticación, licenciamiento a clientes, control de usuarios y soporte hasta que no se tenga una transición completa a un modelo de servicios Cloud.  (+) Se pagará únicamente por los recursos computacionales utilizados contados por horas. |

|  |  |
| --- | --- |
| Nombre del estilo o táctica | Reducción del tamaño de los módulos (Especificación: División de módulos |
| Escenarios Analizados | ESC\_002, ESC\_005 |
| Estilos, tácticas y raciocinio | Desde una vista de alto nivel se observa que la aplicación tiene grandes módulos, para alcanzar un mayor grado de reutilización, se requiere dividir los módulos para que presten capacidades más específicas enfocándose en aquellas que puedan reutilizarse y no simplemente dividirlos por su tamaño. |
| Riesgos | La complejidad derivada de la división tardía de los módulos desde el punto de vista del ciclo de vida del producto, pueden imposibilitar la óptima reutilización o la división misma.  Se incrementará la latencia en la comunicación. |
| Trade-off | (+) Se mejora la especialización de cara a la reutilización.  (+) Se favorece la modificabilidad al desacoplar funcionalidades.  (+) Favorece la migración a la Nube.  (+) Mejora las posibilidades de escalamiento.  (-) Mayor latencia en la comunicación.  (-) Mayor complejidad técnica general.  (-) Se puede generar duplicación de código para lograr la independencia de los módulos. |

|  |  |
| --- | --- |
| Nombre del estilo o táctica | Apoyo de la iniciativa del sistema (Mantenimiento del modelo del sistema) |
| Escenarios Analizados | ESC\_003 |
| Estilos, tácticas y raciocinio | Como resultado del Taller de Atributos de Calidad (QAW) se consolidó el requerimiento de mejorar la experiencia de usuario facilitando el proceso de generación de reportes.  El mantenimiento del modelo del sistema básicamente mantiene explícito lo que el sistema debería realizar al usuario a fin de completar un proceso, un ejemplo de ello es una barra de progreso en la que el usuario puede observar cuantos pasos ha completado de un proceso y cuantos le faltan. |
| Riesgos | Más complejidad, en caso de que, para informar el modelo del sistema, esto implique la modificación de un elevado número de módulos porque no se tiene un modelo sino muchos. |
| Trade-off | (+) Permite al usuario tener claridad sobre el flujo correcto y su relación con él.  (-) Agrega complejidad al sistema e implica más componentes a mantener. |

|  |  |
| --- | --- |
| Nombre del estilo o táctica | Apoyo de la iniciativa del sistema (Mantenimiento del modelo de tarea) |
| Escenarios Analizados | ESC\_004 |
| Estilos, tácticas y raciocinio | Como resultado del Taller de Atributos de Calidad (QAW) se consolidó el requerimiento de mejorar la experiencia de usuario facilitando el proceso de generación de reportes.  El mantenimiento del modelo de tarea, busca ayudar al usuario intuyendo lo que quiere hacer, al basarse en información que le brinda una idea de la acción que el usuario quiere lograr y así proporcionarle ayuda temprana. |
| Riesgos | Mayor complejidad que puede afectar la velocidad para modificar el sistema, dado que el número de formatos es muy alto, aproximadamente 250, la cantidad de ayudas que se podrían generar y la base de ideas que el sistema utilizaría asociada a cada caso es muy alta. |
| Trade-off | (+) Permite al usuario tener claridad sobre una acción concreta sin invertir mucho tiempo estudiando el uso de la aplicación.  (-) Agrega complejidad al sistema e implica más componentes a mantener. |

1. **DECISIONES DE ARQUITECTURA**

|  |  |
| --- | --- |
| **Título** | **División en servicios reutilizables bajo el patrón microservicios con apoyo de Brokers.** |
| **Estado** | Aceptado |
| **Contexto** | La aplicación presenta muy alto grado de acoplamiento al ser un desarrollo principalmente monolítico que ha crecido y generado múltiples versiones que cubren distintas necesidades, esto ha generado dificultad de mantenimiento, desempeño, susceptibilidad a errores colaterales, complejidad y poca flexibilidad al cambio.  Por otra parte, la aplicación procesa grandes volúmenes de datos y requiere gran cantidad de recursos de hardware de forma esporádica, este modelo operacional no resulta sostenible a futuro dado que seguir agregando más potencia de hardware centralizada no sería suficiente para suplir las necesidades de los clientes y mucho menos varios de ellos en simultáneo, por ello se requiere distribuir y facilitar el mejoramiento de segmentos de la aplicación así como realizar una transición a la nube pública, en este caso AWS. |
| **Opciones** | 1. Continuar con el sistema monolítico actual y trabajar con escalamiento horizontal de servidores, esta opción es la más económica sin embargo no satisface todos los objetivos he impediría algunos otros como la mantenibilidad. 2. Utilizar un patrón de arquitectura Broker, este patrón se adapta muy bien porque el principio general de Total Report corresponder con el patrón Broker, pero por sí mismo es insuficiente para facilitar la modificabilidad de la aplicación |
| **Decisión** | Se propone el patrón de diseño de microservicios en combinación con una topología Broker, considerando los atributos:   * Desempeño * Mantenibilidad con enfoque en la modificabilidad. * Usabilidad con enfoque en la administrabilidad.   Estos atributos implican un equilibrio entre aprovechamiento óptimo de recursos de cómputo y la facilidad para realizar cambios, desde este entendimiento los microservicios cuentan con la capacidad de facilitar introducir cambios o realizar mantenimientos sin afectar a todo el sistema, también permiten la especialización de sus partes para mejorar puntos críticos del proceso, crecer y decrecer en función de la demanda gestionada por un Broker y el desarrollo rápido en paralelo con varios equipos de trabajo.  Considerando estas ventajas los microservicios gestionados por un Broker en algunos puntos críticos, esta combinación cumpliría con los requerimientos. |
| **Consecuencias** | El nivel de complejidad de la aplicación actual y algunos de sus problemas podrían trasladarse a los microservicios, adicionalmente la labor de construcción inicial es mucho mayor por tratarse de la construcción de varias pequeñas aplicaciones. |

1. **División en Microservicios con Brokers**

**3.2 Contenedores y funciones Serverless**

|  |  |
| --- | --- |
| **Título** | **Contenedores y funciones Serverless** |
| **Estado** | Propuesta |
| **Contexto** | Al revisar en más detalle el funcionamiento de la aplicación y el requerimiento de mejorar el desempeño, se observa que se necesita escalamiento vertical específico, si se desea aprovechar el desarrollo existente, por ejemplo, al ejecutar formulaciones, esta capacidad si se quisiera procesar en paralelo, se tendría que reescribir totalmente, porque no está diseñada para paralelizarse ni mucho menos está estandarizada para todos los formatos, en consecuencia, un escalamiento horizontal de contenedores no tendría ningún resultado y la ejecución por parte de un solo contenedor tampoco brindaría el resultado, por ello, se proponer potenciar este tipo de funcionalidades con escalamiento propio de la nube como son las funciones Serverless. |
| **Opciones** | 1. Servidores de mayores prestaciones, esta opción podría resolver la problemática, sin embargo, se tendría una pérdida constante de recursos cuando algunos usuarios no hagan uso de toda la capacidad asignada. Esta estrategia será utilizada en una primera etapa del sistema antes de iniciar la descomposición en microservicios. 2. Escalamiento horizontal solo con contenedores o instancias de servidor, este escalamiento es bastante apropiado cuando las condiciones son de alta concurrencia y no tanto procesamiento de gran complejidad variables, es decir no todas las peticiones tienen el mismo nivel de complejidad, en consecuencia, el sistema al escalar horizontalmente no puede determinar qué capacidad asignar a cada instancia nueva creada. |
| **Decisión** | Utilización de contenedores Fargate, funciones Serverless AWS Lambda las cuales permiten una mayor flexibilidad de computo, control de costos al cobrarse por tiempo de ejecución, además compatibilidad para .NET Core 3.1 ó .NET 6, todo soportado por integraciones con la arquitectura REST. |
| **Consecuencias** | La capacidad de ejecución de código independiente implica una adaptación propia para esta forma de trabajo, el retorno a un microservicio u otra forma de ejecución no será transparente o sin esfuerzo, a pesar de que el código si se puede reutilizar, este de por si es una versión muy simplificada lo que implica un esfuerzo mucho mayor para agregarle funcionalidades antes dadas por los sistemas Serverless tales como interfaces de comunicación, resiliencia, seguridad etc.  Las versiones y lenguajes de desarrollo quedan condicionadas a las provistas por el proveedor de la nube, en este caso AWS. |

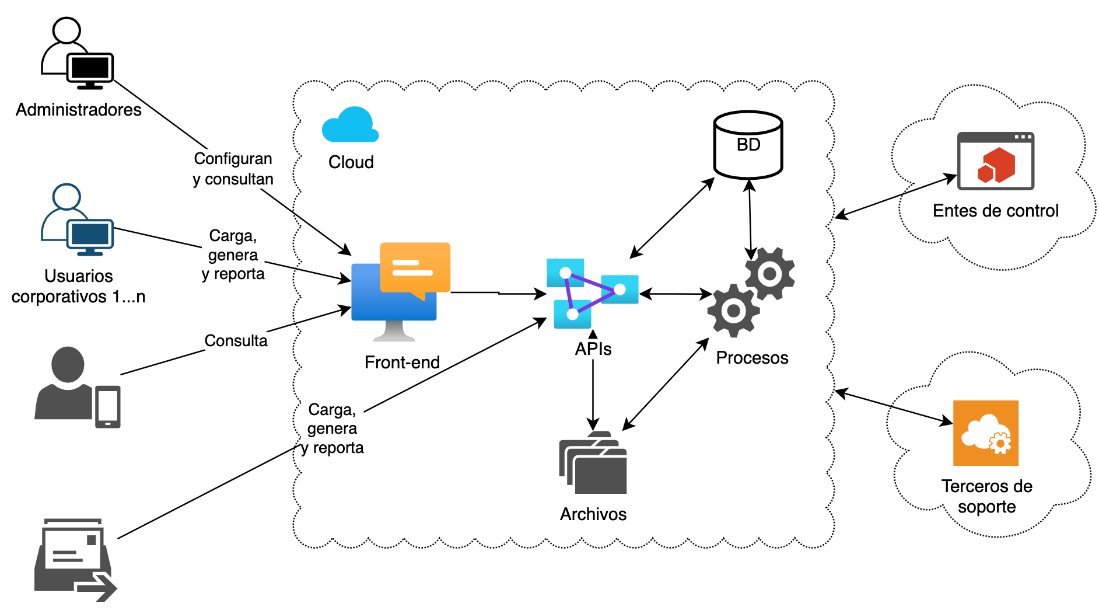
1. **VISTAS DE ARQUITECTURA**
   1. **Vista de Contexto**

El siguiente diagrama brinda una visión holística del sistema,



|  |  |
| --- | --- |
| **Sistema u Actor** | **Descripción** |
| **Sistemas de apoyo** | Los sistemas de apoyo serán aquellos los cuales agregan valor suficiente y no resultaría conveniente incorporar dentro de Total Report, a saber:   1. Monitoreo de logs de ejecución. 2. Monitoreo de logs de auditoría 3. Monitoreo de la salud de la aplicación 4. Monitoreo de hardware 5. Monitoreo para escalamiento automático 6. Reporte de alertas mediante llamadas automáticas |
| **Entes de Control** | Total Report no tendrá código específico que pueda generar acoplamiento u obligación de cambiar por cambios externos, debido a dependencia directa, estas situaciones se deberán mitigar con mediadores.  Los entes de control podrán consultar información y recibirla mediante diferentes mecanismos. |
| **Administradores** | Se contempla la capacidad de que usuarios con mayor nivel administrativo de JW Project House puedan configurar el sistema y obtener información que permita la toma decisiones. |

1. **Vista general de la solución**

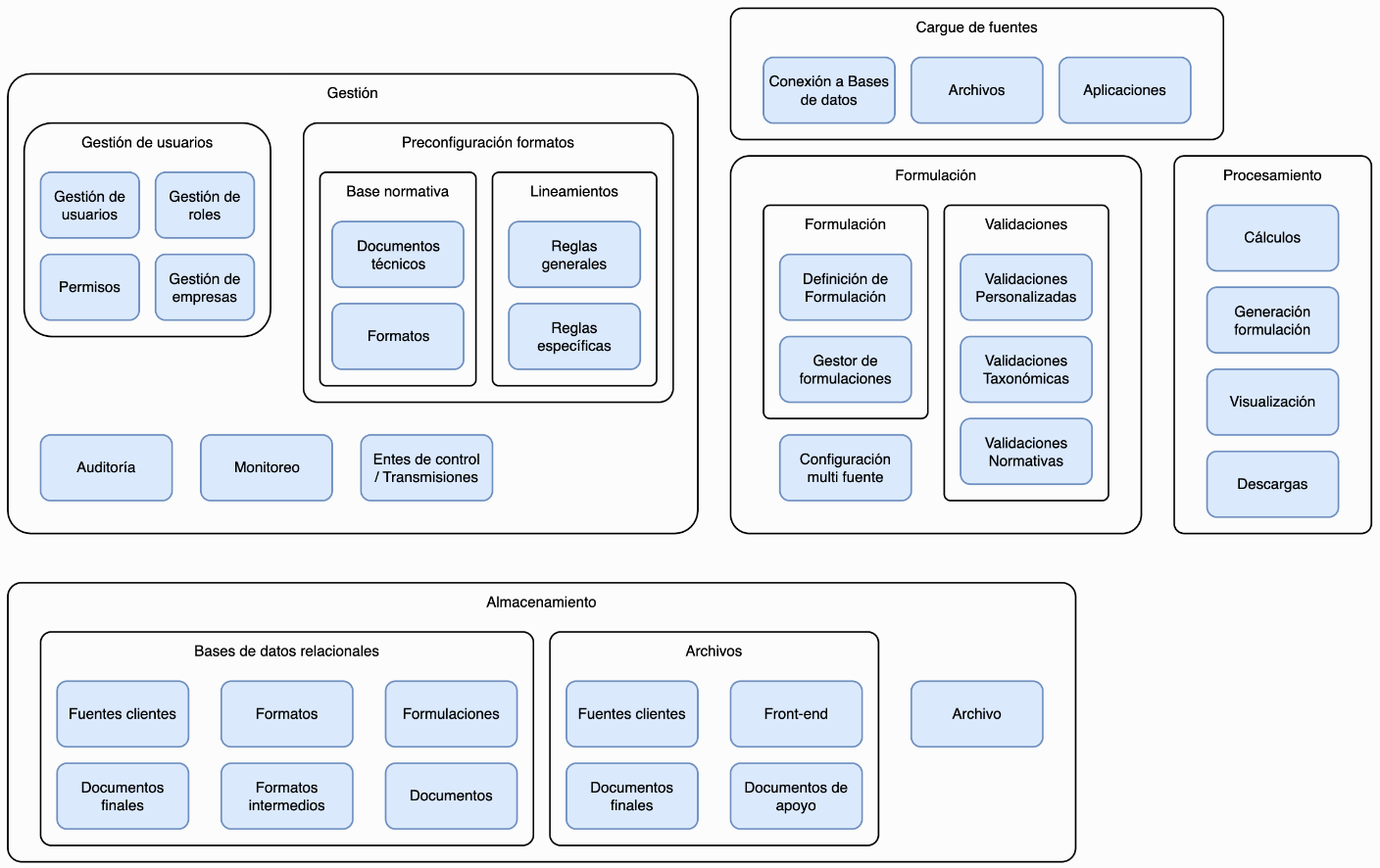


|  |  |
| --- | --- |
| **Sistema u Actor** | **Descripción** |
| **Usuarios** | Se contemplan 5 tipos de usuarios, Administradores, usuarios web, usuarios móviles, aplicaciones consumidoras de las APIs y entes de control. |
| **Front-end** | Se considera de forma general una capa visual o de presentación para la aplicación de forma aislada. |
| **APIs** | Application Programming Interface (API) son un conjunto de aplicaciones especializadas en atender y procesar solicitudes de distintas fuentes y orquestar interacciones con otros módulos. |
| **Archivos** | Representan una forma de almacenamiento simple y rápida (AWS S3) que es necesaria para distintas operaciones. |
| **Procesos** | Este componente representa a las funciones Serverless (AWS Lambda) que se utilizarán para realizar todas las tareas de procesamiento que puedan necesitarse. |
| **BD** | Representa a la base de datos como solución cloud de almacenamiento escalable tal como AWS RDS (Relational Data Base) |

1. **Vista Funcional**
   1. **Blueprint de servicios**

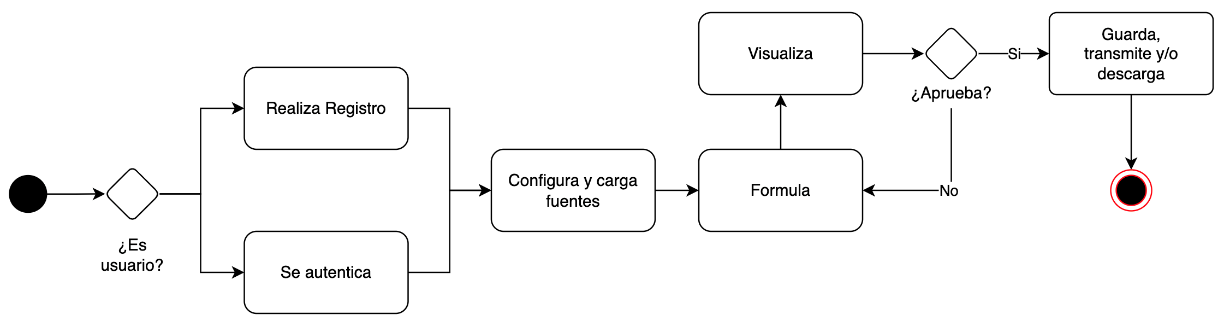
El siguiente diagrama identifica las capacidades funcionales identificadas y necesarias para el funcionamiento de la plataforma, estas capacidades o servicios son una abstracción de las capacidades técnicas que en la mayoría de los casos están representados por uno o varios módulos en la aplicación.

Igualmente, se identificaron grupos de estos como son: Gestión, cargue de fuentes, formulación, procesamiento y almacenamiento.



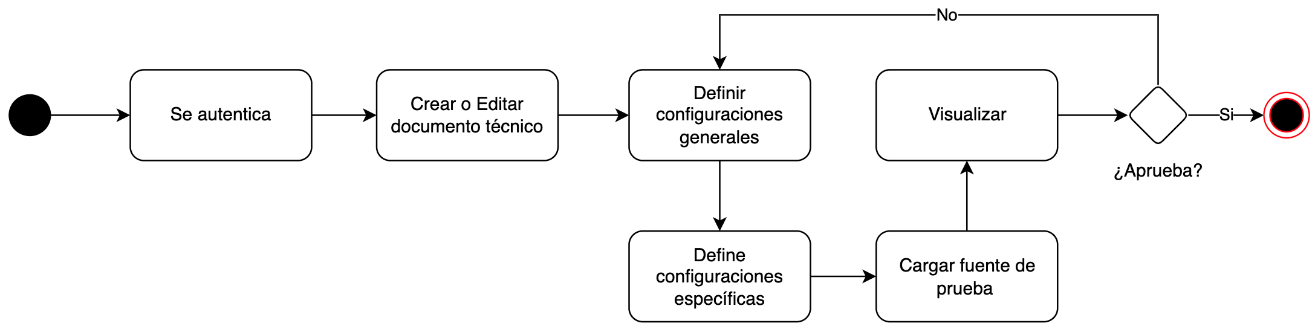
1. **Flujo funcional general de la generación de reportes por los usuarios**

El diagrama a continuación describe el flujo simplificado propuesto para que un usuario pueda generar un reporte de cualquier tipo.



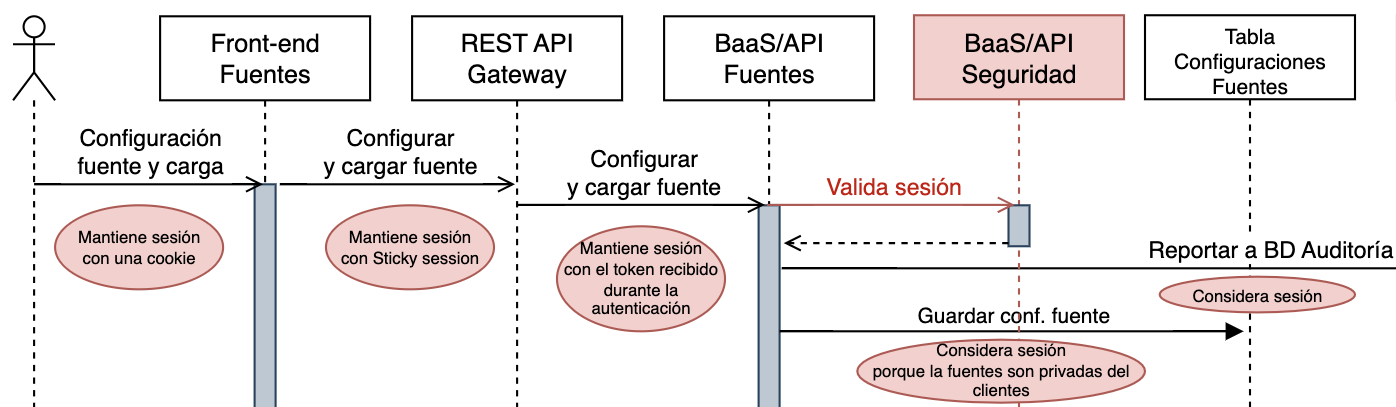
1. **Flujo funcional general de configuración de documentos por administradores**

El siguiente diagrama describe un flujo regular de configuración de reglas basadas en las normas técnias las cuales están asociadas a uno o varios tipos de reportes, estas reglas serán tenidas en cuenta por los procesos de formulación y cálculos que internamente la aplicación ejecutará para los usuarios.



1. **Vista de Desarrollo**

La presente propuesta de arquitectura está considerada para ser desarrollada con un proveedor Cloud como AWS y para brindar servicios a múltiples clientes o *multitenant* en su fase 2 y 3*,* en consecuencia los diagramas proponen estrategias almacenamiento compartidas y los componentes funcionarán para todos los clientes, de acuerdo a sus necesidades, no se estiman componentes exclusivos sino reutilizables y para este fin, podrán tener uno de dos posibles comportamientos:

1. **Stateful**: El componente necesitará verificar para qué usuario debe trabajar “multitenant” y así retornar su respuesta, este es el caso más común. Ejemplo en carga de fuentes:

1. **Stateless**: El componente no necesita identificar el usuario o empresa y solo debe llevar a cabo una tarea agnósticamente y retornar su resultado de acuerdo a las configuraciones recibidas, esta estrategia es la que facilita el escalamiento horizontal. Ejemplo en formulaciones, el proceso de cáculos es un worker que puede ser Stateless:

1. **Estructura estática**

El siguiente diagrama describe la estructura general de aplicaciones o microservicios los cuales son todos aquellos no gestionados por el proveedor de la nube, a saber: MFEs, BaaS y Procesos, todas estas aplicaciones estarán bajo tres sistemas, los MFE serán desplegados en el servicio de almacenamiento estático S3, las aplicaciones Backend as a Service (BaaS) serán en su mayoría desplegadas en contenedores y finalmente los Procesos podrán ser contendedores o funciones Serverless.

Los servicios del proveedor Cloud AWS darán soporte a la infraestructura para evitar la gestión y mantenimiento directo, en su responsabilidad más directa tendremos: Bases de datos, archivos estáticos que son el Frontend y los archivos usados para el funcionamiento normal, la administración de accesos IAM y Cognito, la gestión de acceso a las APIs, el servicio de monitoreo, la gestión de contenedores y el servicio de integración y despliegue continuo.

1. **Estructura de Microservicios base**

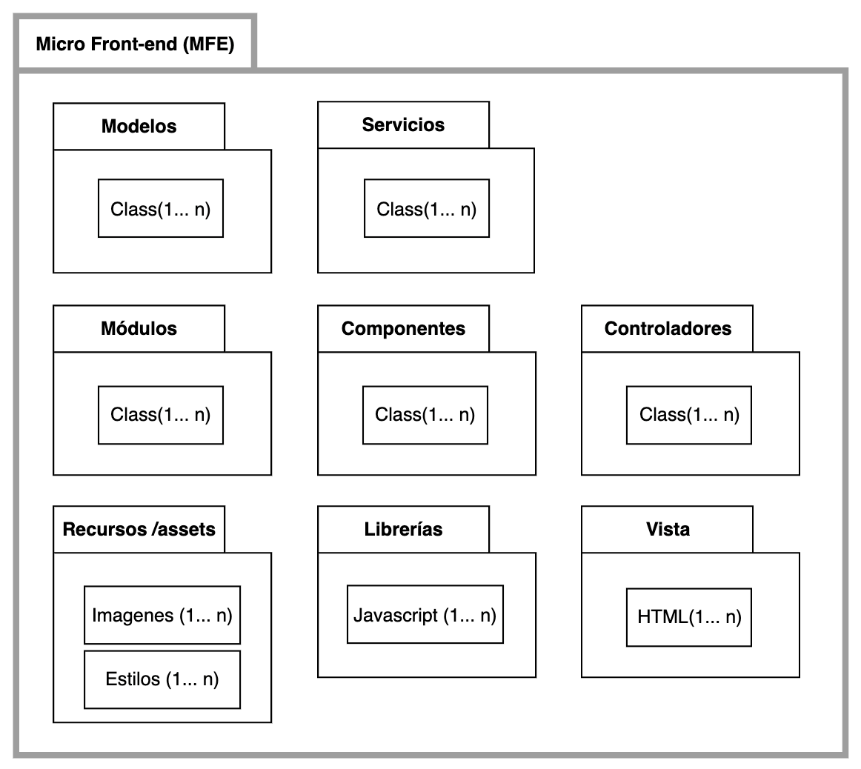
Se espera que existan múltiples casos en los cuales se requerirán desarrollos para dar soporte a

nuevos formatos, para estos casos los paquetes en color beige representan un ejemplo de cómo estos nuevos formatos serían soportados con aplicaciones en los cuatro niveles.

1. **Estructura de las aplicaciones Front-end**

La estructura propuesta para las nuevas aplicaciones Front, deberán considerar esta estructura a fin de mantener desacopladas sus partes internas y facilitar el mantenimiento, los Frameworks Front como Angular o React, facilitan esta propuesta como punto de partida para su uso, su forma básica de comunicación es mediante servicios asíncronos tipo REST.

**Diagrama de paquetes de un MFE**



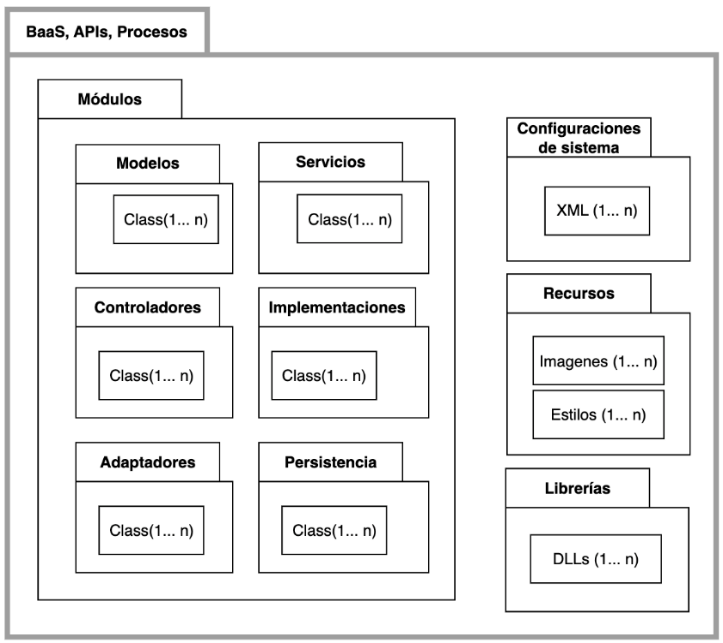
1. **Estructura de las aplicaciones del Back-end**

Al igual que la estructura propuesta para el Front-end, la figura a continuación representa una guía de estructuración del proyecto, no se establece de uso obligatorio, dado que la estructura debe brindar una facilidad de trabajo al equipo implementador, así como apoyar las buenas prácticas de escritura de código.

La estructura es aplicable a microservicios y procesos que puedan ser desplegados en contenedores.

Se debe aclarar de que la mayoría de los procesos se busca sean Serverless, sin embargo, no siempre este servicio será suficiente para satisfacer las necesidades, en esos casos la estructura del diagrama a continuación define su estructura interna.

**Diagrama de paquetes general para las aplicaciones Back-end**



1. **Definición del tablero de control en el Home de Total Report**

El tablero de control en la página principal de Total Report es un escenario de identificado durante el taller de atributos de calidad.

Existen distintas soluciones de tableros en función de las necesidades, el siguiente caso está basado en la información de la empresa que reporta, el usuario que representa a esa empresa y lo anterior en función de los reportes que debe presentar al ente de control, eso significa un tablero para el usuario normal; Para la solución se analizarán dos opciones:

La primera es mediante un sistema avanzado de reportería que facilita la conexión a la base de datos y el análisis de grandes volúmenes de información, son ejemplos de este tipo PowerBI, Tableau y AWS QuickSight, siendo este último el más adecuado para Total Report, sin embargo, este sistema tiene un costo por autor de tableros de 216 USD por año, tiene a su favor de que cada modificación al tablero se puede realizar sin realizar despliegues adicionales, ni tampoco requiere una preparación de la data, solo implica construir la lógica que traerá la información en QuickSight.Diagrama

Descripción generada automáticamente

La segunda opción, es una implementación a la medida, esta implica leer y preparar la información de la base de datos a través del BaaS del Home y el gestor del dashboard, quien la retornarla al Front-end para su visualización con el apoyo de librerías de gráficos, esta opción tiene en contra que cada cambio en el dashboard implicaría un desarrollo y un despliegue.Diagrama

Descripción generada automáticamente

1. **Definición dinámica del sistema**

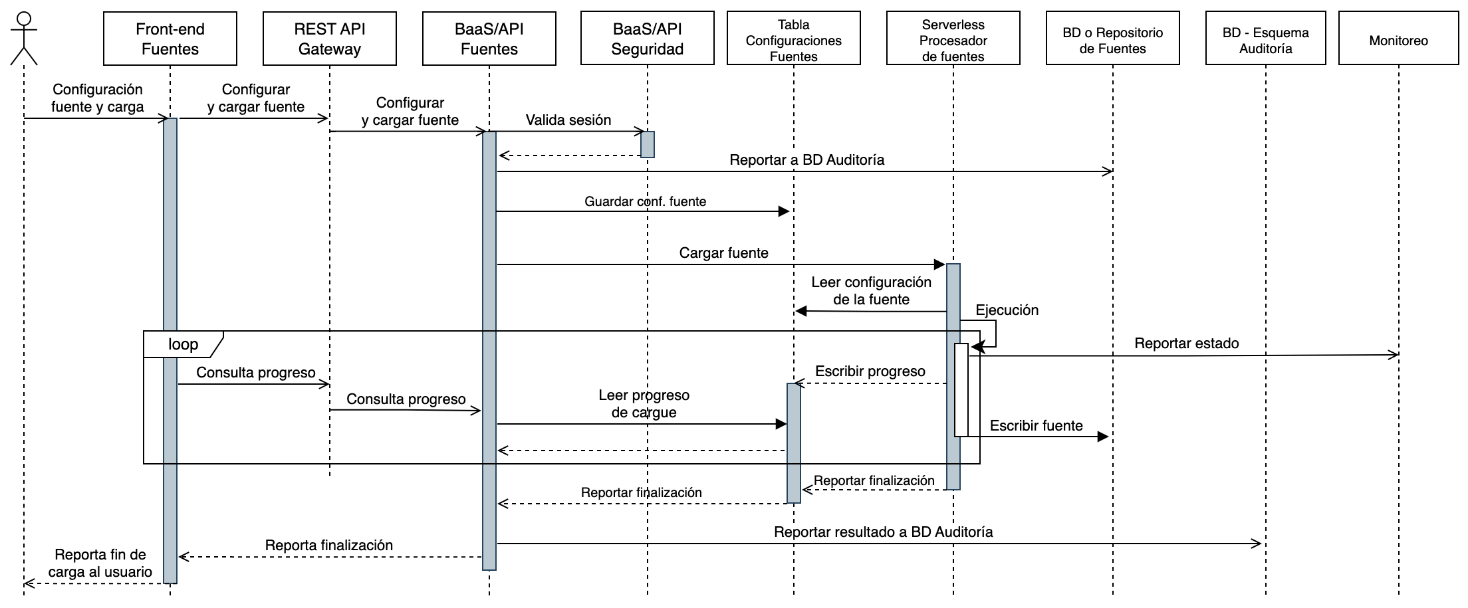
Total Report en su nueva arquitectura de Microservicios debe pasar al uso intensivo de comunicación REST y mensajería, las integraciones a nivel general deberán ser como en el siguiente gráfico.

1. **Detalle del flujo de configuración y carga de fuentes realizada por un usuario**

En el diagrama a continuación se presenta el flujo paso a paso de la configuración y creación de una fuente en Total Report desde una base de datos externa, en este flujo no se discrimina el proceso de consecución del token JWT, se parte desde la base de que el usuario ya realizó la autenticación y cuenta con JWT para realizar esta operación.

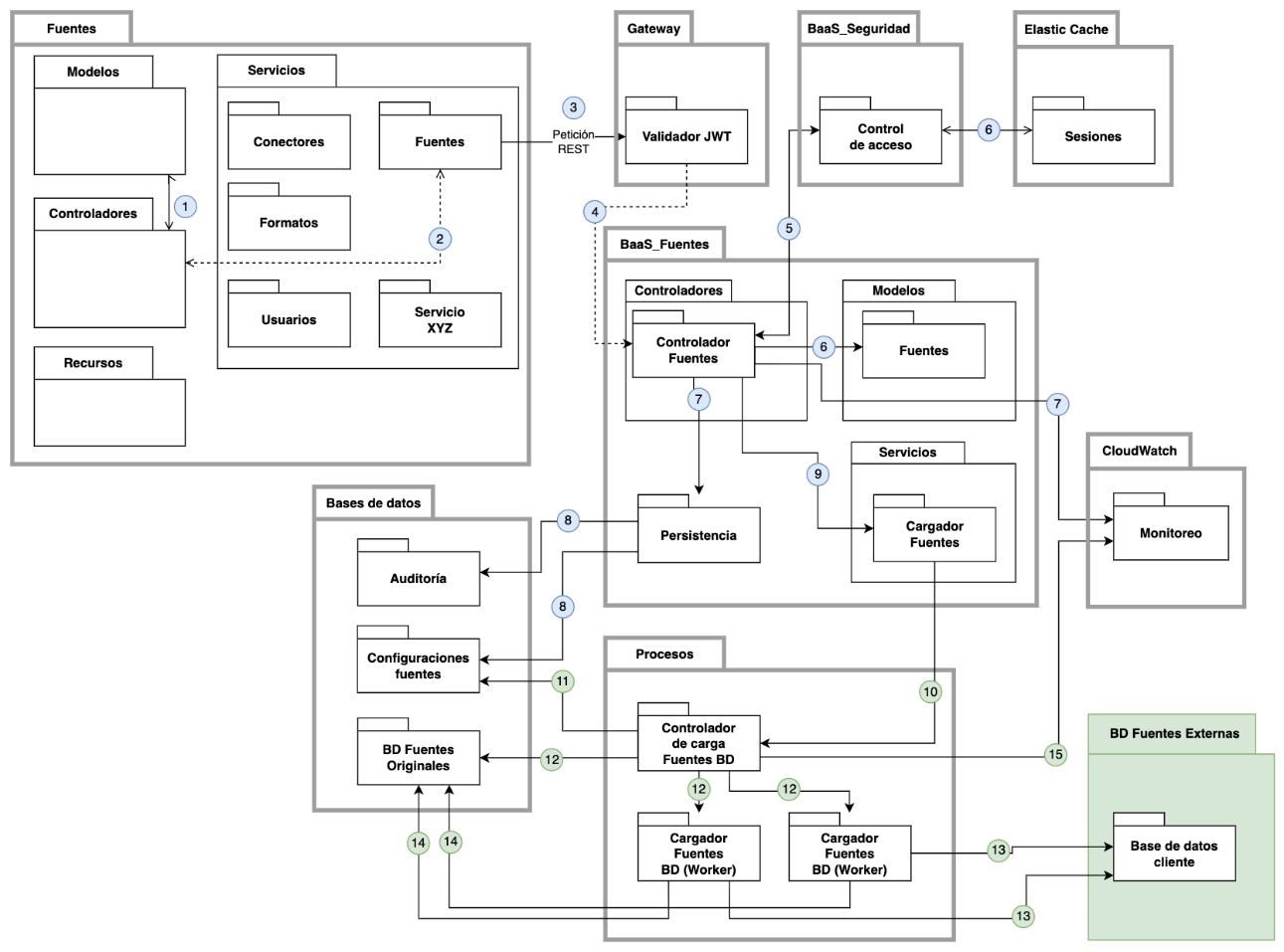
Esta definición permite resolver el escenario de calidad ESC\_001 identificado como prioritario, el cual requiere poder cargar hasta 100 millones de registros en aproximadamente 5 minutos.

**Diagrama de secuencia de carga de fuentes**



**Diagrama de paquetes de la carga de fuentes**

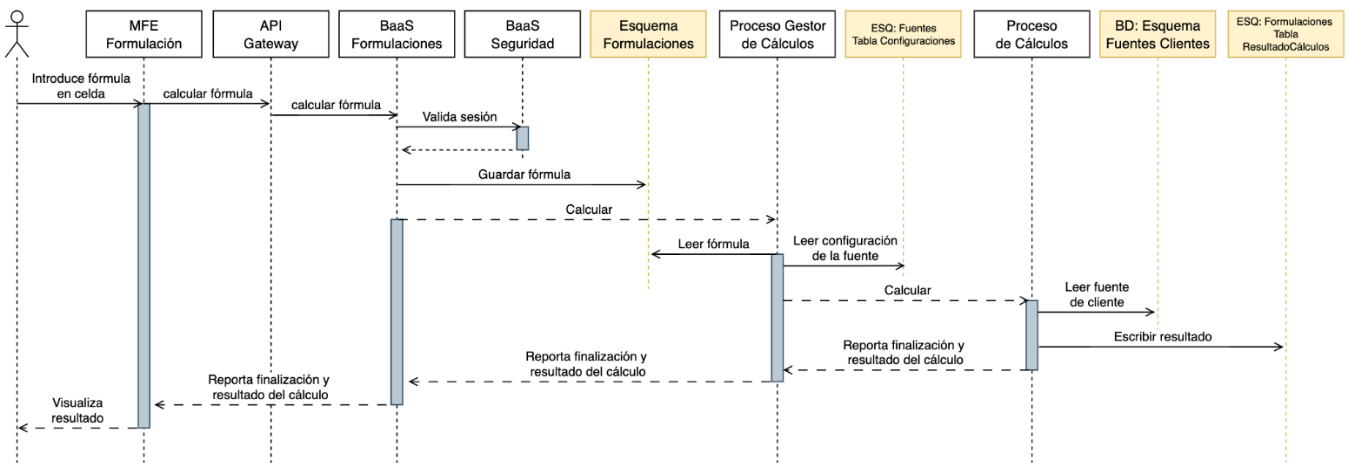
El diagrama siguiente es la vista estática del flujo previo de configuración y carga de fuentes.



1. **Detalle del proceso de formulación con cálculo interactivo y en lote**

El siguiente diagrama representa el flujo desde la base de un usuario ya autenticado y que ha definido las configuraciones generales de su reporte, así como fuentes a utilizar; este flujo representa una capacidad de respuesta cercana a tiempo real, es decir el flujo se inicia cuando el usuario presiona enter o sale de la casilla en la que se encuentra la fórmula que ha definido.

**Formulación y cálculo interactivo por celda**



El diagrama a continuación presenta la forma como el sistema manejaría un grupo de fórmulas de n tamaño, el cual requiere ser calculado rápidamente y visualizado por el usuario, tanto en el cálculo por celda, así como en lote, los flujos no pretenden detallar el subproceso validaciones.

**Formulación y cálculo en lote**Interfaz de usuario gráfica, Diagrama

Descripción generada automáticamente

1. **Detalle del proceso de validación de cálculos**

A continuación, se puede observar cómo el sistema distribuye la validación en tres componentes, un gestor y dos validadores, uno de forma y otro de contenido con base en las definiciones establecidas y almacenadas para una norma técnica y los reportes relacionados, de esta forma el proceso es escalable y claramente desacoplado, facilitando la mantenibilidad y apoyando el cumplimiento de los escenarios ESC\_002 y ESC\_005.

**Diagrama de secuencia del proceso de validación de cálculos**



1. **Detalle del proceso de transmisión**

Es siguiente diagrama proponer una solución de visualización y descarga de reportes de volumétricos, siguiente la misma estrategia de Baas, Gestor y Worker haciendo uso de S3.

Para la definición de la anterior solución se consideró la restricción RST-SIS-07 llegando a la conclusión de que no era conveniente transicionarla por los siguientes motivos:

Solo presenta como novedad el uso de una cola de mensajería y su enfoque fue para uso on-premise no tipo SaaS.

No presenta un mecanismo para obtener más rendimiento por mejor utilización del hardware, las colas de mensajería son un mecanímo de “sala de espera” no de mayor rendimiento por su simple implementación.

No presenta un mecanismo en el código que pueda representar una mejora significativa de rendimiento.

Python no es un lenguaje más veloz que .NET

El enfoque de esa solución no aborda Total Report de forma holística y no fue pensado para Microservicios.

No resulta más económico o reutilizable para la presente estrategia de solución, tampoco se recomienda transicionarlo en su estado actual e invertir esfuerzos para conectarlo con la presente solución.

El equipo de desarrollo de Project House no domina técnicamente la solución.

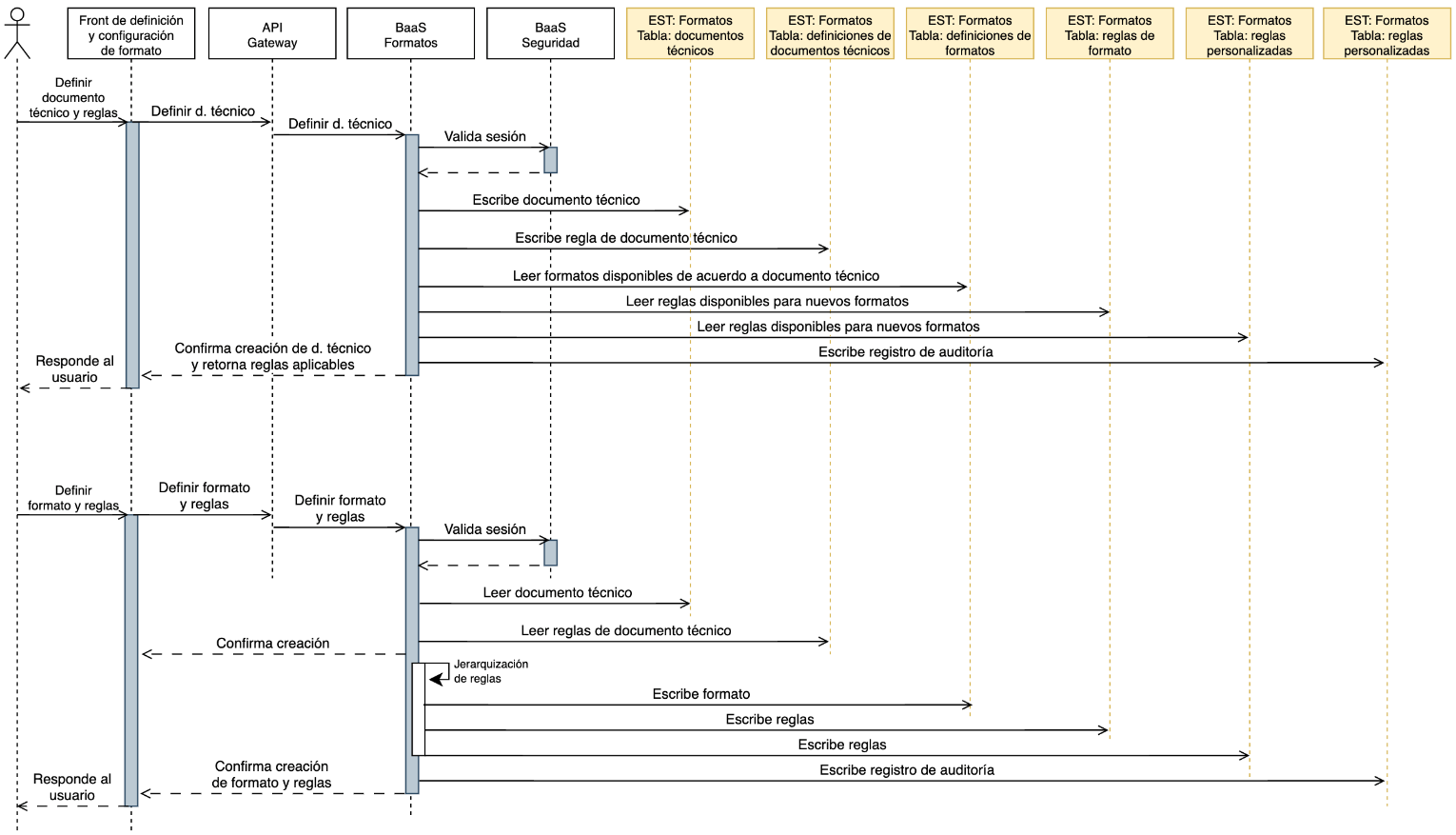
El componente no se utiliza y no es obligatorio su uso.

1. **Detalle del proceso de definición de reglas en documento técnico y formato**

El siguiente diagrama que describe el flujo técnico de la solución al ESC\_005, el cual solicita mejorar la implementación de reglas de negocio y procurar evitar esfuerzos de desarrollo, para este propósito, se han definido las siguientes premisas:

1. El usuario inicia desde el punto en que las reglas que puede aplicar al documento técnico ya fueron cargadas en el front-end.
2. Una regla aplicable a un reporte pertenece a un formato.
3. Un formato puede estar asociado a un documento técnico.
4. Una regla puede ser de tipo legal o normativo lo cual hace alusión las condiciones que la información debe cumplir o puede también ser de forma, es decir, cómo debe ser presentada dicha información al ente de control.
5. Las reglas pueden asociarse a un usuario, rol o empresa, sin embargo, el manejo de permisos no debe estar acoplado a la gestión de reglas.

**Diagrama de secuencia de la creación de reglas de negocio en documentos técnicos y formatos**



1. **Definición del proceso de generación de reportes por fecha**

Considerando la definición de los Microservicios responsables de la transmisión, formulación y cargue de fuentes, para este proceso se estableció el uso de un proceso Serverless Lambda con la configuración tipo cron o expresión agendada.

Para el siguiente diagrama se toma como base de que se debe incorporar en la transimisión la opción para el usuario de generar reporte de forma automática mediante fecha, dicha funcionalidad debe escribir las configuraciones de fuentes, formulación y transmisión utilizadas, desde este punto de partida se plantea el siguiente diagrama que da respuesta y solución al escenario ESC\_003.

Para referencia detallada sobre la configuración de funciones Lambda tipo cron, por favor revisar el siguiente link: <https://docs.aws.amazon.com/lambda/latest/dg/services-cloudwatchevents-expressions.html>

**Secuencia del proceso en segundo plano programado para generación de reportes automáticos**Diagrama

Descripción generada automáticamente

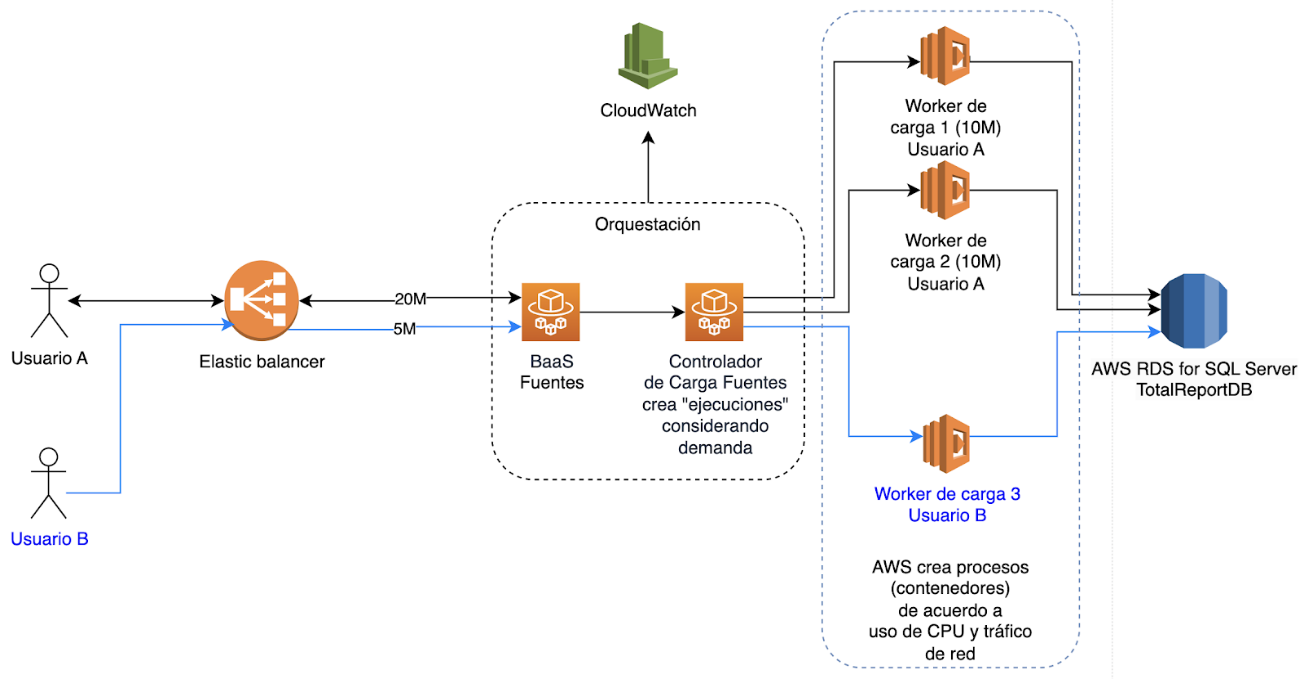
1. **Concurrencia y estrategia de escalamiento**

Para las condiciones específicas de Total Report, la aplicación deberá escalar y crecer horizontalmente, además, contar con recursos de hardware suficientes para su trabajo base, es decir, para aquellos componentes que administran la ejecución y no se espera escalamiento de ellos como los BaaS y los gestores o controladores de ejecución.

El escalamiento horizontal se podrá lograr bajo dos estrategias, la primera será bajo la lógica de los controladores o gestores, los cuales al recibir la carga de trabajo y estimarla, podrán iniciar múltiples procesos y división de los mismos a fin de completar la tarea en el menor tiempo posible, esto afectaría al cargue de fuentes, cálculos y generación de archivos planos o transmisiones tipo Web Service; esta forma de trabajo será la adecuada para procesos Serverless.

La segunda forma en la que la que los gestores o controladores podrán escalar será mediante el uso de procesador, delegando el proceso de creación de workers y destrucción de los mismos al proveedor de infraestructura AWS, esta segunda forma estará reservada para cuando el worker esté desplegado en un contenedor, el siguiente diagrama ilustra estas 2 opciones para los procesos de cargue de fuentes, cálculos y transmisiones.

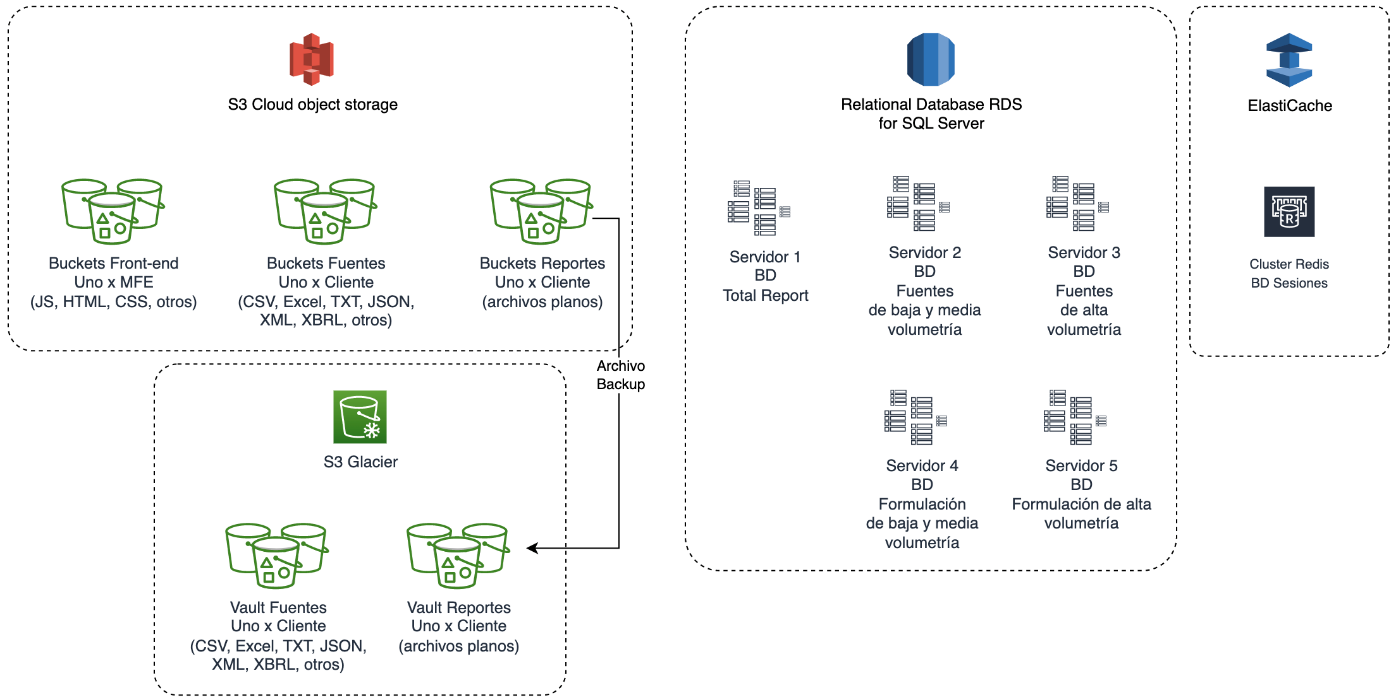
**Diagrama de escalamiento y concurrencia**



1. **Vista de Información**

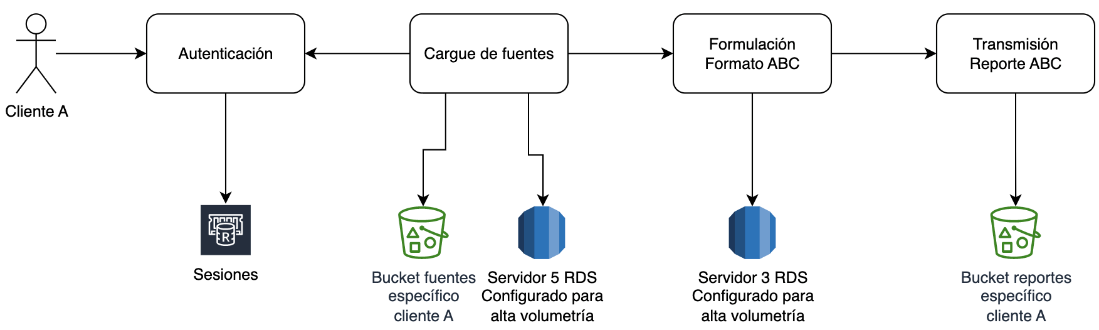
La situación actual de Total Report a nivel de base de datos dificulta su entendimiento, sin embargo, la estructura que se propone para dar soporte a la presente propuesta se puede entender en alto nivel en el diagrama a continuación, de este se destaca la separación en 3 tipos de almacenamiento:

1. Almacenamiento por objetos con AWS S3 el cual contendría:
   1. Las aplicaciones Front-end
   2. Las fuentes tipo archivo que los clientes carguen durante un tiempo límite de 1 a 3 meses.
   3. Los reportes generados a través del proceso de transmisiones a archivo plano, durante un tiempo límite de 1 a 3 meses.
2. Bases de datos RDS for SQL Server, este tipo de base de datos son SQL Server con capacidad de auto escalamiento, se estiman 3 servidores de este tipo:
   1. El primero para el uso de la aplicación misma.
   2. El segundo servidor para el almacenamiento y manipulación de formatos con volumetrías pequeñas y medianas.
   3. El tercer servidor tendría el propósito de manejar grandes volumetrías.
3. Base de datos ElastiCache – Redis para el manejo de sesiones de los usuarios.
4. Base de datos S3 Glacier para el archivo de documentos y fuentes por periodos superiores a 3 meses hasta varios años, esta información aún estaría disponible para los usuarios dentro de la aplicación y la vería de forma normal.



Las aplicaciones en la arquitectura de Microservicios emplean una combinación de bases de datos compartidas e independientes; para el caso de Total Report se estima dividir las bases de datos por clientes, volumetría de las fuentes y la formulación en consecuente, no por formatos, dado que la lógica y esfuerzo de cálculo residirá en los Workers, no en las bases de datos, como se sugiere en el siguiente gráfico.

**Flujo general y almacenamientos utilizados**

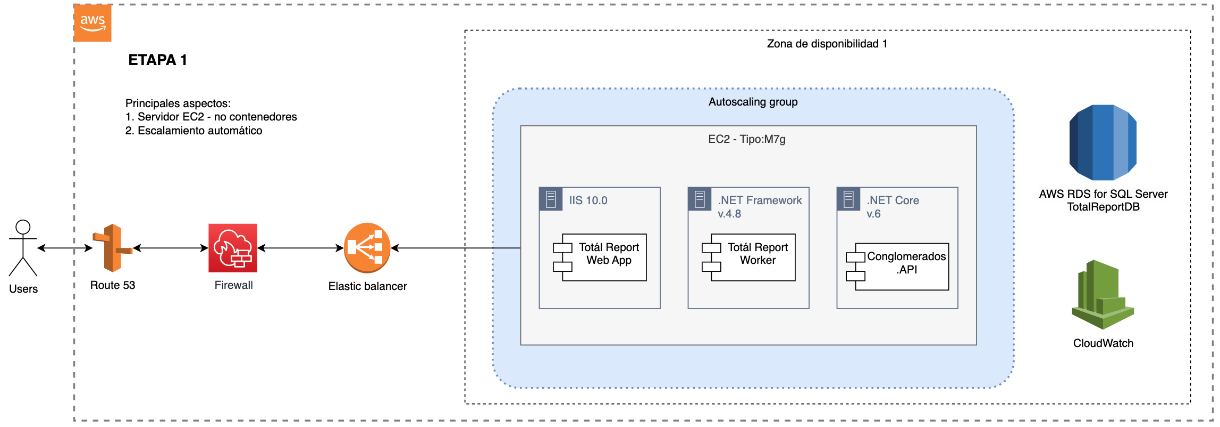


Se considera fundamental durante el proceso de migración de las funcionalidades, verificar en cada microservicio a construir, que la optimización de las estructuras, la normalización y uso de índices se cumpla para poder sacar el máximo desempeño.

1. **Vista de Despliegue**

La vista de despliegue presenta 3 etapas para minimizar el impacto que implica un gran cambio tecnológico, este proceso es lleva consigo el cambio de procesos manuales a automatizados es decir, en cada etapa se adoptarán cada vez más las capacidades propias de la nube que son automatizadas, a fin de reducir el esfuerzo administrativo y técnico derivado del manejo de la infraestructura, de este modo se podrá controlar mejor los costos teniendo en cuenta de que el principio general es el pago por consumo al final de las 3 etapas.

1. **Infraestructura de la etapa 1: Migración en su estado actual.**



En esta etapa se trasladará la aplicación en su estado actual a un servidor EC2, se recomienda tipo m7g.4xlarge el cual dispone de 16 núcleos y 64 GB de RAM con almacenamiento elástico, for favor consultar la información detallada en: [https://aws.amazon.com/ec2/instance-types/c7g/.](https://aws.amazon.com/ec2/instance-types/c7g/)

Como aspecto clave de esta etapa se considera una migración a una base de datos RDS for SQL Server la cual es realmente una base de datos SQL Server pero con capacidades de escalamiento.

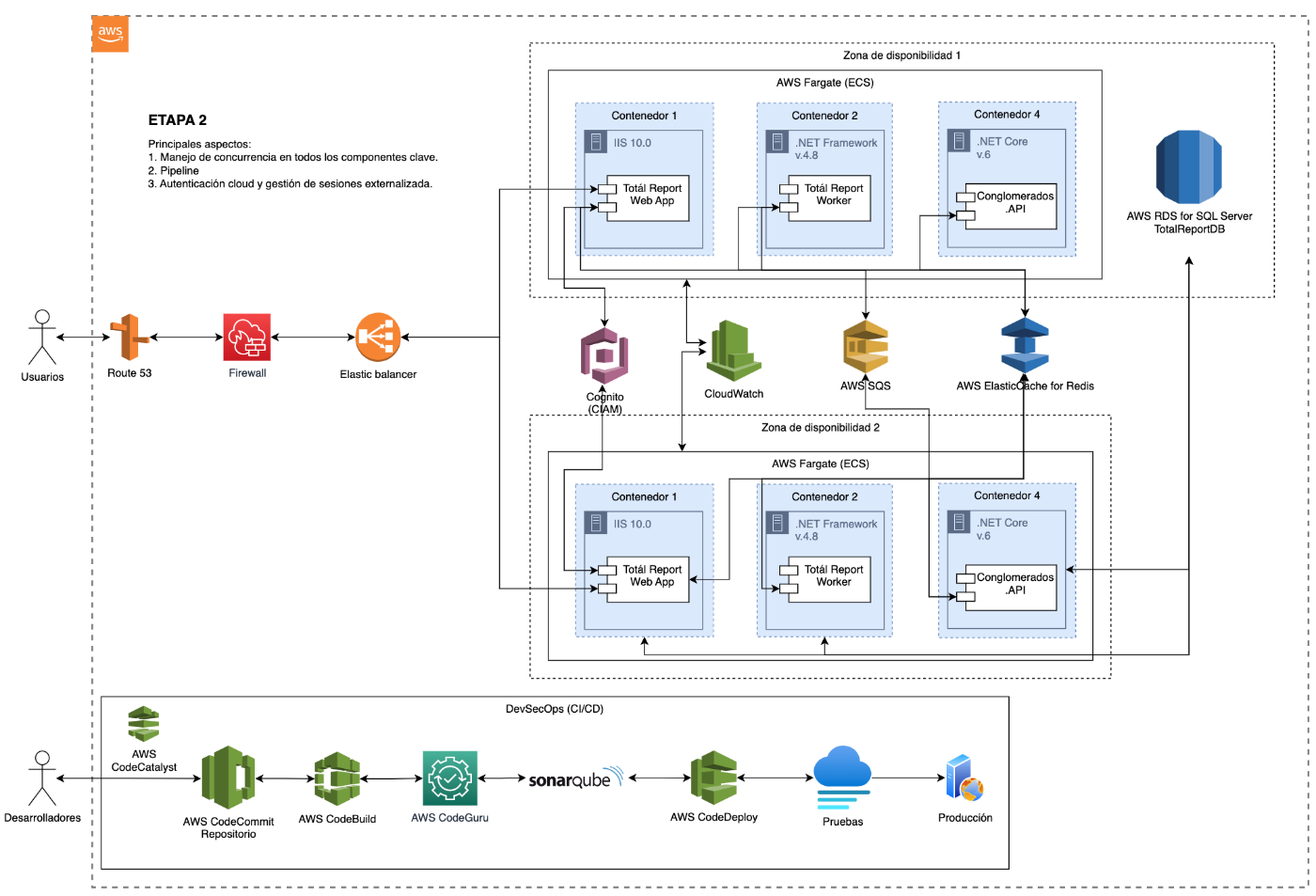
Como soporte al sistema y entendiendo de que la lógica no fue adaptada para trabajo en la nube, se considera importante el monitoreo continuo con CloudWatch de la infraestructura, a fin de poder reaccionar ante problemas inesperados.

1. **Infraestructura de la etapa 2: migración en etapa intermedia**

En la etapa 2 se contempla el uso de contenedores administrados por el proveedor AWS con el servicio Fargate, esto implica haber iniciado el proceso de división de la aplicación y la construcción de interfaces de comunicación REST.

Tambien se contempla el uso de servicios Cloud como:

* Cognito para las autenticaciones de los usuarios nuevos,
* CloudWatch para monitorear la aplicación tanto a nivel de infraestructura como de logs,
* Simple Queue System (SQS) como broker de mensajería,
* Elastic Cache para el almacenamiento de las sesiones
* Code Catalyst para la gestión completa DevOps despliegue e integración continua



1. **Infraestructura para la etapa 3 y selección tecnológicasd**

En esta etapa se tendrá toda la base de la transformación lista para evolucionar funcionalmente y con la capacidad de afrontarlo a nivel de desempeño, seguridad, mantenibilidad.

En adición a las etapas previas, se destacan los siguientes cambios:

* La gestión del front-end, el cual estará totalmente separado, almacenado en el sistema de archivos S3 y gestionado por el proveedor Cloud, no requerirá un servidor HTTP independiente.
* Todos los servicios y procesos en contenedores Fargate y funciones Serverless Lambda.
* Se empleará una puerta de acceso o Gateway gestionado por el proveedor para garantizar el uso seguro de los servicios, gestión de tokens y seguridad adicional.
* Gestión de accesos de administradores con AWS Identity Access Management (IAM), no con el sistema interno

Diagrama

Descripción generada automáticamente

1. **Proveedor Cloud y compatibilidad con otros proveedores**

Para este proyecto se ha recomendado el proveedor Amazon Web Services AWS por dos razones:

1. Es el proveedor más antiguo de los 3 más grandes (Azure, GCP y AWS) el cual ha demostrado que es muy sólido, confiable, rápido y especialmente más sencillo de utilizar.
2. Va en línea con la selección tecnológica de clientes importantes con Nequi a fin de intercambiar información con mayor seguridad.
3. Todos los servicios seleccionados en este diseño se encuentran homologados en las otras nubes y se puede cambiar de proveedor Cloud total o parcialmente.

* <https://labs.sogeti.com/aws-vs-azure-vs-google-cloud-comparison/>
* <https://www.cloudthat.com/resources/blog/aws-vs-azure-vs-google-cloud-top-3-cloud-platform-comparison-2>

1. **Modelo de responsabilidad y gestión de recursos Cloud**

Durante el proceso de migración hasta la etapa 3, la presente arquitectura tendrá un proceso de responsabilidad que pasará desde una gestión controlada por Project House o Infrastructure as a Service (IaaS) pago por tenencia o alquiler, hasta un modelo Software as a Services (SaaS) de responsabilidad compartida y pago por uso, como se ilustra en el siguiente diagrama de ejemplo:

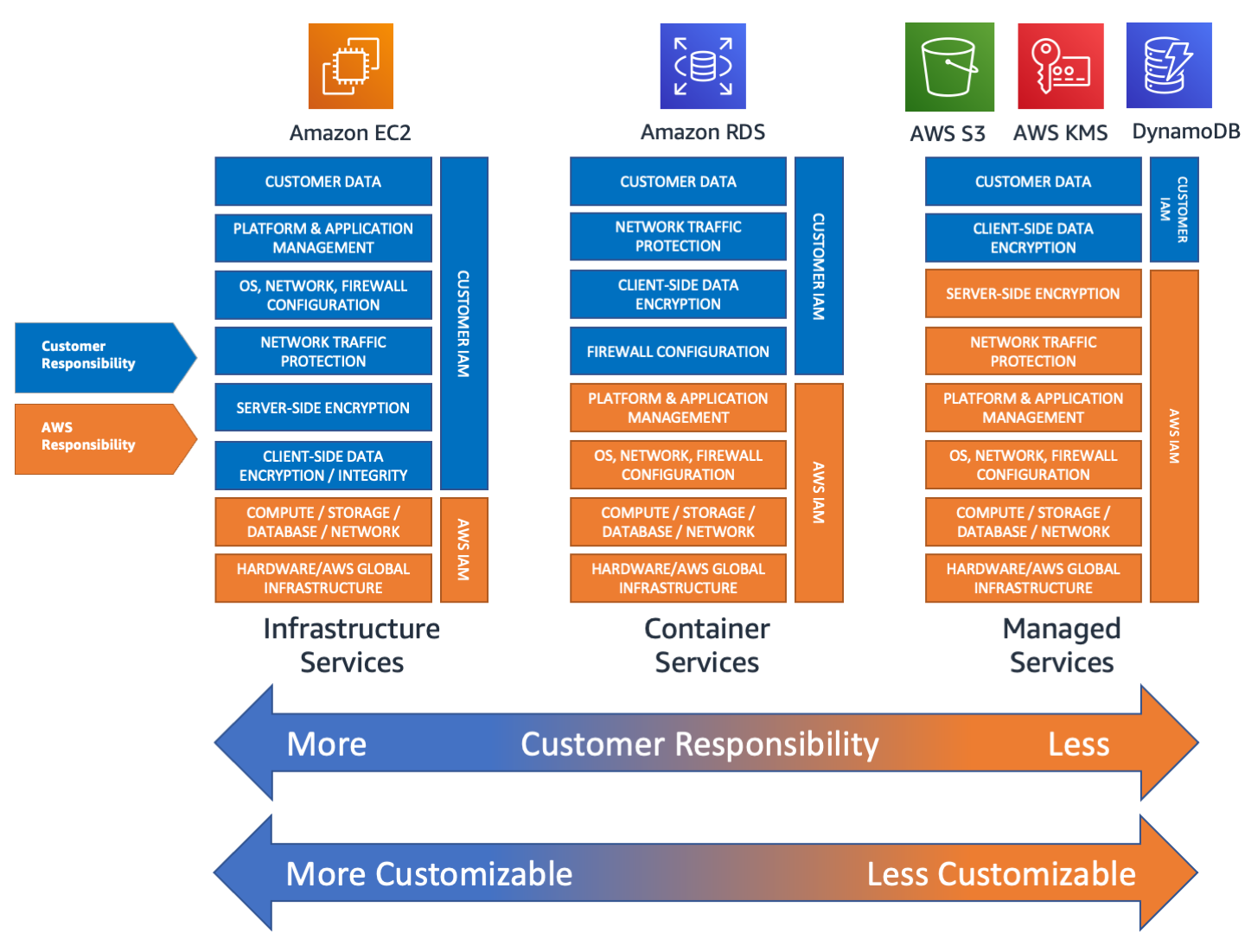
**Shared Responsibility Model by Service Types**

  Image tomada de: <https://aws.amazon.com/blogs/industries/applying-the-aws-shared-responsibility-model-to-your-gxp-solution/>

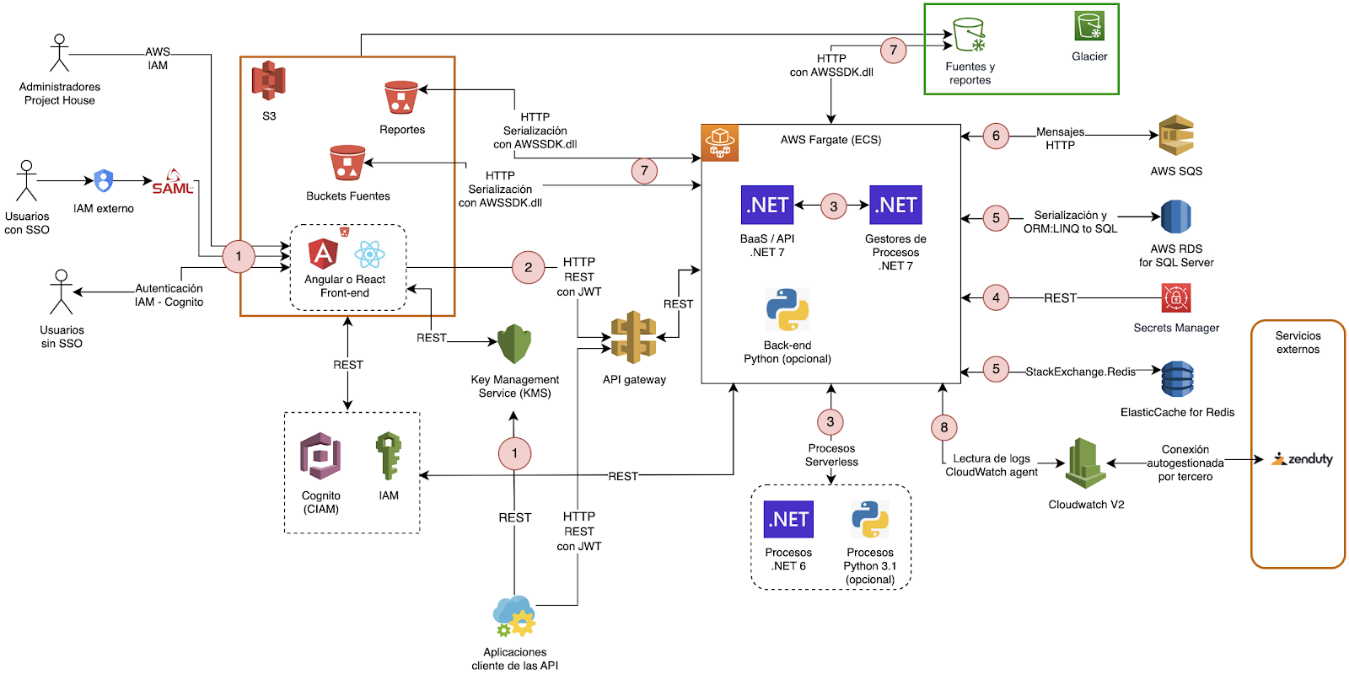
Los servicios de AWS cuentan con capacidades de escalamiento automáticos en sus versiones SaaS y no hay necesidad de hacer una gestión manualmente controlada del escalamiento, esto si es obligatorio en los modelos IaaS, el proceso para obtener esta gestión automática se puede definir en 2 momentos, sin riesgo para la aplicación, durante la configuración inicial del servicio, o posterior a su implementación, en el panel administrativo del mismo.

El reto del escalamiento está definido en la lógica, entendiendo que por ejemplo el próximo request de procesamiento podría ir a una nueva instancia de ejecución, por ejemplo un nuevo Lambda o “Worker”, la infraestructura gestionada por AWS solo monitoreará el uso del hardware y en caso de que necesite más potencia, automáticamente se tendrá una nueva instancia lista para trabajar, no hay que preocuparse por la IP de la instancia para poder enviarle peticiones como en las infraestructuras IaaS. <https://docs.aws.amazon.com/lambda/latest/dg/lambda-concurrency.html>

Este mismo principio se aplica para los casos en los que la lógica propia de la aplicación entra en juego, como Fargate, Lambda, RDS,  pero es aún más automático en servicios más abstractos como el API Gateway, IAM, etc. en estos últimos no hay que definir políticas, sino solo usar el servicio tanto como se desee, estos están totalmente ligados al principio de pago por uso y no por tenencia.

1. **Vista de Integración y Seguridad**

A continuación se describen las integraciones y la seguridad nivel general, entendiendo que ciertas tecnologías requieren un conocimiento profundo para su implementación a pesar de que el proveedor Cloud ofrezca facilitar su implementación, se debe mencionar que no es objetivo de esta arquitectura solución entrar en todos los detalles de todas las temáticas involucradas en las arquitectura de Microservicios.

La seguridad de la aplicación está basada en la correcta configuración de las políticas de seguridad  que AWS obliga a definir al crear y utilizar cada servicio, esto limita a través de IPs de ciertos rangos los accesos no deseados, adicionalmente, con implementación de los servicios IAM, Cógnito y CIAM para los usuarios, API Gateway y KMS para la aplicaciones que deseen comunicarse hacia el Back-end, no viceversa, del Back-end al front-end, finalmente AWS Secrets Manager para almacenar credenciales de bases de datos y evitar quemarlas en el código o servidores; este conjunto de soluciones son las que brindan seguridad a la aplicación junto con las buenas prácticas de desarrollo, no hay un orden de implementación, dado que simplemente se deben utilizar y evitar construir mecanismos ad-hoc de seguridad.

La siguiente tabla hace referencia al diagrama anterior, a fin de delinear los concepto técnicos y aplicaciones relacionadas en la seguridad e integración propuesta.

| **Componente** | **Descripción** |
| --- | --- |
| REST | Se hace referencia al estilo de arquitectura que se comunica por el protocolo HTTP con formatos JSON, no a la implementación específica RESTful. |
| JWT | JSON web token, es un estándar web para asegurar la comunicación en servicios web mediante una firma y encripción. |
| AWS IAM | Identity Access Management, es un aplicación para el manejo seguro de identidades posterior a los directorios activos y para distintos propósitos. AWS cuenta con este servicio para la creación y mantenimiento de usuarios, tanto administradores, como clientes personas o APIs que deseen consumir servicios. |
| SAML2 | Es un protocolo para la comunicación relacionada con la autenticación federada mediante certificados, permite el uso de credenciales únicas o Single Sign-On (SSO) de las empresas clientes en Total Report. |
| AWS Cognito | Es un servicio específico de AWS para la autenticación y manejo de clientes de la aplicación, este servicio es también conocido como Customer Identity Access Management (CIAM), este servicio no está enfocado a la autenticación de usuarios SSO. |
| AWS KMS | Key Management System, es un sistema enfocado en el almacenamiento, generación y manejo seguro de llaves criptográficas, este sistema es recomendado para la comunicación del fron-end con el back-end así como para la comunicación con terceros que deseen consumir los servicios de Total Report, utiliza servicios REST para la entrega de secretos. |
| AWS Secrets Management | Este servicio permite el almacenamiento y gestión seguro de claves de acceso a bases de datos, así como la rotación de las mismas. |
| AWSSKD.dll | Librería provista por AWS para facilitar la comunicación entre las aplicaciones y los servicios de AWS. |
| AWS API Gateway | Servicio de AWS que permite gestionar y proteger las aplicaciones tipo API que exponen servicios, responsables de la seguridad mediante tokens JWK. |
| AWS Fargate | Fargate es un servicio de AWS basado en AWS Elastic Container System (ECS), el cual facilita la gestión de contenedores. |

Todos los servicios AWS cuentan con amplia documentación en línea, se recomienda la revisión para su utilización.

Link documentación: <https://docs.aws.amazon.com/index.html>

**Descripción general de las integraciones:**

1. La primera integración utilizaría los servicios de IAM, Cognito y SAML2.
2. El usuario debe de crearse en IAM y contar con un rol definido, esta información debe de leerse por el Microservicio de Seguridad.
3. El usuario ingresaría con credenciales y IAM validaría la autenticación más autorización para algunos casos que se requiera acceso a servicios AWS, no solo en Total Report.
4. La versión SAML2 primero validaría las credenciales en el IAM externo de la empresa cliente y una vez autorizado se comunicaría con TotalReport con datos previamente acordados, a fin de poder ingresar a la aplicación.
5. Las APIs consumidoras de servicios, primero deberán obtener un token JWT en KMS y con él presentar sus peticiones a al API Gateway lo cual validará.
6. Luego de cualquiera de estos procesos el sistema debe crear la sesión en Redis y permitir al usuario ingresar.
7. Las aplicaciones Front-end deberán obtener un JWT en KMS antes de poder llamar al Back-end representado por el API Gateway.
8. La comunicación interna de la aplicación se recomienda sea directa REST con JSON sin uso de JWT.
9. La comunicación al Secret Manager debe ser de tipo consumir de API REST y así obtener usuario y clave.
10. La comunicación a las bases de datos RDS y Redis por parte de las aplicaciones, deberán obtener primero las credenciales del Secrets Manager, esta comunicación se facilita por el conector AWSSDK.dll.
11. La comunicación a la cola de mensajería es de tipo TCP/IP y se facilitar por el conector AWSSDK.dll que deberán tener las aplicaciones que lo requieran.
12. La comunicación a los servicios de almacenamiento de archivos deberán ser tipo HTTP con métodos PUT, GET, POST, DELETE etc. con el apoyo del AWSSDK.dll.
13. La comunicación entre los LOGS de las aplicaciones AWS CloudWatch se llevará a cabo con el apoyo del mediador CoudWatch agent.

1. **MODELO DE ENTREGA**

El modelo de entrega define la forma y actividades principales relacionadas con la materialización de la presente propuesta y los tiempos de cada una, así como el relacionamiento generado.

Se estima que se requiere establecer una aplicación base con la funcionalidad general como soporte a la migración de los formatos, entendiendo que cada formato puede requerir un esfuerzo separado y son claros candidatos para convertirse en Microservicios.

1. **Estrategia de migración**

Como premisas importantes, se recomienda realizar la migración por funcionalidades, siguiente el flujo normal de los usuarios, es decir, creación de usuario, empresa, autenticación, creación de fuentes, formulación y transmisión.

Se debe tener en cuenta de gran parte del Front o interfaz de usuario debe reconstruirse para poder funcionar independientemente, el sistema actual de páginas renderizadas desde el servidor impide la modificabilidad y reusabilidad, puesto que cada cambio significa compilar y desplegar otras partes de la aplicación que no fueron modificadas, esto es acoplamiento.

También se debe mantener una versión actual de la aplicación como soporte a la operación y tener en cuenta de que no todas las funcionalidades o formatos deberán migrarse a microservicios Cloud.

1. **Pasos para la ejecución de la migración**

La siguiente lista es aplicable tanto a la consolidación de las aplicaciones base como a la implementación de nuevos formatos.

* 1. Definir el costo – beneficio de realizar la migración de la funcionalidad objetivo.
* 2.  Definir los componentes (clases, variables, tablas, librerías, etc.)  que intervienen actualmente en el flujo a migrar.
* 3. Identificar en detalle los métodos, variables y dependencias involucradas.
* 4. Determinar la viabilidad y esfuerzo de la migración teniendo en cuenta si se desea migrar en las condiciones actuales o se desea implementar una mejora en la funcionalidad.
* 5. Determinar las responsabilidades que deben ir en BaaS, a saber:
  + Respuesta a las peticiones del front
  + Orquestación del flujo
  + Comunicación con los gestores de procesos batch y componente de seguridad,
  + Comunicación con base de datos
  + Gestión de otras lógicas particulares que no justifican a un nuevo microservicio ej. notificaciones, actualizaciones de estado, etc.)
* Determinar las funcionalidades que deben ir en el administrador o gestor asociado, estas incluyen:
  + Evaluación y control de la ejecución de los escalamientos.
  + Procesos batch o en segundo plano.
  + Control de la comunicación y ejecución de los Workers.
  + Consolidación de resultados
  + Comunicación con los sistemas de almacenamiento.
* 6. Determinar las funcionalidades que deben estar en los workers, considerando que son escalables horizontalmente, deben leer la configuración de trabajo de una cola o base de datos, ejecutar una tarea específica y terminar su ciclo de vida, estos workers pueden ser contenedores o funciones Serverless.
* 7. Determinar el esfuerzo necesario para construir el Front-end, es decir, pantallas o páginas, funcionalidades, componentes evaluando complejidad y dependencias con el back para establecer el orden.
* 8. Definir los servicios Cloud necesarios para la funcionalidad, CloudWatch, Elastic Cache, IAM, Fargate y opcionalmente SQS están casi siempre involucrados activamente.
* 9. Definir estimación, plan de ejecución y pruebas.

1. **Plan de acción**

**Fases parincipales de la transformación de Total Report a Microservicios en la nube**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Fase** | **Momento 1** | **Momento 2** | **Momento 3** | **Momento 4** | **Momento 5** |
| **1. Migración de la aplicación actual** | Creación de cuenta y servicios | Transferencia y ajustes | En operación Operación | En operación Operación | En operación Operación |
| **1.1 Transición de servicios y adaptación inicial a la nube** |  |  | Configuración, desarrollo y pruebas |  |  |
| **2. Adaptación y/o construcción de nuevas aplicaciones base** |  |  |  | 1.Desarrollo y pruebas  2. Inicio migración clientes | Migración clientes |
| **3. Adaptación y desarrollo de formatos específicos (proceso ciclico)** |  |  |  |  | Desarrollo y pruebas |

El siguiente plan de acción sugiere el relacionamiento entre actividades y el equipo de trabajo mínimo, los conocimiento ideales a tener son los que se encuentran en el listado del anexo 6.4 :

**Personal de trabajo sugerido fase 1 y 1.1:**

* Líder técnico con conocimientos medios-alto de AWS
* Un desarrollador back-end con conocimiento de la aplicación.
* Opcional: Experto en infraestructura AWS

**Personal de trabajo sugerido fase 2 y 3 (Listado para 2 equipos o squads):**

* 2 líderes técnicos con conocimientos medios-alto de AWS y el listado de tecnologías  del anexo 6.4
* 4 a 6 desarrolladores Back-end con conocimientos nivel médio en .NET, MySQL Server
* 2 Desarrolladores Front-end con conocimientos nivel medio en Angular, Javascript, HTML y CSS.
* Business Analyst o UX designer - Persona con conocimiento funcional para apoyo a la definición de historias y diseños de interfaces.
* Opcional: DBA
* Opcional: Arquitecto de software
* Opcional: Experto en infraestructura AWS
* Opcional: Scrum master o project manager.

**Personal mínimo requerido para las etapas 2 y 3**

Este listado es una opción al listado anterior, el cual permite avanzar el proyecto a un menor ritmo:

* Un líder técnico con conocimientos medios-alto de AWS y el listado de tecnologías  del anexo 6.4
* 2 desarrolladores Back-end con conocimientos nivel médio en .NET, MySQL Server
* 1 Desarrollador Front-end con conocimientos nivel medio en Angular, Javascript, HTML y CSS.

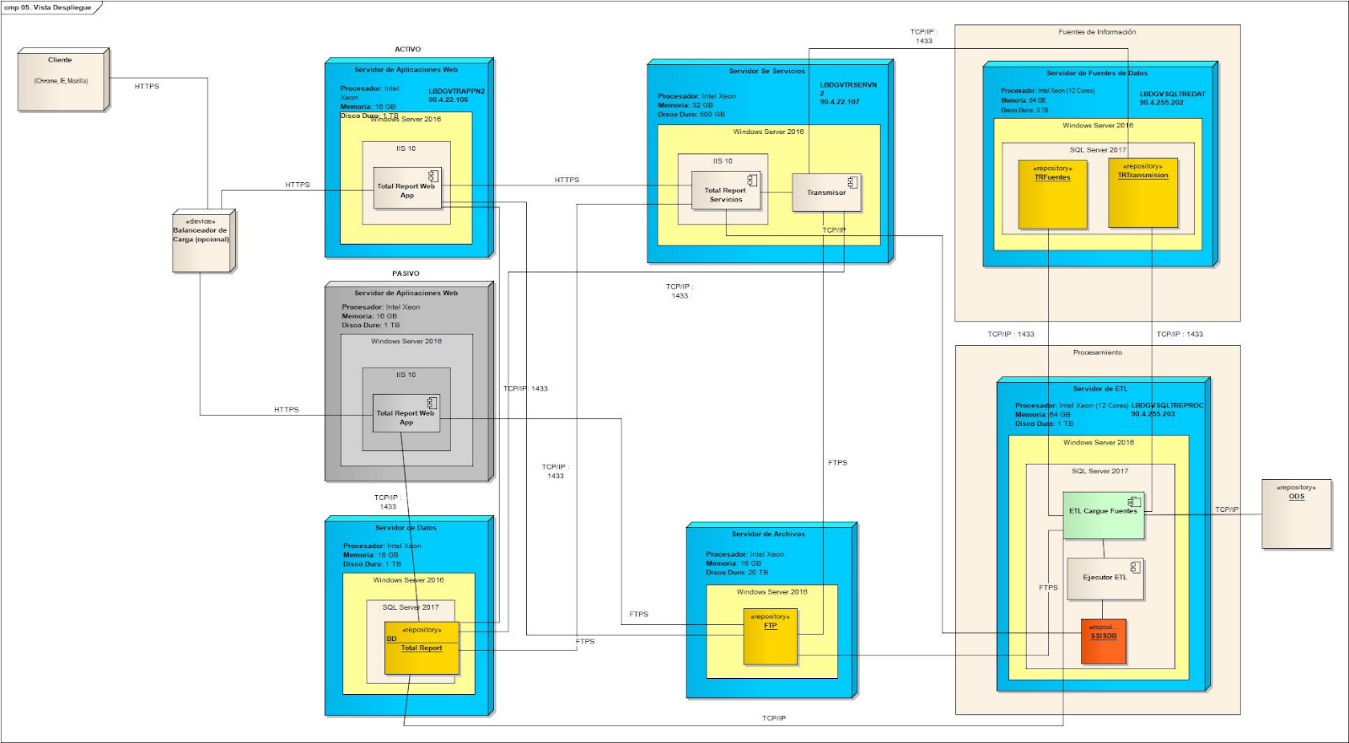
**Aproximación al detalle de actividades y tiempos**

Listado pensado para copiar a MS Project

|  |  |
| --- | --- |
| **Actividad** | **Semanas** |
| **Preparación ambiente de desarrollo** | 4 |
| Configuración de accesos | 0,2 |
| Preparación repositorios | 0,2 |
| Preparación ambientes | 0,2 |
| Configuración del pipeline | 1 |
| Configuración de bases de datos | 2 |
| Configuración de CodeGuru | 0,2 |
| Configuración de contenedores (Fargate) | 0,2 |
| Configuración DNS, Firewall y balanceador | 0,2 |
| Preparación de almacenamiento estático | 0,2 |
| Configuración de CloudWatch | 0,2 |
| Configuración del broker SQS | 0,2 |
| Pruebas de despliegue | 0,2 |
| **Desarrollo y configuración del módulo de Administración, autenticación y sesiones** | 8 |
| Identificación de elementos y objetos migrables | 0,6 |
| Desarrollo del front-end | 7 |
| Migración y desarrollo front gestión de usuarios | 2 |
| Migración y desarrollo front-end gestión de permisos | 2 |
| Migración y desarrollo front-end configuración de sistema | 2 |
| Desarrollo back-end | 7 |
| Configuración de Elastic Cache | 0,4 |
| Desarrollo módulo de autenticación federada SAML2 | 2 |
| Migración y desarrollo BaaS Administración | 4 |
| Desarrollo BaaS Seguridad | 3 |
| Desarrollo BaaS Seguridad - módulo de sesiones | 1 |
| Desarrollo BaaS Seguridad - módulo de autenticación entre servicios | 2 |
| Pruebas | 1 |
| **Migración y desarrollo del Home / Dashboard** | 3 |
| Identificación de elementos y objetos migrables | 0,4 |
| Desarrollo del front-end | 2 |
| Desarrollo del back-end  - BaaS | 2 |
| Pruebas | 1 |
| **Migración y desarrollo de cargue de fuentes** | 8 |
| Identificación de elementos y objetos migrables | 0,6 |
| Desarrollo del front-end | 3 |
| Cargue desde bases de datos | 1 |
| Cargue desde archivos | 1 |
| Cargue desde servicio web | 2 |
| Desarrollo del back-end | 6 |
| Migración y desarrollo del controlador y servicio | 1 |
| Migración y desarrollo del módulo de persistencia | 1 |
| Migración y desarrollo del proceso administrador de carga | 1 |
| Migración y desarrollo del worker de carga de bases de datos | 1 |
| Migración y desarrollo del worker de carga de archivos | 1 |
| Migración y desarrollo del worker de carga desde aplicaciones (web services) | 2 |
| Migración y desarrollo de la base de datos de carga de fuentes | 1 |
| Pruebas | 2 |
| **Migración y desarrollo de configuración de formatos** | **4** |
| Identificación de elementos y objetos migrables | 0,4 |
| Desarrollo del front-end | 2 |
| Desarrollo del back-end  - BaaS | 3 |
| Pruebas | 1 |
| **Migración y desarrollo de formulaciones** | **10** |
| Identificación de elementos y objetos migrables | 1 |
| Desarrollo del front-end | 3 |
| Construcción del módulo de configuraciones (fuentes, formatos, otros) | 2 |
| Construcción del módulo interactivo “Excel” | 3 |
| Desarrollo del back-end | 6 |
| Migración y desarrollo del controlador y servicio | 3 |
| Migración y desarrollo del módulo de persistencia | 0,8 |
| Migración y desarrollo del gestor de cálculos | 2 |
| Migración y desarrollo del procesador de cálculos | 2 |
| Migración y desarrollo del procesador de fuentes formatos | 1 |
| Desarrollo del manejo de mensajes por AWS SQS | 1 |
| Migración y desarrollo de la base de datos | 1,4 |
| Pruebas | 3 |
| **Migración y desarrollo de validaciones** | **7** |
| Identificación de elementos y objetos migrables | 0,6 |
| Desarrollo del front-end | 6 |
| Construcción del módulo de creación de documentos técnicos | 2 |
| Construcción del módulo de creación de formatos | 1 |
| Construcción del módulo de creación de reglas | 2 |
| Construcción del módulo de gestión de validaciones | 2 |
| Desarrollo del back-end | 5 |
| Construcción del módulo de gestión de validaciones | 2 |
| Construcción del módulo validador de reglas de contenido | 2 |
| Construcción del módulo validador de reglas de forma | 2 |
| Migración y desarrollo de la base de datos | 1 |
| Pruebas | 2 |
| **Migración y desarrollo de Transmisión** | **4** |
| Identificación de elementos y objetos migrables | 0,6 |
| Desarrollo del front-end | 2 |
| Migración y desarrollo del módulo de visualización de documentos | 0,8 |
| Construcción del módulo de configuración de transmisión | 2 |
| Desarrollo del back-end | 3 |
| Migración y desarrollo del BaaS de transferencias | 2 |
| Migración y desarrollo del gestor de transferencias | 2 |
| Migración y desarrollo del worker de transferencias | 1 |
| Pruebas | 1 |
| Pruebas generales end to end (opcional) | **2** |
| **Total lineal aproximado** | **48** |

1. **Pasos para la migración de Total Tax**

Total Tax es una aplicación para el mismo propósito de generación de reportes pero con otra arquitectura basada en cliente servidor, la transición para ella es un poco más simple, el proceso de migración de Total Tax se contempla en 2 etapas:

**Vista de despliegue de Total Tax**

Etapa 1, migración de la aplicación actual a contenedores de Fargate y RDS for SQL respectivamente:

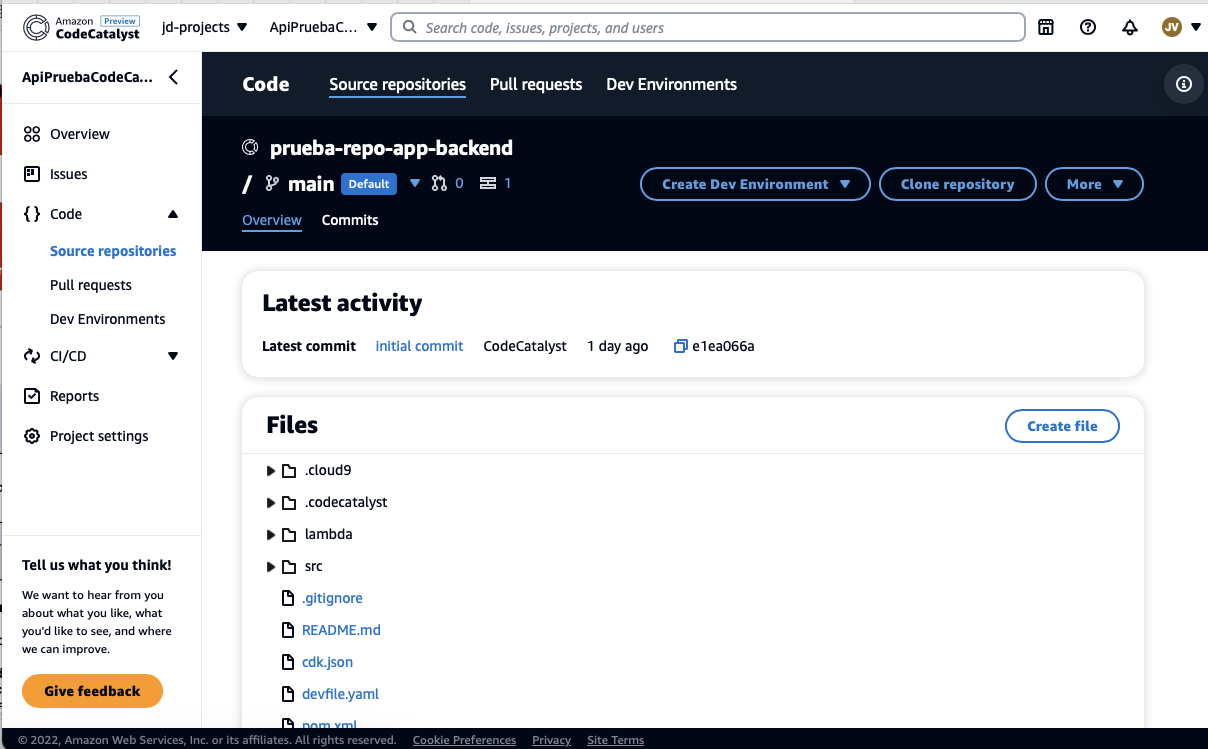
* Total Report Web, pasarse a contenedor.
* Servidor de Datos, pasarse a RDS for SQL server.
* Servidor de fuentes, pasarse a RDS for SQL server.
* Servidor ETL, pasarse a RDS for SQL server.
* Servidor de archivos, pasarse a S3 o EC2 con Windows server 2016.

Se debe procurar permitir a la aplicación ejecutarse de forma igual a su estado actual pero en la nube.

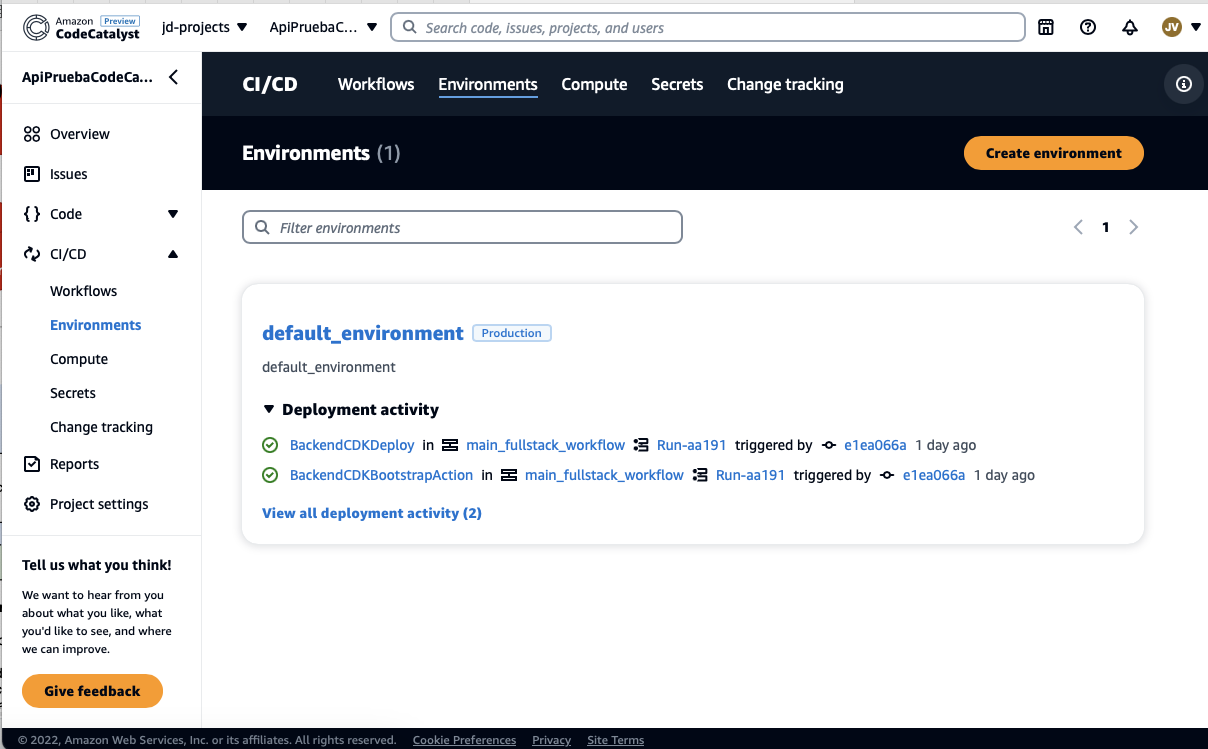
Etapa 2. Se debe revisar formato por formato para aplicar los pasos descritos en el numeral 5.2 y extraer la lógica para adaptarla a nueva disposición de Microservicios, esto se debe llevar a cabo posterior al plan descrito anteriormente, a fin de poder adaptar Total Tax a la nueva a arquitectura y no viceversa.

1. **PRUEBAS DE CONCEPTO Y ANEXOS**
   1. **Prueba 1. Verificación de CodeCatalyst como solución DevOps**

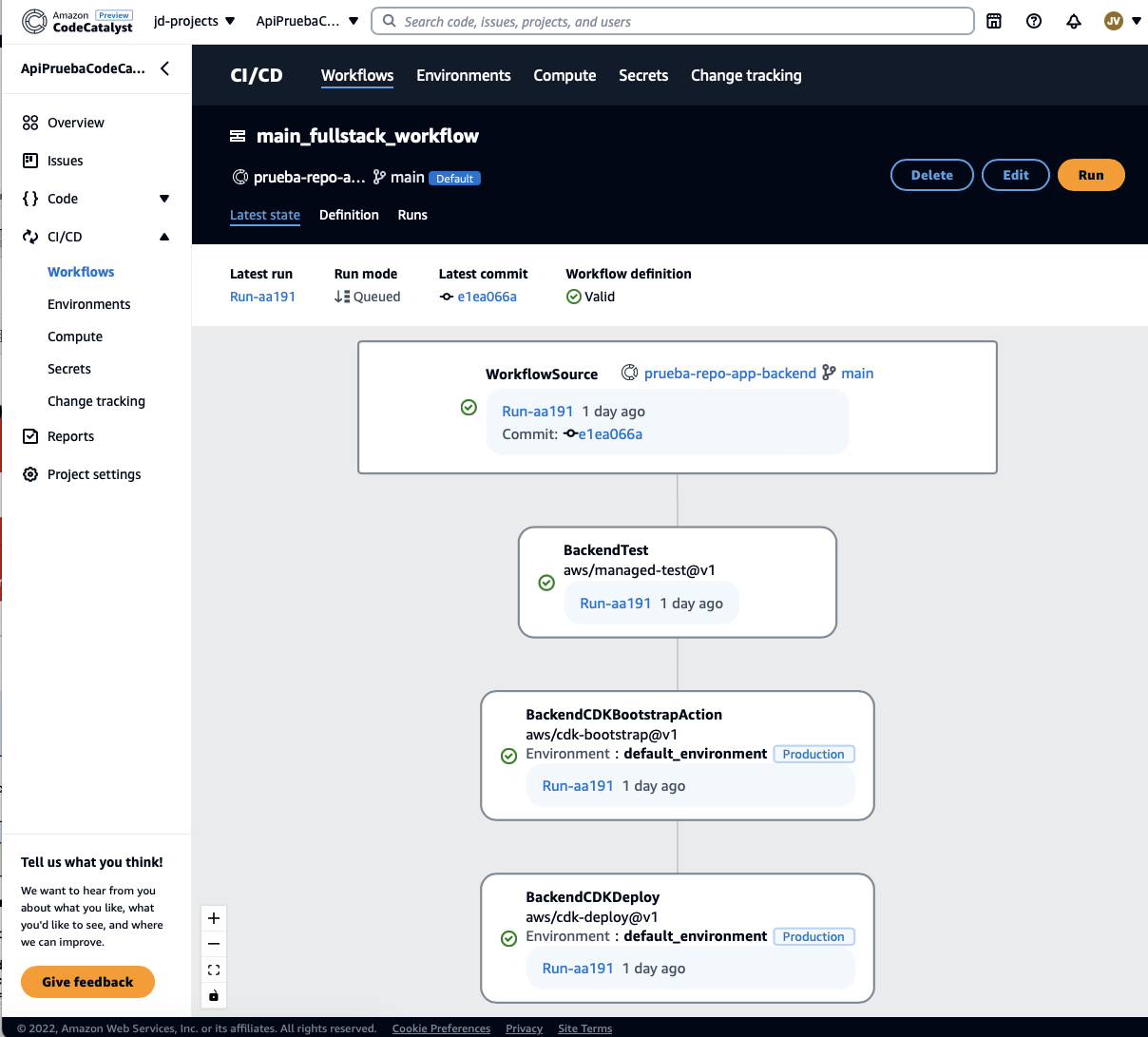
Las siguientes imágenes evidencian las capacidades para gestionar la integración continua y despliegue continuo de la aplicación a través de CodeCatalyst el cual es la nueva solución de AWS.



Visualización de un repositorio.



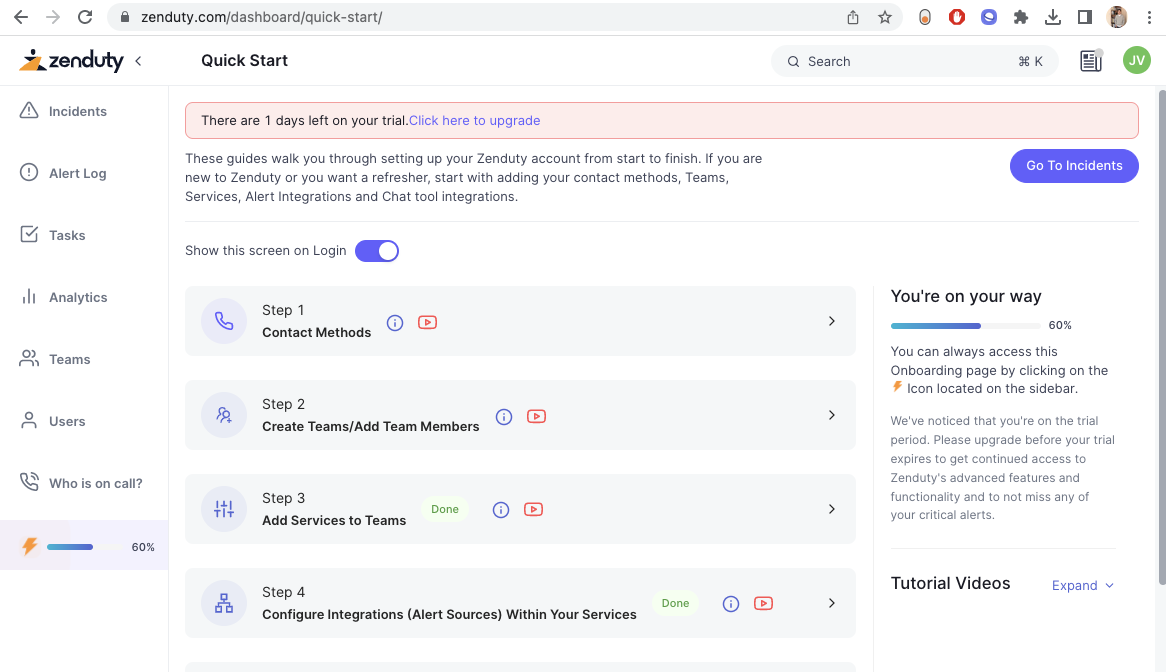
Visualización de la gestión de ambientes.

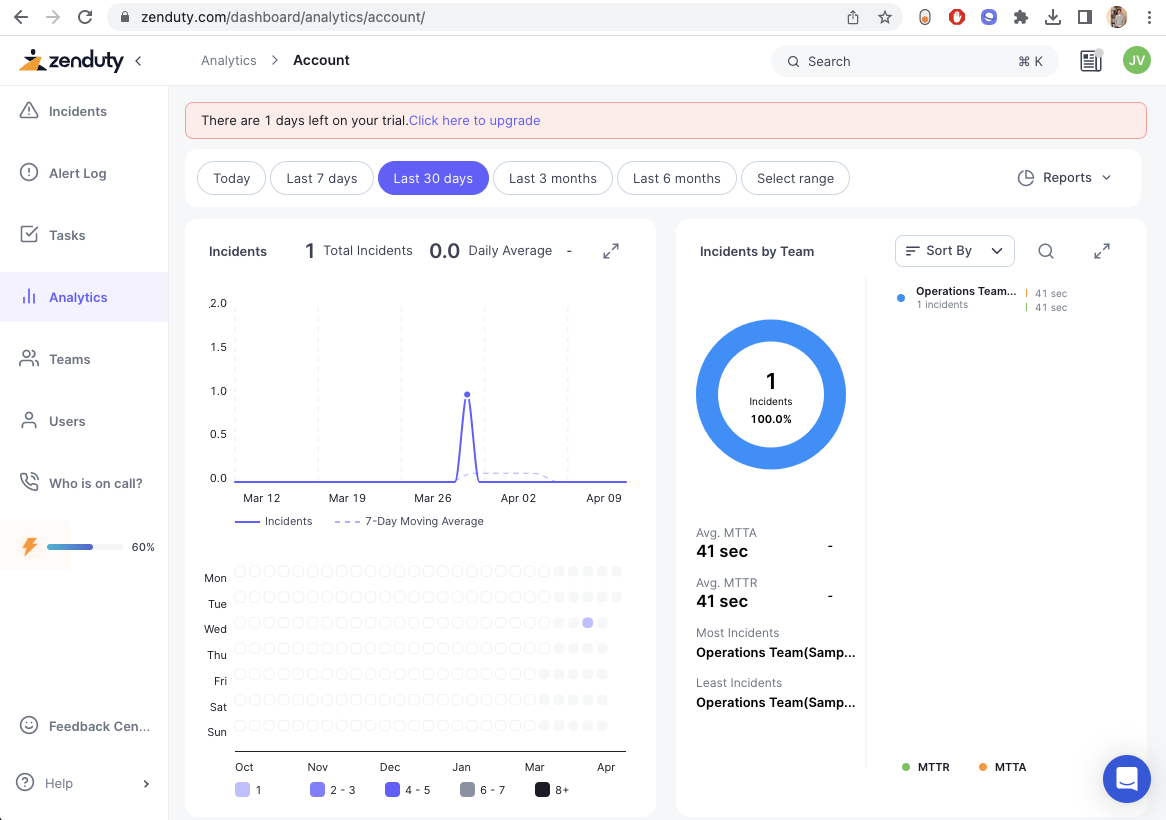


Visualización de la gestión de despliegues

1. **Prueba 2. Zenduty como solución de llamadas de soporte.**

Esta aplicación realiza llamadas de soporte automáticas a problemáticas encontradas en la ejecución Cloud, facilitando la atención inmediata cuando el sistema más lo requiera.





1. **ANEXO 1. Taller de Atributos de Calidad**

Durante el taller se identificaron y priorizaron los siguientes escenarios de calidad:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Escenario** | **Prioridad** | **Darwin** | **John** | **Oscar** | **Ricardo** | **Omar** | **Daniela** | **Total** |
| 1 | Como Ingeniero Implementador quiero que se pueda incorporar una Dashboard después de la autenticación del usuario, donde se visualice (Formatos a presentar, áreas implicadas en el proceso, avisos por correos electrónicos, gráficos donde muestre el comportamiento de los formatos ante las entidades, informes automáticos)  esto con el fin de tomar decisiones a través esta dashboard. (usabilidad, integración) | 3 |  | 1 | 1 |  | 1 |  | 3 |
| 2 | Como usuario final, al hacer la automatización de generación y transmisión de formatos, esta pueda efectuarse para 10000M de registros en solo 30 minutos. | 1 | 2 | 1 | 1 | 1 | 1 | 2 | 8 |
| 3 | Como especialista en regulación quiero que la aplicación tenga una usabilidad intuitiva con las validaciones que sean fáciles de crear, modificar y eliminar con el objetivo de no depender de implementación o del equipo de desarrollo. (usabilidad, modificabilidad) |  |  |  | 1 |  | 1 |  | 2 |
| 4 | Un usuario requiere que al poner la fecha de corte la aplicación le genere los siguientes procesos: Cargue de fuentes - generación formato - generación Archivo plano (usabilidad) | 3 |  | 1 |  | 1 |  | 1 | 3 |
| 5 | El usuario solicita crear un reporte dinámico y con orígenes de datos distintos y relacionados a los datos de los formatos.  (usabilidad, escalabilidad, modificabilidad) | 4 | 1 |  |  |  |  |  | 1 |
| 8 | Como Ingeniero Implementador quiero que el software solicite doble autenticación por token  o email con el fin de brindar seguridad a nuestros clientes. |  |  |  |  |  |  |  | 0 |
| 9 | Como proveedor, al momento de incorporar nuevos formatos, estos estén habilitados de forma segura dentro del producto, que se puedan instalar o desinstalar según la adquisición del cliente. | 2 | 1 |  | 1 | 1 | 1 | 1 | 5 |
| 10 | se requiere que la aplicación cargue varias fuentes con un solo clic | 4 |  | 1 |  |  |  |  | 1 |
| 11 | El usuario solicita funcionalidad tenga una autenticación de doble factor. (Seguridad) |  |  |  |  |  |  |  | 0 |
| 12 | Como especialista en regulación quiero que el software tenga la opción de modificación de parámetros ayudando a que pueda controlar el cumplimiento de los reportes que deba entregar a los entes reguladores | 4 |  |  |  | 1 |  |  | 1 |

Resultados votación sobre los atributos de calidad sin enfoque en escenarios

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Atributo** | **Darwin** | **John** | **Oscar** | **Ricardo** | **Omar** | **Daniela** | **Total** | **Prioridad** |
| 1 | DESEMPEÑO | 1 | 1 | 1 | 2 | 1 | 2 | 8 | 1 |
| 2 | ESCALABILIDAD | 2 | 1 |  |  |  |  | 3 | 3 |
| 3 | MODIFICABILIDAD |  | 1 | 2 |  |  |  | 3 | 3 |
| 4 | SEGURIDAD |  |  |  | 2 | 1 | 1 | 4 | 2 |
| 5 | DISPONIBILIDAD | 1 | 1 |  |  |  |  | 2 | 4 |
| 6 | INTEGRACION |  |  |  |  | 1 |  | 1 | 5 |
| 7 | USABILIDAD |  |  | 1 |  | 1 | 1 | 3 | 3 |

1. **ANEXO 1. Tecnologías y tópicos relevantes**

**Del proveedor cloud AWS:**

**IAM**: Administración de autenticaciones para empleados,

**Cognito**: Administración de autenticaciones para usuarios externos

**CIAM**: Administración de autenticaciones para usuarios externos

**S3**: Almacenamiento de objetos.

**KMS**: Almacenamiento de secretos como llaves privadas o claves

**API Gateway:** Autenticación de aplicaciones

**Fargate**: Administrador de contenedores

**Lambda**: Instancias de ejecución sin servidor

**S3 Glacier**: Almacenamiento de objetos gran capacidad y bajo costo

**SQS**: Broker de mensajería simple

**Secrets** Manager: Almacenamiento de credencias de acceso a base de datos

**ElasticCache** for Redis: Almacenamiento en memoria volatil, de alta velocidad para guardar sesiones.

**Cloudwatch**: Herramienta de monitore de hardware y software a nivel de logs.

**De otros proveedores o estándares de la industria aplicables al proyecto:**

**OAuth2**: Protocolo para la gestión de autenticaciones, en especial APIs.

**JWT**: Estándar utilizado por OAuth2.0 con o sin cifrado para los mensajes, en formato JSON.

**Angular:** Framework para el front-end con Typescript, importante no guardar ningun secreto en el Front.

**SAML2**: Protocolo basado en XML para la autenticación de personas entre sistemas externos o expuestos en Internet.

**Key pair:** llaves privada y pública necesarias para la autenticación de las APIs con OAuth2.

**Criptografía asimétrica:** Principio para la gestión de las autenticaciones y algoritmos de crifrado.

**SHA256:** Algoritmo de cifrado utilizado durante las autenticaciones

**SALT**: Mecanismo de generación aleatorio de claves cifradas con SHA256 y dificulta la ingeniería reversa.

**TLS 1.2:**  Protocolo criptográfico que asegura la comunicación HTTPS.

1. **ANEXO 1. Estimación aproximada del desempeño**

La estimación de desempeño se debe considerar en dos partes:

1. El proceso que desee realizar de forma paralela con los servicios Serverless Lambda
2. El rendimiento de la base de datos, de acuerdo a esta tabla de valores base a continuación:

| Amazon RDS almacenamiento por instancia | | | | | |
| --- | --- | --- | --- | --- | --- |
|  | **Tipo** | **HD** | **Tamaño HD** | **In/Out por segundo (IOPS)** | **Rendimiento BD o Throughput por segundo** |
| **SQL Server** | Uso general gp2 | SSD | 1072 GB | 2997 | 262 MB |
| **SQL Server** | Uso general gp2 | SSD | 17592 GB | 16000 | 262 MB |

Fuente: <https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_Storage.html>

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Velocidad de formulación =** | Cantidad de cálculos a realizar | X | Tiempo por cálculo | + | MBs a procesar |
| Cantidad de procesos | Rendimiento HD |

Con la siguiente la fórmula y tomando como ejemplo la formulación, se puede obtener una estimación aproximada de la potencia de procesamiento:

Consideraciones:

1. El procesamiento de un cálculo con .NET requiere 0,2 segundos.
2. Se tiene 10 procesos o funciones Lambda
3. Se desea procesar 1000 MB.
4. Se necesitan ejecutar 1000 cálculos o celdas con formulas.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1000 | X | 0,2 | + | 1000 | = | 23,8167938931298 |
|  | 10 | 262 |  |  |

Ejemplo con cifras de esta fuentes:  <https://filia-aleks.medium.com/aws-lambda-battle-2021-performance-comparison-for-all-languages-c1b441005fd1> ,  <https://filia-aleks.medium.com/aws-lambda-battle-2021-performance-comparison-for-all-languages-c1b441005fd1>

No es posible determinar de forma simple un límite para el máximo de cálculos que se pueden realizar, porque los servicios Lambda son paralelizables hasta 1000 por región (<https://docs.aws.amazon.com/lambda/latest/dg/gettingstarted-limits.html>)  y , cada uno puede procesar como un aplicación-servidor independiente, quizas solo el hardware que siendo utilizado de forma básica y sin paralelizarlo podría llegar a 262MB de transferencia por segundo para un disco de uso general, es decir 15.7 TB/minuto en el mejor de los casos, pero aún esto es mejorable con discos de ultra velocidad que AWS ofrece, con todo esto, no hay restricciones para no paralelizar los discos también cuando los volúmenes de datos son excesivos, porque en los Microservicios, las bases de datos se pueden definir por aplicación y no una única para todos.

La capacidad máxima posible en la infraestructura de AWS es casi ilimitada, su límite radica en la lógica puesta en la implementación .NET y los límites que se deseen poner al hardware por motivos económicos, actualmente empresas como Netflix, Facebook, Walt Disney, SAP, Pfizer, Sony, General Electric y muchas otras más tienen sus servicios desplegados en AWS, ellos fueron los pioneros con este tipo de servicios, incluso proveedores PaaS como Heroku, se basan en AWS.

El objetivo del primer escenario de procesar 100 millones de registros en 5 minutos podría darse de forma aproximada, bajo estas condiciones ideales:

1. 5 minutos = 300 segundos (objetivo)
2. 100.000.000 Registros \* 15 bytes por registro = 1.500 MB
3. 1000 procesos Lambda y un tiempo de cálculo objetivo de 0,0029 segundos.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 100000000 | X | 0,0029 | + | 1500 | = | 295,725190839695 |
|  | 1000 | 262 |  |  |

Preguntas y Respuestas

1. Cuál es el gap y cómo atenderlo .

Actualmente, contamos con un backlog de gaps, donde hay funcionalidades faltantes para cumplir el RFP de NEQUI y funcionalidades importantes para ir cerrando las brechas para alcanzar la arquitectura objetivo.

<https://docs.google.com/spreadsheets/d/1-5sAt6LkorvliINQsrSPGIbFniNcMfPB/edit?usp=share_link&ouid=111550789523792719668&rtpof=true&sd=true>

Comparto el enlace donde se encuentran priorizadas las actividades del backlog.

También se tiene las HU que están en este drive: <https://drive.google.com/drive/folders/1odvRYWrqB32t_fJTLjx2HnEJnSiCbSbG?usp=share_link>

El plan que se tiene de llevar el producto AS-IS a la arquitectura TO-BE, es con una arquitectura transicional, la cual será necesaria para ir extrayendo los diferentes componentes críticos que se tienen actualmente, como:

* Cargue de fuentes\*
  + Validaciones de datos
* Generación de datos de formato
  + Formulación y extracción
  + Validaciones de reglas de negocio
  + Uso de proforma como visualizador de datos
* Generación de archivos de transmisiones
  + Validaciones de reglas de norma o estructura
  + Creación del archivo y adaptado para altos volúmenes de datos.
* Automatizaciones de procesos

\* ya se comenzó con la extracción del proceso de cargue de fuentes a 4 microservicios, el cual han sido un éxito en el sentido de mejoramiento para el cargue de altas volumetrías, 50 millones de registros por 20 columnas de diferentes tipos de datos en tan solo 27 minutos.

Por lo tanto, se deberá ir construyendo alrededor del monolito los servicios que se necesiten para suplir las necesidades de los formatos de alta volumetría y con esto, las estrategias de bases de datos, tanto relacionales como no sql.

Qué tipo de ingenieros necesitamos?

De acuerdo al documento, recomiendan 2 equipos de trabajo con 7 personas cada uno:

* 3 desarrolladores intermedio o senior para backend (.Net 6 Core, Docker y un poco de AWS)
* 2 desarrolladores intermedio o senior para frontend (Angular, javascript, html, css y un poco de AWS)
* 1 ingeniero de pruebas intermedio (Conocimiento básico en programación y enfoque en frameworks de pruebas para backend y frontend)
* 1 lider técnico (conocimiento en .net 6 Core, Angular, pruebas unitarias y automatizadas, DevSecOps y AWS nivel medio o más en todo, habilidades de liderazgo)

En total serían 14 personas y sin contar con una persona en Scrum Master para etapas de alta incertidumbre, este lo marcan como opcional.

para utilizar lo que hizo AS con el módulo de transmisión y llevarlo a una arquitectura cloud,  reutilizarlo, evitando que la respuesta sea la común :”volver a hacerlo”

1. La información de este documento fue contemplada en el documento entregado?

Se mencionó varias veces en las sesiones que se tuvieron, en las observaciones del entregable parcial y en el final, la respuesta fue la siguiente: “Estamos de acuerdo, sin embargo no encontré documentación de este tema en las arquitecturas recibidas, por fortuna se tuvo en cuenta y no encontré riesgos”. Pero en el documento final, no se evidencia algo parecido.

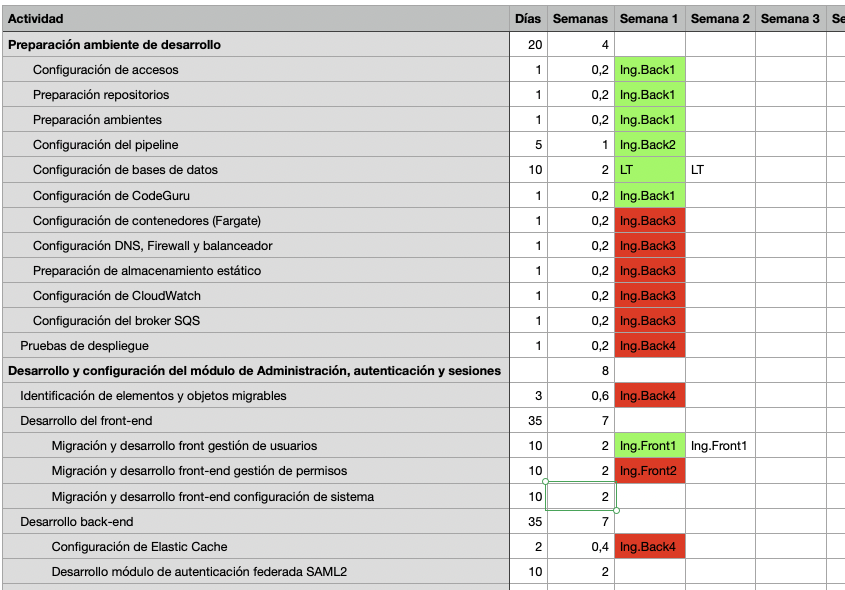
Con esto, habría que agregar otro perfil de desarrollador intermedio o senior para cubrir la tecnología en Python

1. Cuáles son las fases de transición , qué programadores contratar en cada etapa hasta llegar a la arquitectura to-be. Iniciando con los desarrollos del reléase 2. Y siguientes.

Son 3 etapas.

La etapa 1 es pasar la aplicación tal como está y ajustar la conexión a la base de datos que ya no estaría en el mismo servidor, ajustar el balanceador y las políticas de escalamiento, esto lo puede hacer un líder técnico, no se requiere todo un equipo.

Se puede iniciar con equipo menor e ir incrementando en cada etapa, 2 back-end y un front-end + un líder técnico,  solo irían un poco más lento y no se lograría paralelizar el trabajo al máximo, en especial el back-end,  ejemplo(en rojo lo que no se podría ir avanzando):



1. Lenguajes free o de Microsoft  , siempre han sido cuestionados por los grandes bancos debido a aspectos de seguridad.  Qué dice AS al respecto de las tecnologías recomendadas ¿

Por experiencia no es así en la práctica, yo trabajé en BBVA, Data Crédito y Scotiabank Canada, México y Colpatria, también he conversado de estos temas con el líder de arquitectura de ATH, en todos trabajan con nube Híbrida (AWS, Azure, Heroku y Datacenter propio) Java, DB2, framework Spring y AS400 Cobol, al terminar los desarrollos se someten a herramientas de escaneo de vulnerabilidades y pentest con empresas índias, el atributo de calidad número 1 de ellos es seguridad.

Las fallas de seguridad se dan por malas prácticas y desconocimiento, no por los lenguajes o frameworks, la infraestructura provista por estas grandes compañias es lo más seguro que hay.

El modelo planteado permitiría ser SaaS, multiplataforma ¿ para clientes masivos?  Es decir que podamos tener una misma aplicación que se conecte a diferentes bases de datos?

Correcto, solo SaaS los microservicios (etapas 2 y 3).

Cómo reutilizar la formulación en las proformas?, con rendimiento eficiente ,  está previsto en la arquitectura?

Si, de acuerdo a los diagramas expuestos en la documentación, se tienen 2 planteamientos para las fórmulas y sus validaciones

Está prevista la línea de big data y analítica ¿ la arquitectura está describiendo este aspecto y sus componentes?

Si se puede implementar una solución de analítica de forma "fácil" pero no fue una problemática cubierta en esta solución, en esencia sería conectar las bases de datos propuestas a los servicios de analítica deseados y obtener insights, este link podrán observar la lista de servicios de AWS para esto: https://docs.aws.amazon.com/es\_es/whitepapers/latest/aws-overview/analytics.html

Está prevista la configuración autónoma de las salidas de los archivos planos y otros tipos de salidas¿

No estoy muy seguro de entender la pregunta, tampoco fue un caso específico que cubriéramos, pero en el diseño hay margen para implementarlo entre BaaS de Transmisiones y el Gestor o Controlador de transmisiones, quizás se requiera una tabla para almacenar estas configuraciones, funcionalidades similares a nivel general siguen este mismo patrón.

1. Qué indican con respecto a la extracción ¿ se están planteando buses de datos?

No, no es necesario, los buses estaba más orientados a múltiples aplicaciones, su símil son los brokers de mensajería como SQS o Kafka, los cuales solo recomiendo cuando se cumplen ciertos criterios de volumen, concurrencia,  performance y volatilidad de la data.

1. Qué indican en cuanto a DATA QUALITY?
2. Esta parte la conversamos en una reunión y es flexible, de momento se propuso en 2 etapas:
3. i. 1. Validaciones generales localizadas en el BaaS de Carga de Fuentes.
4. 2. Validaciones específicas ejecutadas por el worker de carga de fuentes.
5. Diagrama

   Descripción generada automáticamente
6. Considero que hay 3 categorías en la validación:
7. 1. Seguridad, (toda entrada en el sistema es un punto crítico) se deben implementar las protecciones ante el top 10 de OWASP.
8. 2. Integridad, en este punto existe AWS Data Migration, que compara el original con el cargado en AWS. https://aws.amazon.com/blogs/database/migration-validation-part-2-introducing-data-validation-in-aws-data-migration-service/
9. 3. Validaciones de negocio, las cuales deben estar en los workers de carga de fuentes (casos generales) y posteriormente en los validadores "workers" de la formulación, estos deben verificar las reglas definidas y reportar el error para que el cliente entienda porque no funciona la formulación.
10. Qué indican con respecto al módulo de validaciones y su modernización?

Las validaciones se plantean dentro del documento como 3 componentes; un gestor y 2 validadores. Un validador para las reglas de validación de norma y el otro, para el contenido o reglas de validación de negocio.

1. Está prevista la implementación e incorporación de cambios progresivos en cuanto a UX?

No, el alcance de la arquitectura solo va hasta la definición del backend y frontend.

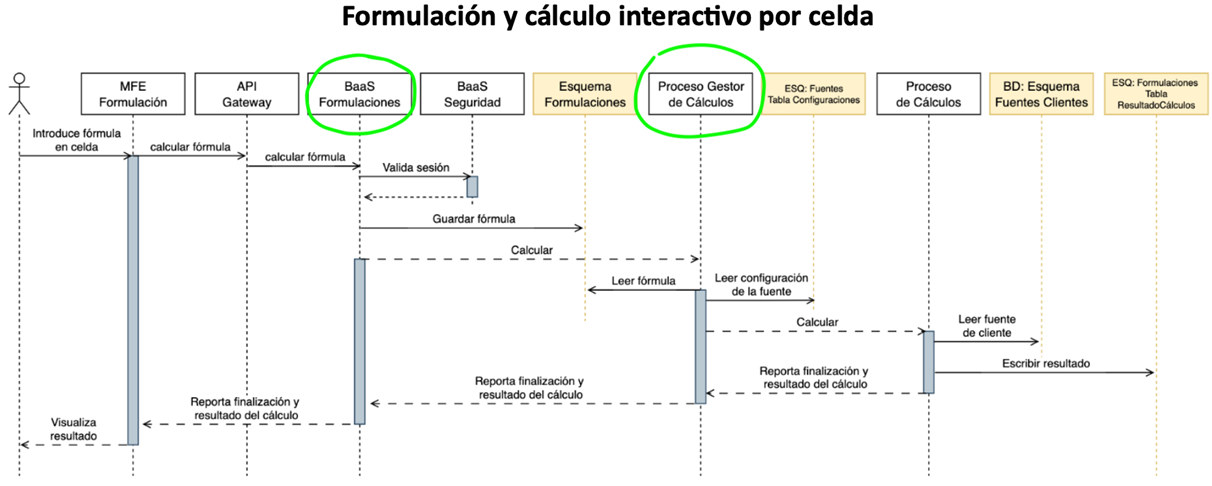
1. Cómo van a convivir los formatos del monolito con los demás que requieren alta volumetría , y microservicios?

El monolito es una opción de mejora al desempeño acutal y no compartirán infraestructura con los microservicios vivirán aparte, el monolito estará dando soporte hasta que los microservicios estén 100% operativos.

1. Hasta donde llegaría la versatilidad de los usuarios con respecto a validaciones y configuración de los formatos y salidas ¿

Están limitados por la cantidad de posibilidades de reglas aplicables programadas para su uso, si por ejemplo disponibilizan la validación de formatos de fecha, solo entonces pordrá se utilizada por los usuarios-reguladores-internos, de lo contrario no habrá mucha reutilización.

Las salidas están más asociadas a los formatos, no se plantearon como algo independiente que el usuario pueda elegir durante o después de formular, si eso se desea, una opción sería implementar la opción de elegir la salida en el BaaS de Formulaciones para que este a su vez, pueda llamar al proceso gestor de cálculos correspondiente a la salida elegida.



La arquitectura descrita permitirá por parámetros la asignación dinámica de recursos y capacidad de máquina?

Este deber típico de soluciones como Terraform ahora está en manos de AWS, los cuales monitorean el uso de recursos basados en criterios definidos por ustedes y así podrá aprovisionar más o menos capacidad.

Cómo dispone la arquitectura planteada el módulo de tax y el módulo de tótal report de manera independiente ¿

De acuerdo con lo mencionado por el arquitecto, los servicios de tax se pasarán tal cual a esta arquitectura. Será decisión de Project House realizar su migración a microservicios y procesos gestionados por la nube.

Finalmente el documento entregado es amplio y suficiente para que el equipo de desarrollo pueda guiar su trabajo y empezar con los trabajos de modernización del core TÓTAL Report ¿

No, está descrito a alto nivel. Según el arquitecto, no daba el tiempo para entrar en detalles pues se volvería un documento interminable.

Por favor abrir un espacio en un drive interno que permita que las versiones queden allí almacenadas y no pase como en la versión anterior que se perdió la trazabilidad de los documentos por tenerlos en discos fijos o quedaron traslapados en correos.

Comparto la url de la unidad compartida por Andrey para el almacenaje de la documentación referente a la arquitectura: <https://drive.google.com/drive/folders/0AK7GlKEpdc6HUk9PVA?usp=notify_sd_md>

La arquitectura definida, herramientas, y otras características técnicas definidas en el documento de Asesoftware, están orientadas y soportan implementaciones multi-cloud?

Si, los nombres de los componentes pueden variar entre uno y otro proveedor cloud. Pero a la larga todas las nubes ofrecen lo mismo. Unos quizás más costosos que otros.

Comentarios 27/04/2023

Hay confusión en los términos que maneja el arquitecto con respecto a la Transmisión, en múltiples secciones del documento, se habla de: migración, transferencia. Por favor organizar este termino en todo el documento. Resuelto

Cuando se habla del módulo TOTAL Tax, no se observa ninguna referencia a los servicios en las etapas de la arquitectura.

Se agregó el numeral 5.4 para dar respuesta.

En el escenario ESC\_003 se habla sobre la automatización de procesos, pero en la definición de la arquitectura (diagramas) no se vuelve a mencionar esta necesidad. Se agregó el numeral 4.4.6 para dar respuesta.

En la vista de contexto (numeral 4.1) se habla sobre el actor “Entes de Control”: Es interesante la propuesta, pero hay que tener precaución de aquellos archivos de transmisión que deben ser firmados previamente por el usuario final de la entidad vigilada. Por lo que se tiene en consideración que esta situación es para el 80 o 90 % de los formatos. Por lo tanto, esto solo aplicaría al 10% de formatos que tenemos actualmente (Conglomerados y SAC) - Este diagrama solo indica qué estará dentro de la aplicación y qué no.

En la vista general de la solución (numeral 4.2) el arquitecto propone la exposición de apis como mecanismo de acceso para otras aplicaciones (integraciones),

No le da mucha transcendencia en la arquitectura como propuesta. Se han agregado los detalles relevantes en la sección 4.7, dado que ya se describe la estructura de los microservicios, el flujo de los más relevantes y los componentes que intervienen, los detalles del desarrollo como endpoints, firmas, especificación etc. Se deben resolver basado en los estándares actuales.

Que mecanismo de seguridad se estaría manejando?

R: para la comunicación del front-end al back-end Oaut 2.0 es el recomendado y gestionado a través de AWS API gateway.

En el numeral 4.3.2, el flujo que describes no tiene en cuenta la generación del archivo de transmisión como un proceso importante, sino que aparece como un proceso secundario. Se agregó un nuevo diagrama para resolver este punto.

En el numeral 4.3.3,

el flujo va hasta la visualización de la generación del formato o va hasta la transmisión de este? Visualización

Las generaciones no serían persistidas en el sistema, pues serían procesos de configuración y esto debería no ser persistido para no dañar el record de la cuenta del cliente. Esta decisión sigue en manos de PH, deben tener en cuenta de que el costo de almacenar la generaciones es muy bajo.

Cuando se habla de contenedores, se refiere a el uso de Docker? Si

No se especifica el uso de los lenguajes de programación, de cara al backend. Se hace referencia a Angular o React como frameworks para el front-end. Debe ser .NET 6+, se agregó recomendación en el punto 4.7

Con la carga de archivos planos y exceles como fuentes de orígenes, esto como se vería reflejado en la arquitectura, y a esto se le suma el peso por la volumetría, que mecanismos se utilizaría para completar la tarea? Carga por el front-end

Se podría pensar en una VPN para el paso de archivos de este calibre? No es necesario

Se podría pensar un FTP o SFTP para este propósito? No es necesario

No se tiene claramente como será el mecanismo de visualización de los datos de formatos y archivos de transmisión cuando estos superen los 100 millones de registros. Sería importante rescatar lo que se hizo en la arquitectura anterior liderada por Diego Soler donde destacan un visor por lotes. No hay un requerimiento específico de este punto identificado hasta ahora, la visualición no es un desarrollo interno sino el uso de una solución de tercero.

Con la parte de DevOps, en caso de detectar errores, este mecanismo permitiría conectarse con JIRA o con que herramienta se adaptaría para el control de incidentes?

La identificación de errores de ejecución se vería en Cloudwatch.

Los errores de pipeline o devops en Code Catalist.

No se tiene claro los lineamientos de desarrollo de los microservicios, a nivel de herramientas y a nivel de política de desarrollo: Este punto está resuelgo a nivel general en Modelo de Entrega.

No se indica el uso de un Discovery Service.

No se requiere en este diseño, el concepto aplica más para otra disposición de escalamiento y arquitectura por eventos tipo publicador suscriptor.

Patrones de OpenTelemetry para aplicar la observabilidad o con CloudWatch lo remedia? Con Cloudwatch

Patrones como HealthCheck para el estado de salud de los servicios.

El monitoreo está a cargo de CloudWatch y la notificación externalizada,

Para servicios ya existentes y que están en SOAP/WFC (servicios TAX), que habría que hacer para que fueran compatibles? No se deben cambiar, el numeral 5.4 lo explica.

Herramientas como un Vault para el control de las contraseñas. Se recomiendan dos, KMS para llaves de los servicios y AWS Secrets Manager para las bases de datos.

Que riesgos se podrían mitigar si no se adopta una política de desarrollo, pues se entiende que en la nube el costo de operación se hace más elevado si no se tiene una buena codificación. ¿Quién controlaría esto? En la nube se paga por el uso en la mayoría de los casos, la administración de costos se puede llevar a cabo mediante el administrador de AWS el cual discrimina el costo.

Que posibles arquitecturas tendrían cada microservicio. La estructuar interna de cada microservicio va en función de la funcionalidad que presta, en la mayoría de los casos pueden tomar conceptos de MVC.

Que estrategias se podría utilizar como el patrón CQRS o DDD, se manejaría ORMs, Patrón Repository con UnitOfWork. Se tiene contemplado un módulo de presistencia en las aplicaciones back, no se estima que se requiera articular queries entre varios servicios, esto genera el riesgo de acomplamiento.

Se debe seguir con LINQ to SQL como ORM con Repository, implementar otros patrones más complejos deben ser para formatos específicos, CQRS se recomienda para arquitecturas por eventos. El patrón intrínseco en la arquitectura es sagas tanto orquestadas como coreográficas.

Que se usaría para conectar varios microservicios, un bróker de mensajería, o que estrategia se podría aplicar?

La conexión entre servicios está definida en el punto 4.7

En el front-end, que estrategias se aplicarían si la aplicación crece mucho y su compilación hace que sea imposible de efectuar?, Dado el tamaño de la aplicación, no se estima que pueda crecer mucho el front-end, la compilación o transpilación para el front estaría a cargo de AWS Code Catalyst.

Para las bases de datos de las fuentes estándar y fuentes formato, no es mejor manejarlas con una base de datos NO relacional y distribuida para la mejor ingesta frente a altos volúmenes de datos (mas de 1 mil millones de registros) esto no haría parte de la definición de esta arquitectura. Desde el punto de vista del desempeño pasar a RDS es lo correcto, tiene una capacidad de 40 millones consultas por segundo, comparativa: <https://dynobase.dev/dynamodb-vs-amazon-rds/>

Segundo, la data de los formatos es estructurada y jerarquizada, características que las bases de datos relacionales manejan mejor.

No se ve muy claro el patrón multi-tenant, sería bueno darle más protagonismo.

La temática no es compleja, la arquitectura está pensada para multi-tenant, desafortunadamente tampoco era el enfoque principal del trabajo ni se tuvo tiempo para abordarlo.

Tampoco se observa los medios de notificación que se usarían para describirle al usuario el proceso que está llevando, workers, barras de progreso, mensajería y alertas dentro de la aplicación, etc.

Este es un detalle de la implementación, que se debe manejar a través de las aplicaciones propuestas porque es una propiedad más que una nueva capacidad, igual que el anterior no se pudo resolver cada uno de los detalles, estos deberán abordarse durante la implementación o la asesorías.

Comentarios 02/05/2023

De acuerdo a las respuestas a los comentarios de la semana anterior:

Era importante haber tenido presente el trabajo que se hizo sobre el transmisor y visor hecho por el anterior arquitecto y adaptarlo (actualizarlo) para esta nueva arquitectura. No se puede perder este trabajo.

(Estamos de acuerdo, sin embargo no encontré documentación de este tema en las arquitecturas recibidas, por fortuna se tuvo en cuenta y no encontré riesgos)

No estoy de acuerdo con el uso de LINQ to SQL como ORM para los microservicios, pues hemos sabido que dicho ORM solo funciona con SQL Server.

No pretendo convencerlos, pero yo no veo un riesgo o incompatibilidad y de forma muy resumida es por lo siguiente :

**Todo en la base de datos.**

Rendimiento actual = Lectura DB x Cálculo formulas DB x Escritura DB

Rendimiento actual ejemplo = 3seg. x 100seg. x 6seg. = 1800 seg.

**Paralelismo en la lógica y la BD solo para escritura y lectura.**

Rendimiento nuevo = Lectura DB x Cáculo formulas Workers /(cantidad de workers) x Escritura DB

Rendimiento nuevo ejemplo = 3seg. x 100seg./10 x 6seg. = 180 seg.

La latencia generada por el ORM solo se aplica a los gestores de procesos, los workers trabajarán con el contector de AWS, también debes recordar que Microservicios te brinda libertad en cada aplicación, lo que ofrece suficiente libertar para implementar diferentes estrategias de persistencia y bases de datos.

No me convence mucho la respuesta sobre la base de datos para la fuentes estándar y fuentes de formato. En nuestro roadmap ya están identificadas varias funcionalidades que se deberán trabajar con bigdata y RDS for SqlServer no creo que sea la mejor opción para este tipos de estructuras.

Igualmente, no pretendo convencerlos, mi análisis me indicó que continuar el trabajo con datos estructurados, no está mal y les ahorraría mucho esfuerzo que tendrían que hacer si se cambian a otro paradigma, el costo beneficio no me indicó que un cambio así los llevaría a cumplir los objetivos, BigData es un concepto muy amplio, no se realmente cómo lo están analizando.

Ahora, con lo que entregaste como documento final:

Fue un trabajo interesante lo que planteaste, es bueno que hayas acatado la mayoría de los comentarios que se te solicitó para esta entrega final.

No obstante, aún queda el sinsabor de como aplicar esta arquitectura, no sé si otro arquitecto diferente, sea capaz de interpretar lo que se plantea en este documento.

A nivel técnico, siguen siendo confuso los términos de serverless y de cómo esto nos ayudará a cargar un servicio o un fragmento de código, no me es muy claro.

Para el documento del AS-IS

No tengo observaciones.

Para los documentos tipo acta de las asesorías

Considero que la información suministrada en las actas de las sesiones de asesoría, queda mucho por desear, se supone que en estos documentos solo está los temas tratados en la sesión, pero nunca se trataron los requisitos que se solicitaron al inicio de la asesoría, como por ejemplo:

Selección de tecnologías y sus posibles riesgos

Metodología de trabajo sobre el desarrollo de los requerimientos.

Lineamientos de entrega o integración hacia la Arq.ToBe

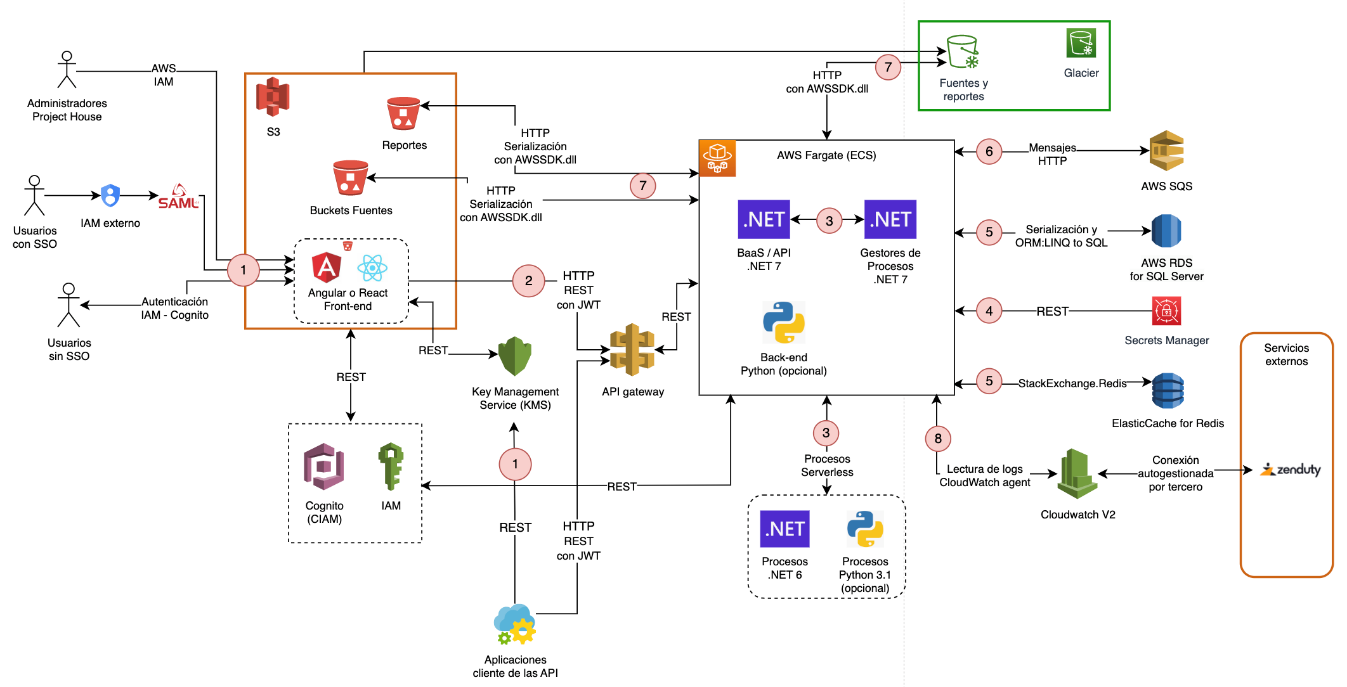
Definición de equipo para el desarrollo de los requerimientos.

Es importante que revise esto, pues se mencionó al inicio de la asesoría que se tuvo el 24 de abril.

Considero que el contenido del acta de la ultima sesión no concuerda con lo manifestado inicialmente por nosotros. Solo se menciona los puntos revisados de la nueva arquitectura pero no está acorde a lo solicitado como asesoría.

Tecnologías con las que se espera trabajar con el fin de alinear los nuevos desarrollos con el objetivo de arquitectura.

El stack tecnológico sugerido es el siguiente (diagrama tomado de la arquitectura):



**Del proveedor cloud AWS (extrapolable a otros como Azure o GCP):** IAM, Cognito, CIAM, S3, KMS, API Gateway, Fargate, Lambda, S3 Glacier, SQS, RDS for SQL Server, Secrets Manager, ElasticCache for Redis, Cloudwatch.

**De otros proveedores:** .NET 6 y 7, Angular, Docker, SonarQube, conectores (LINQtoSQL y AWSSDK.dll), REST y XML Protocolo SAML2.

Metodos, procesos y/o metodologías que permitan una integración de los nuevos requerimientos de manera que se mantengan los atributos de calidad para un software en producción.

Las metodologías de desarrollo de software son 2, Kanban y Scrum

Kanban: Para las etapas con definición clara como lo es el plan de trabajo entregado.

Scrum: Para la migración de formatos con mayor incertidumbre.

Metodología de versionamiento recomendada: Git-flow

GIT: Para el código fuente

LiquidBase : Para el versionamiento de la base de datos.

Verificación y observancia del correcto de las ramas por parte del líder técnico.

Verfificación del cumplimiento de la calidad y buenas prácticas por parte del líder técnico.

El pipeline deberá incorporar:

1. Permitir hacer push a la rama correspondiente al cambio

2. Validar el cumplimiento de la cobertura de las pruebas escritas en el código del 80%, calidad de código y comentarios.

3. Ejecutar las pruebas

4. Recibir validación y aprobación del líder técnico para hacer el build.

5. Permitir el merge.

6. Realizar el despliegue.

Posterior al despliegue en ambiente de pruebas:

1. El desarrollador debe probar sus cambios en la nube.

2. Debe remitir su cambio a revisión por el QA funcional (QA interno) y este realizar las pruebas de certificación.

3. El cambio debe quedar en una lista para posterior liberación para pruebas del equipo de negocio.

Luego de confirmación por parte de negocio de que los cambios son correctos, se confirma su paso a producción.

Recursos humanos (tipos de profesionales) que deberían considerarse para tal finalidad.

En concordancia con lo mencionado en el documento de arquitectura se recomienda:

2 equipos de trabajo con 7 personas cada uno:

3 desarrolladores back-end (Con conocimiento término medio a alto de .NET y conocimiento básico de AWS y contenedores)

2 desarrolladores fron-end (Con conocimiento medio a alto de Angular, Javascript, HTML y CSS, más conocimiento básico de AWS)

un ingeniero de pruebas (Perfil con conocimiento básico de programación y enfoque en frameworks de pruebas front y back ej: Karma, Jasmine, Protractor, Selenium, Mocha, Cucumber, Jacoco)

un líder técnico (Conocimiento en: .NET, Angular, pruebas unitarias y automatizadas, DevSecOps y AWS nivel médio o más en todo, más habilidades de liderazgo)

Scrum master (opcional, solo para etapas con alta incertidumbre o formatos

Consideraciones a tener en cuenta para mitigar riesgos sobre los nuevos requerimientos.

Las consideraciones están dadas desde distintas perspectivas, ejemplo: seguridad, desempeño, usabilidad etc. el documento de arquitectura ya cuenta con múltiples recomendaciones y no hay forma de responder de forma resumida a esta pregunta porque los frentes de trabajo son todos y todos tienen consideraciones.

Sin embargo, el riesgo presente que observo en varios proyectos son los cambios de tecnologías en la implementación, muchos desarrolladores creen que las soluciones se deben basar en lo que una nueva tecnología promete y no saben que se debe solucionar de forma inversa, es decir, basado en mi problemática, qué tecnología se adapta mejor a mí (mi capacidad de desarrollo, conocimiento, requerimientos no funcionales, compromisos derivados), por ello se debe tener máximo cuidado y llevar a cabo cambios con razones objetivas y sustentadas, teniendo en cuenta que mañana será necesario dar soporte a una tecnología adicional y quizás no todos la conozcan.

1.   Mencionar la compatibilidad del diseño en otros proveedores de nube (*MultiCloud*) a parte de AWS.  La explicación se agregó en el punto 4.7.4

2.   Especificar cómo la arquitectura está pensada para trabajar de manera *SaaS* y, además, que pueda soportar múltiples clientes (*MultiTenant*). Se agregó la explicación en el punto 4.4

3.   Mencionar cómo con la arquitectura planteada, a través de la solución desplegada se *permitirá gestionar los recursos tecnológicos dispuestos* a fin de que se puedan asignar dado el caso a un formato o proceso. Se agregó la explicación en el punto 4.7.5

4.   Indicar actividades identificadas por el arquitecto donde se describan los módulos o procesos críticos. Se agregó el angexo 6.4

5.   Especificar las modificaciones a realizar para seguridad de acceso, esto dentro de la fase de transición de la aplicación hasta llegar a la TO-BE. La explicación se agregó en el punto 4.8

6.   Especificar los expertos que se necesitan por cada fase, desde la fase inicial hasta llegar a la TO-BE, para poder contratar al personal idóneo por cada fase y no durante el proceso completo.

Este tipo de detalles de la implementación no deben estar en el documento de arquitectura.

La lista de personas sugeridas previamente en esta cadena de correo está bien para la construcción, sugeriría adicionar a alguien de negocio que entienda bien los requerimientos y los pueda explicar al equipo.

Opcionalmente en las empresas se tienen personas específicas para apoyar estos proyectos o su mantenimiento y operación:

Arquitecto de software o soluciones (Etapa de construcción)

Ing. DevSecOps (Etapa de construcción)

DBA  (Construcción y mantenimiento)

Infraestructura (Construcción y mantenimiento)

Seguridad (Operación, mantenimiento y despliegues)

Definiciones de equipo por fases en el punto 5.3

7.   Especificar la necesidad de actualización del componente de transmisión de la aplicación y los riesgos que este cambio conlleva.

Esta solicitud implica un esfuerzo de análisis mayor que no estaba dentro de los objetivos y escenarios que se pidió inicialmente resolver. El componente de migración construido en 2019 se mencionó de que no se utilizaba por su complejidad, por ello contar con él, adaptarlo y transicionarlo no era recomendable, ya que agregaba complejidad incensaría a una parte del software que se resolvió de una nueva y más efectiva forma.

Definición de la aplicación de transmisión hecha en 2019, esta fue agregada como restricción en el punto 1.6 y resuelto en el punto 4.4.5.

8.   Especificar la volumetría que permitirá manejar la aplicación con la arquitectura TO-BE propuesta. Se agregó la explicación en el anexo 6.5

He agregado la explicación adicional del desempeño al final del documento, en el punto 6.5.