Consorci
Administració Oberta
de Catalunya

LOT 2 - Arquitectura EACAT 3.0

Documentació d'Arquitectura

Novembre 2021

opentrends

Full de Control

Títol	**Documentació d'Arqutiectura**		
Autor	Autor document		
Versió	vxx.xx	Data Versió	dd/mm/aa
Revisat/Validat per	Revisor	Data Revisió/validació	dd/mm/aa
Aprovat per	Aprovador	Data Prevista Aprovació	dd/mm/aa

Registre de Canvis

Versió	**Comentaris**	**Autor**	**Data**
xx.xx	Comentaris	Autor	dd/mm/aa

Índex

1. Introducció

1.1. Propòsit

Aquest document serveix per a donar un marc de desenvolupament al projecte **EACAT 3.0** explicant les decisions d'arquitectura preses**.** El projecte consisteix en proveir una plataforma de tramitació Inter-administracions.

1.2. Abast

El document es centra en una visió global de la solució reflectint en major mesura totes les decisions d'arquitectura preses.

1.3. Document dirigit a

A totes les persones amb perfil tècnic del projecte, ja siguin Arquitectes, Enginyers de software, Desenvolupadors, etc.

2. Definicions, abreviacions i acrònims

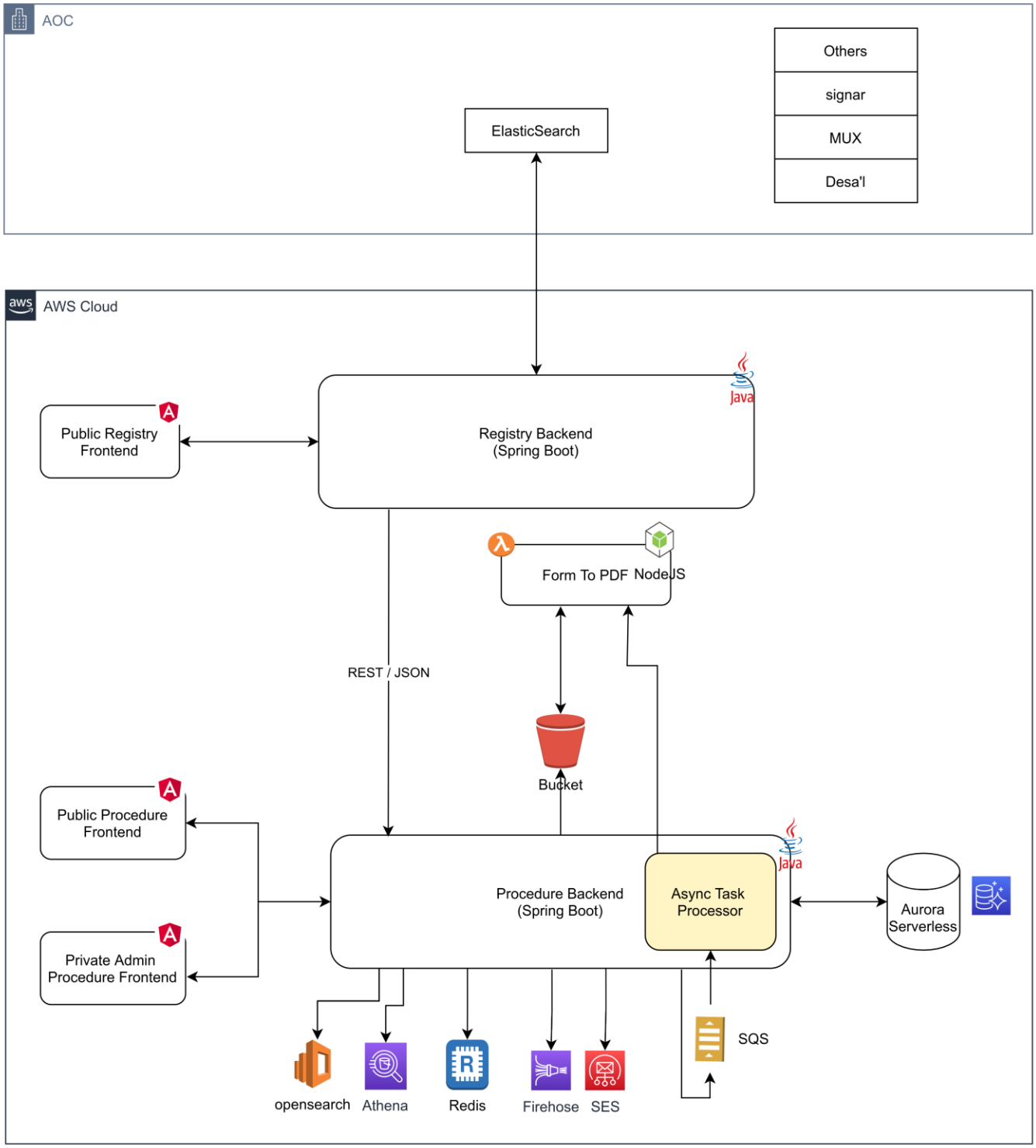
Concepte	Descripció
AWS	
CDN	
Serverless	
API	
JSON	

3. Visió general

Una de les principals característiques d'aquest projecte es que es desenvolupa al núvol d'**AWS** fent servir els seus serveis per a proporcionar robustesa i rapidesa al desenvolupament**.**

3.1. Vista lògica

En el següent diagrama es presenta els principals mòduls detectats:



On es distingeixen els següents mòduls:

- **Procedure Backend**: Backend desenvolupat en java basat en Spring Boot. El qual te la lògica de negoci necessària que dona suport als frontals tant públic com privat de la part de tramitació.

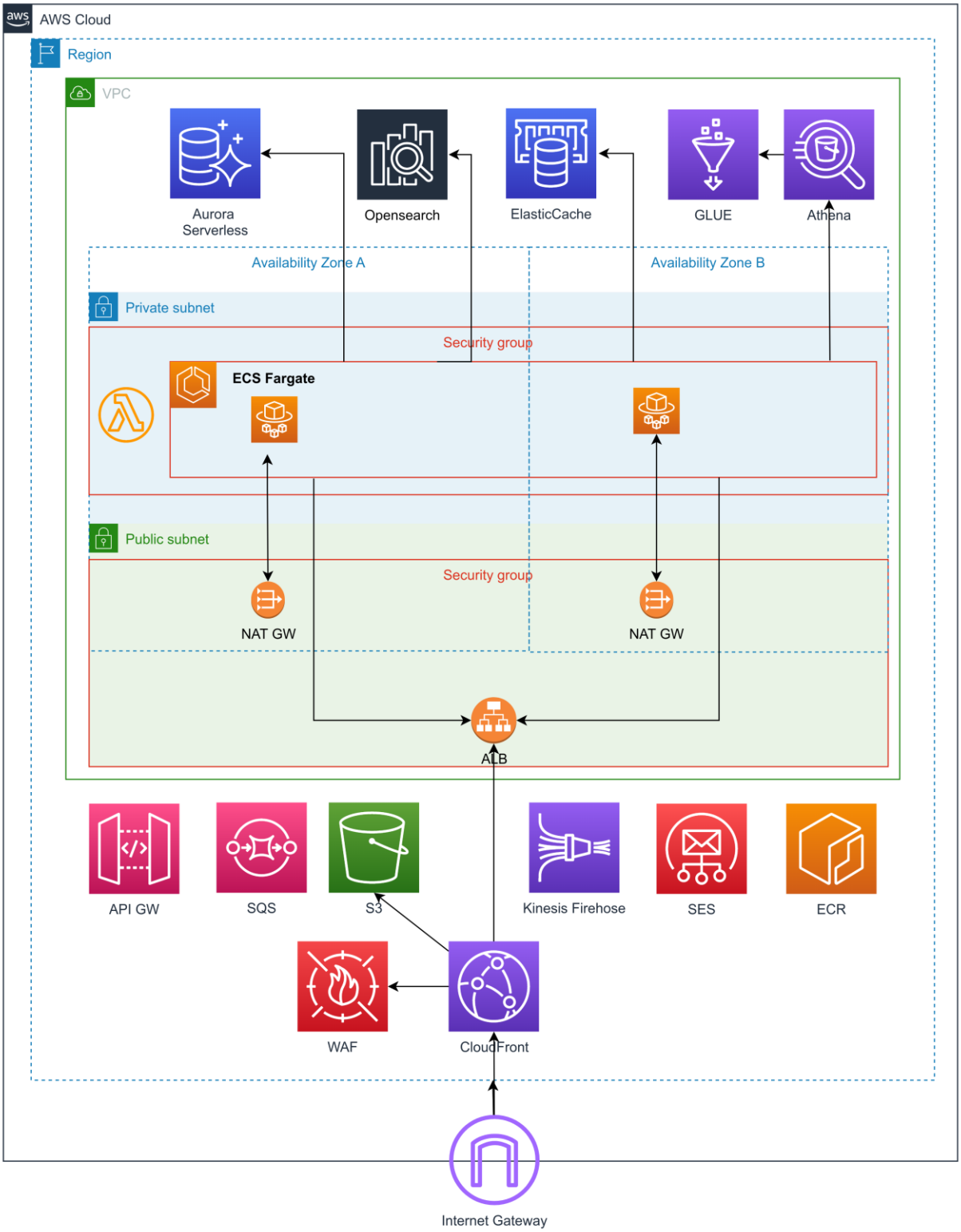
- **Async Task Processor:** Motor de tramitació amb totes les integracions necessàries amb l'**AOC**.
- **Public Procedure Frontend:** Frontend públic de tramitació, on entren els diferents **ENDs.** Un desenvolupament basat en Angular.
- **Private Admin Procedure Frontend:** Frontend privat d'accés restringit a **l'AOC.** On es fan tots els manteniments necessaris de la plataforma. Un desenvolupament basat en Angular.
- **Aurora Serverless**: Base de dades Aurora Serverless amb la seva versió **PostgresSQL.** Que fa de repositori relacional de les dades. La seva versió serverless fa que la mida dels servidor s'ajusti en funció de la carrega de forma automàtica. Durant la nit la BD es redueix al mínim estalviant costos.
- **Form to PDF**: Funció lambda que aixeca un Chrome e imprimeix els formularis **HTML** en **PDF**. Les llibreries que proporciona google per a fer aquestes tasques estan en **NodeJS**.
- **Public Registry Frontend:** Frontal que dona accés a les pantalles del registre. Desenvolupat en Angular
- **EACAT Registry Backend:** Backend que dona suport al frontal del registre per accedir al ElasticSearch. Desenvolupant en java i basat en spring boot.

Els serveis **d'AWS** que es fan servir son**:**

- **Redis:** Es la cache per emmagatzemar dades que canvien molt poc o que necessiten ser consumides molt ràpidament. Com:
  - Bústies dels **ENS**.
  - Definició de Serveis i tràmits.
  - Etc.
- **Firehose**: Cua especialitzada que emmagatzema les dades en un bucket de **S3** que després pot ser consumit per el **Athena.** Aquesta cua es utilitzada per emmagatzemar les dades d'auditoria de l'aplicació i per el manteniment del **SES,** enviament de correus, per saber si els enviaments han anat correctament o no.
- **SES:** Servei d'enviament de correus d'**AWS.**
- **SQS:** Cua permet implementar tasques de forma asíncrona, reintents, etc.
- **S3**: Emmagatzemat temporal de fitxers necessari entre els diferents mòduls de la solució.
- **Opensearch:** Aquest servei es necessari per a poder fer cerques en el catàleg amb full text.

3.2. Visió d'infraestructura

L'esquema general d'infraestructura seria el següent:



On:

- **WAF:** Es el Firewall que entre altres coses permet una defensa contra atacs **DoS, XSS** etc.

- **Cloudfront:** Es el **CDN** d'**AWS**, dona la funcionalitat de cache de tots el continguts estàtics de les aplicacions, així com dona accés a tots el continguts dinàmics dels diferents mòduls.
- **API GW:** Actua com a punt d'entrada de **API**s per a sistemes **serverless** ja siguin serveis REST JSON com a **Websockets**
  - Servei d'impressió en PDF
  - Servei de consum de cache's
  - Servei de notificacions de tràmits entrants (**Websockets**?)
- **SQS:** Simple Queue Service. Es un servei de cues que facilita:
  - El processament de forma asíncrona.
  - Separa el emissor del receptor.
- **S3:** Buckets per emmagatzemar fitxers estàtics de la web, fitxers d'auditoria. Etc
- **Kinesis Firehose:** Es una cua especialitzada que emmagatzema els registres **json** en fitxers en un bucket s3. Necessari per emmagatzemar els logs d'auditoria.
- **SES:** servei d'enviament de correus.
- **ECR:** Repositori de contenidors docker.
- **ALB:** Balancejador a nivell d'aplicació.
- **NAT GW:** Nat gateway. Permet fer nating entre la xarxa publica i la privada.
- **ECS Fargate:** Sistema de control de containers escalable amb conjunció de Fargate que permet no tenir que gestionar el servidors físics.
- **Lambda**: Funcions no lligades a servidor per a tasques com la de generar un PDF fent servir el navegador Google Chrome, etc.
- **Aurora Serverless:** Servei de base de dades auto-escalable el qual permet definir una configuració mínima i màxima de capacitat. D'aquesta manera quan hi ha poca carrega la BD es fa petita i quan hi ha molta demanda creix.
- **ElasticCache:** Servei de cache de AWS. La opció contemplada es un **Redis**.
- **Glue i Athena:** Servei de ETL i Consulta SQL d'informació emmagatzemada en buckets en format **JSON**
- **Opensearch**: Permet fer cerques "full text" al catàleg.

4. Mòduls de la solució

La solució es divideix en dons desenvolupaments ben diferenciats

- **Mòdul d'accés al Registre**
- **Mòdul d'accés a les tramitacions**

4.1. Mòdul Registre

El mòdul del **Registre** dona accés a tots els usuaris de l'**AOC** a veure els registres d'entrada i sortida que es fan entre totes les aplicacions. Ex: **ETAULER**, **EACAT**,
Etc. (Veure documentació funcional per a mes detalls)

El repositori de dades es un **ElasticSearch** situat a la xarxa de l'**AOC** i es alimentat per els servei de **MUX.** El rol d'aquest mòdul es únicament de consulta de les dades, ni les organitza, ni transforma de cap manera.

L'**AOC** no proporciona serveis d'accés al **ElasticSearch.** Per tant el mòdul accedeix directament al **ElasticSearch** amb el seu **API.** Es considera que aquest mòdul es part de visualització de la aplicació de Registre i per tant pot accedir directament al **ElasticSearch.**

Consideracions:

- No es proporcionen serveis de creació de contingut.
- No es proporcionen serveis de consum de l'informació mes enllà de l'interficie gràfica.
- No es proporcionen cap mecanisme de manteniment de **l'ElasticSearch.**
- Es un mòdul merament de consulta.

Al ésser un mòdul independent, l'autenticació es fa mitjançant el servei ja implementat a **l'AOC** basat en **l'OAuth** 2**.** L'autorització anirà a càrrec del mòdul de tramitacions (**EACAT 3.0**) .

Per el desenvolupament es fa servir el framework SpringBoot escrit en java. (Veure matriu de desenvolupament) per la part de backend i Angular per a la part de frontend.

En un principi no es necessita cap servei especial proporcionat per **AWS**. Ni base de dades, ni buckets, etc.

4.2. Mòdul Tramitacions

El mòdul de tramitacions té les següents responsabilitats:

- Proporcionar les funcionalitats de tramitació de l'**AOC** amb el seu manteniment.
- Proporcionar l'autorització d'accés per el resta d'aplicacions l'**AOC.**
- Repositori d'usuaris.

Per l'implementació del mòdul s'han pres les següents decisions de disseny.

4.3. Auditoria / log

A l'aplicació **EACAT 2.0** l'auditoria / log es fa a la base de dades amb un volum de uns 40M de registres. Aquestes taules només son de consulta i no es una consulta exhaustiva amb el que fa engronsar la base de dades innecessàriament.

Al **EACAT 3.0** el control d'auditoria / log es treu de la base de dades per a posar-lo en buckets **S3** mitjançant **kinesis firehose** i poder ser consultats mitjançant **Athena.** El **S3** ja està pensat per emmagatzemar aquest tipus d'informació amb grans volums. (https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html)

4.4. Particionament de taules

El volum de registres en algunes taules de l'aplicació actual creixen a un ritme de 2M de registres anuals i augmentant. Per el tipus de negoci que es manega, els registres creats fa dos o mes anys no son utilitzats o la seva utilització es anecdòtica. Particionar aquestes taules per la data de creació del registre, mantenint en particions petites els registres mes utilitzats i en grans els més antics pot augmentar considerablement la velocitat d'accés i creació dels registres. Amb aquesta tècnica la base de dades pot mantenir un històric molt elevat de dades.

Els candidats per fer servir el particionament de taules son les entitats de instancia de tramis e instancia de serveis.

El servei d'**Aurora Serverless (PostgreSQL)** permet el particionament de taules de forma nativa.

4.4.1. Historificació de les bústies

Per evitar que les bústies creixin de forma desmesurada i accelerar l'accés a les dades més recents, el tamany de les dades les bústies estan acotades a emmagatzemar només les dades X temps.

Aquesta historificació s'emmagatzema en un bucket **S3** mitjançant **kinesis firehose** per a poder ser recuperat si es necessari en un moment donat mitjançant **Athena**.

4.5. Emmagatzemament de formularis.

Els formularis **HTML** generats per al motor de formularis es representen mitjançant **JSONs.** Aquest **JSON** son dades no relacionals i per tant no te sentit guardar-les en la base de dades relacional. Aquesta informació lligada a un tràmit s'emmagatzema en buckets **S3** amb una estructura especifica per a poder cercar per el identificador de tràmit.

4.6. Cache d'informació

Les entitats que no canvien molt sovint, com son serveis, tipus de tràmits, modificació de bústies, etc, son candidats per a ser emmagatzemats en una memòria cau i no anar-los a cercar a la base de dades.
Com a memòria cau es farà servir el servei de **Redis** que proporciona **AWS.**
Aquesta cache s'invalidarà per defecte cada X temps per no crear fuites de memòria o quan aquestes entitats canviïn.

4.7. Comunicació via email

La comunicació via email amb el servei **SES** d'**AWS**, està subjecte a quotes de volums de missatges i missatges per segon. Per tal de controlar que en tot moment s'està dintre d'aquestes quotes es necessari encolar els missatges en una SQS y anar-los enviant controlant els missatges per segon amb un algorisme de **backoff.**
Per assabentar-te de si un missatge ha anat be o malament, el servei **SES** et proporciona dos mecanismes:

- Mitjançant el **cloud watch** on es pot veure una estadística dels nombre de missatges que han anat be o malament.
- Emmagatzemant el resultar en un S3 per a poder consultar els resultats mitjançant **Athena**. En cas d'error es pot saber la causa. Servidor banejat, direcció d'email incorrecte, etc.

Per a tenir un control mes adient del enviament de correus es fa servir el emmagatzemament del resultat en **S3.**

4.8. Impressió de formularis

La impressió de formularis es fa aixecant un navegador Chrome e imprimint en **PDF** tota la pàgina que conté el formulari. Això es fa mitjançant una funció **lambda** en **NodeJS** ja que les llibreries per fer aquesta tasca proporcionades per google estan en **NodeJS**.

4.9. Async Task, execució d'un tràmit

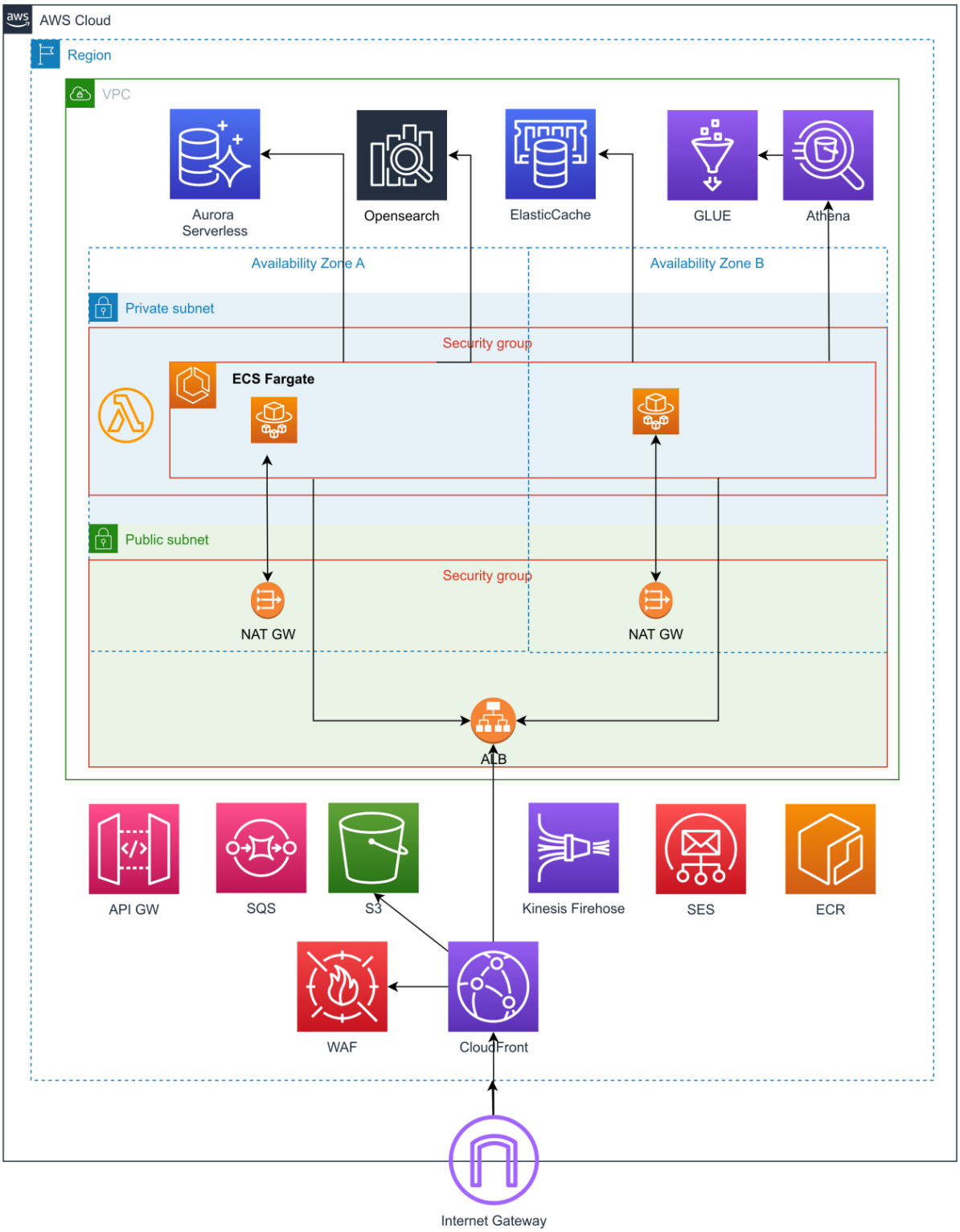
La execució d'un tràmit pot ser un procés lent quan hi ha una sobre carrega en els sistemes implicats com mux, signador, etc. Es per això que aquest procés s'ha d'executar d'una forma asíncrona.
Per poder executar aquesta tramitació de forma asíncrona i distribuïda es fa servir el serveis de cues **SQS** d'**AWS.** De tal forma que quan s'envia a tramitar, un job descrit en json es fica en el **SQS** i qualsevol instancia del backend el processa.

Per ser tolerant a fallides, tots els jobs han de ser idempotents, de tal forma que es puguin executar més d'una vegada amb el mateix resultat sempre.

4.10. Cercador de catàleg

La principal característica del cercador del catàleg es poder fer-ho per contingut. Per això es fa servir el opensearch, un servei d'**AWS** . Al catàleg hi han dos vistes, la del **ENS** i la general on es veu tot sense restricció. La complexitat d'aquest sistema es mantenir sincronitzats les dues bases de dades, la del **Aurora serverless** i la del **opensearch.** Ja que tots els canvis d'**ENS** en un servei, tràmits i demes es tenen que replicar al opensearch.

5. Infraestructura requerida de la solució



En el apartat de mòdul de l'aplicació s'han descrit la necessitat de la majoria de serveis d'**AWS.** Però en aquest apart s'explica la resta d'infraestructura necessària.

La infraestructura es divideix en dues subnets, una publica on bàsicament tenim el balancejador d'aplicació **ALB** i el nating per a poder accedir des de la xarxa publica a la privada. Existeixen dos "security groups" un en cada xarxa per obrir nomes els ports que es necessitin.

A la xarxa privada tenim el **ECS (Elastic Container Service)** per a la gestió de contenidors junt al servei del **Fargate.** El **ECS** amb **Fargate** ens proporciona d'una forma molt senzilla:

- Gestió dels contenidors (parada, arrancada, etc)
- Auto escalat de contenidors en funció de la memòria consumida, cpu consumida, alarmes definides a mida , etc.
- Configuració senzilla
- Molt bona integració amb la resta de serveis de **AWS**

En aquest escenari, no tenim un volum gran de "micro" serveis de tipus diferents.

No es recomana la utilització de **EKS** ja que:

- Es molt mes car que l'**ECS** .
- Es molt mes complicat de mantenir i configurar que l'**ECS**
    - El stack de kubernetes es extens
- No necessitem cap característica que tingui el **EKS** que no tingui el **ECS**
- No partim d'un producte que ja funcioni en kubernetes i que vulguem desplegar
- Si algun dia volguessis emportar-te el projecte a una altra núvol o entorno s'hauria de tornar a configurar i adequar el kubernetes a la versió del moment.
    - Lo més normal es que la versió **EACAT 4.0** es migri a una nova tecnologia.

Si durant la vida de l'aplicació, es detectés alguna característica de **EKS** que es necessites i que no tingues l'**ECS,** es podria canviar el gestor de contenidors de forma transparent per l'aplicació.

Per assegurar que la aplicació està sempre disponible, el desplegament es fa amb una mateixa regió però amb les tres zones de disponibilitat. En aquest sentit la base de dades **Aurora Serverless** ja manega la alta disponibilitat.

Per mantenir la infraestructura com a codi es planteja fer servir **Terraform** per la seva versatilitat i facilitat de d'us. No es recomana fer servir **CloudFormation** degut a la seva complexitat.

6. Seguretat de la plataforma

Totes les comunicacions que surten a internet es porten a terme mitjançant **HTTPS** ja siguin des de el **Cloudfront** com des de el **API Gateway.**

Aquest dos serveis permeten fer servir el servei **WAF** com a Firewall web el qual permet bloquejar tràfic, etc. Fer us de AWS Shield per a protegir-se dels atacs DDoS (Distributed Denial of service)

Internament

Tot el que representa custodia de documents associats a la tramitació es delega en el servei **Desa'l** ja implementat a **l'AOC**.

Per el que respecta als diferents serveis de **AWS** que es fan servir a la solució suporten xifrat de dades:

Servei	Nivell de xifrat de dades
S3	AES-256
Redis	AES-256
SQS	AES-256
PostgreSQL	AES-256
Firehose	AES-256
OpenSearch	AES-256

En tots aquests serveis es poden xifrar les comunicacions internes.

7. Model de convivència EACAT 3.0 i EACAT 2.0

L'aplicació **EACAT 2.0** es molt extensa, la migració al núvol no es pot fer tota de cop, en "**big bang**". Per tant la versió **EACAT 2.0** i la **3.0** hauran de conviure mentre es desenvolupen totes les funcionalitats.

Per a que es puguin posar serveis a la plataforma **EACAT 3.0** ha de permetre poder executar tot el cicle d'un tràmit:

- Configurar el tràmit en el seu servei.
- Crear-lo / Editar-lo amb totes les funcionalitats necessàries.
- Tramitar-lo amb totes les seves integracions pertinents.

L'estratègia de migració es anar migrant poc a poc les serveis quan la base de la plataforma ja estigui desenvolupada. Anar treien serveis de la plataforma antiga i anar-los posant a la nova.

Regles de convivència

- L'**EACAT 3.0** mostrarà a les seves bústies tots els expedients i tràmits que es facin des de el **EACAT 2.0.**
    - L'accés a aquestes dades concretes dels expedients / tràmits es faran des de **EACAT 2.0**. L'**EACAT 3.0** redirigirà a la versió antiga (link).
- L'EACAT 2.0 deixarà de mostrar aquells expedients i tràmits dels serveis que ja s'hagin migrat al EACAT 3.0
    - D'Els tràmits que s'hagin migrat a **EACAT 3.0** només es podrà consultar els seus **PDFs** generats però no el formularis **HTML** perquè no tindran.
- A l'**EACAT 3.0** es podrà consultar la configuració dels ENS, SERVEIS, TRAMITS, etc, de totes les entitats del **EACAT 2.0** però no les podrà modificar fins que la funcionalitat hagi sigut migrada. A la sincronització es farà una configuració bàsica dels serveis, tràmits, etc.

Amb aquestes regles l'EACAT 2.0 cada cop podrà veure menys coses y l'EACAT 3.0 sempre ho veurà tot però cada vegada podrà fer mes coses sense tenir que anar a l'EACAT 2.0

7.1. Sincronització de dades EACAT 2.0 i 3.0

L'aplicació es divideix en dos grans blocs d'entitats:

- Configuració del negoci: ENS, SERVICES, PROCEDURES, INBOX, etc
- Execució del negoci: SERVICE INSTANCE, PROCEDURE INSTANCE, etc

Després es tenen altres entitats: USERS, EXTERNAL APPLICATIONS, etc, que son d'una naturalesa molt més estàtica i per tant a priori no es necessita una sincronització automàtica.

Necessitats de la sincronització:

- Ha de ser robusta per a no perdre dades ni crear inconsistències de dades. Per això es faran servir cues **SQS.** Les quals asseguren l'entrega de les dades entre els dos costats de la cua.
- La sincronització ha de ser **idempotent** per a poder executar la sincronització tantes vagades com faci falta. D'aquesta manera en cas d'inconsistència es pot executar una re-sincronització parcial o de totes les dades.
- La sincronització de les dades sempre serà des de **l'EACAT 2.0** al **EACAT 3.0.**

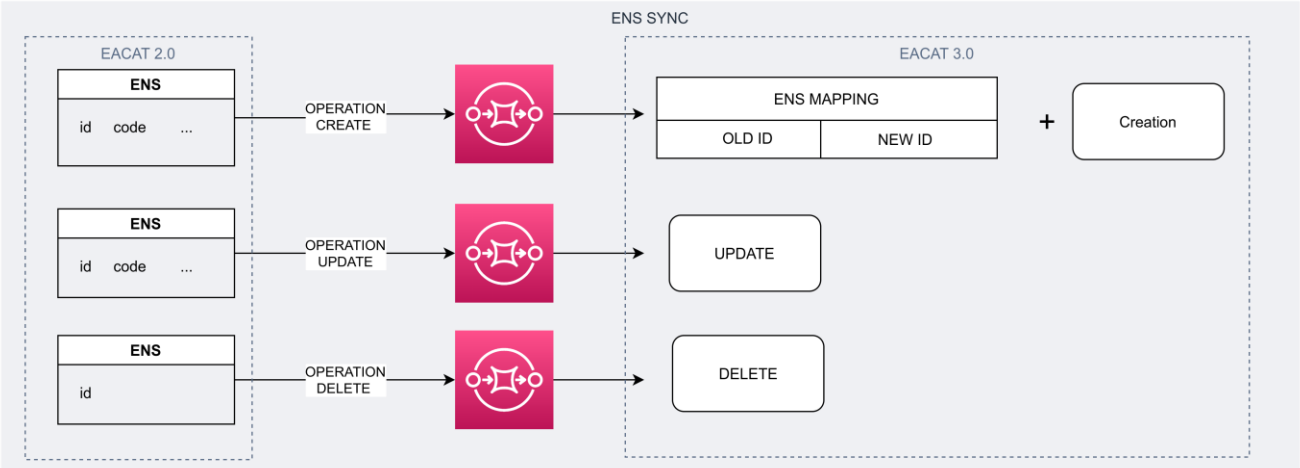
Per a que les sincronitzacions siguin **idempotents** es te una taula de mapeigs entre les entitats a la base de dades antiga i la nova. D'aquesta manera es senzill saber si s'ha de fer un "insert" o un "update" buscant si el mapeig existeix.

Per accelerar les migracions inicials, que son les més massives, es tindrà un “flag” on s’indiqui que es una carrega inicial i per tant tot el que s’han de fer es un “inserts”.

7.2. Sincronització de les entitats de configuració de negoci

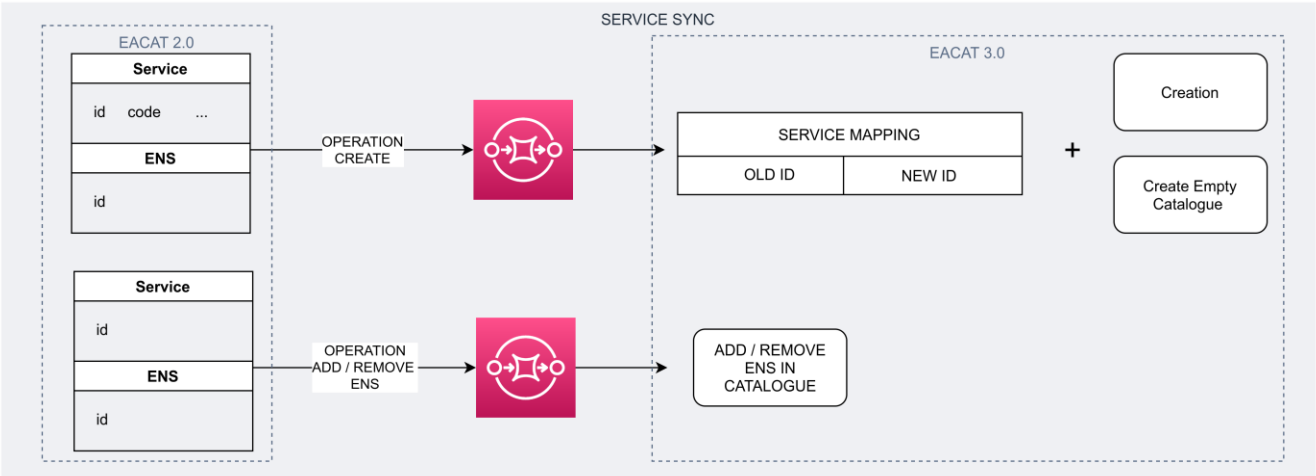
Aquestes sincronitzacions es llençaran de forma manual ja que el procés de canvi d’aquestes entitats es fa des de l’AOC mitjançant un usuari d’administració.

7.2.1. Sincronització dels ENS



Es el **CRUD** bàsic d’una sola entitat.

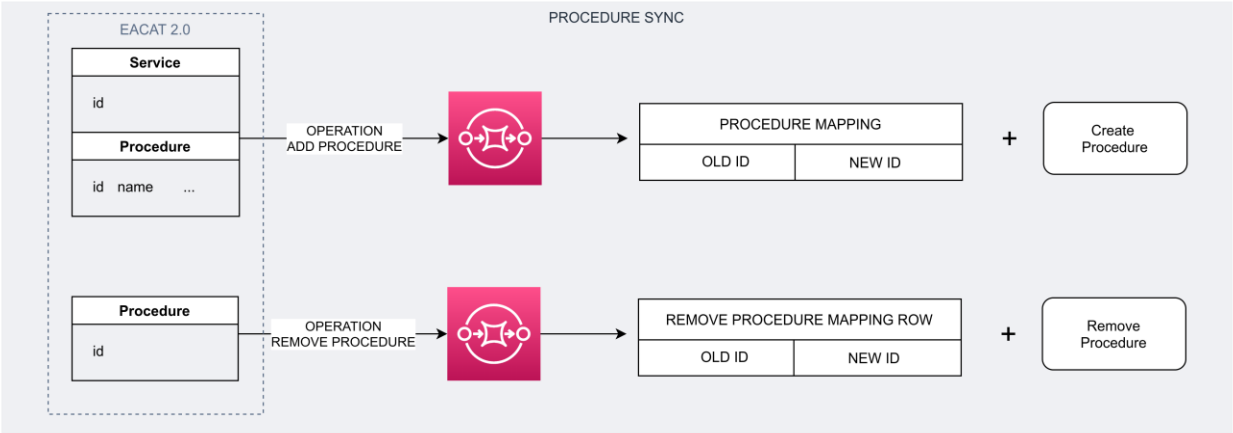
7.2.2. Sincronització dels SERVICES



Només es sincronitzen les dades del servei u les relacions que tenen amb el **ENS prestador** i els **ENS receptors**.

Les relacions que ni ha entre els tràmits d'un servei no es migren, ja que aquestes es configuraran quan el servei es migri definitivament.

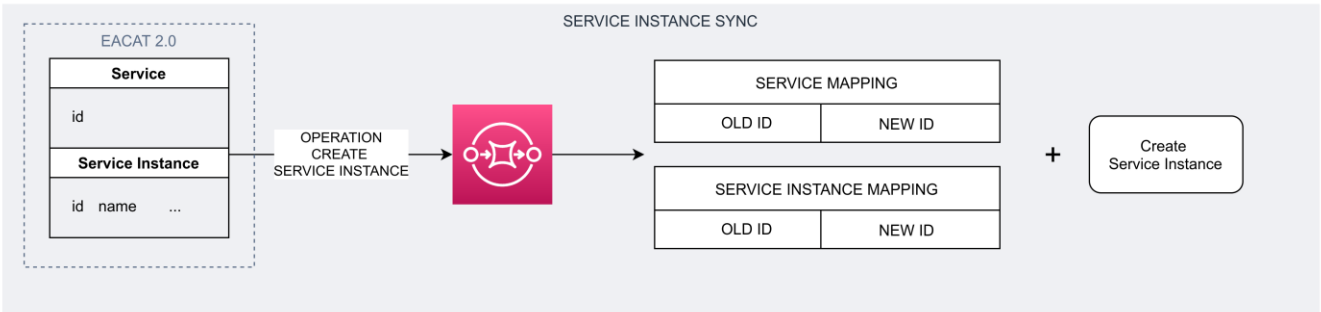
7.2.3. Sincronització dels PROCEDURES

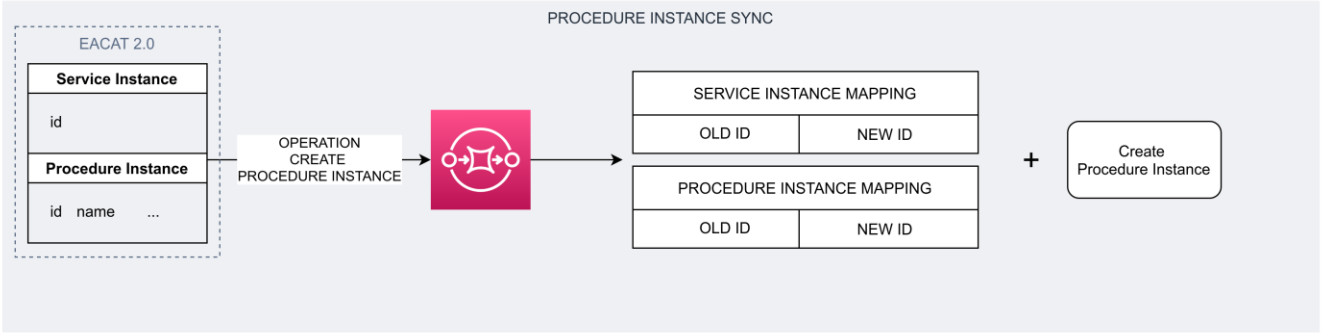


Es una sincronització d'una entitat, un **CRUD**.

7.3. Sincronització de les entitats d'execució

Aquesta sincronització es automàtica en el moment que es crea/modifica un expedient o es crea/modifica un tràmit.





Només es sincronitzaran els expedients e instancies de tràmits que prèviament s'hagin sincronitzat el seu servei amb els seus tràmits.

8. Serveis oferts per el **EACAT 3.0**

Existeixen dos tipus serveis oferts per **EACAT 3.0** , els serveis:

- **De consulta d'informació**. Ex: Recuperar tràmits d'un expedient, cercar tràmits entre dues dates, etc.
- **Creació / Modificació de les entitats**. Ex: Executar una tramitació, afegir un document adjunt, signar, etc.

Per els dos tipus de serveis es farà servir crides **HTTP REST JSON** i depenent de si el client es legacy o no s'hauran de fer adaptadors a **Web Services**, etc. En el cas dels serveis que creïn o modifiquin entitats de negoci s'estudiarà si han de ser síncrones o asíncrones com el cas de **PICA.**

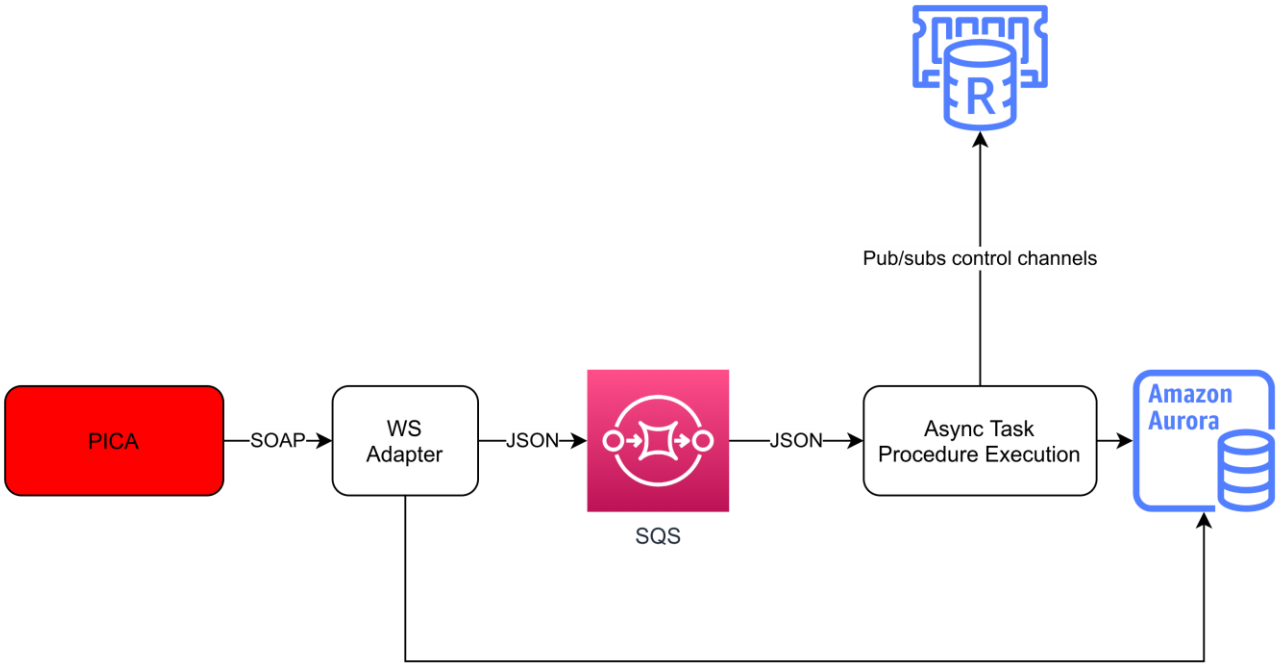
8.1. Autenticació i autorització dels serveis oferts

Els serveis només son oberts pers els BackOffice dels **ENS** i per a sistemes de dintre de l'**AOC.** L'autenticació dels clients del serveis es fa mitjançant **JWT** com una capçalera de les peticions **HTTP REST**.

La gestió de la clau privada per a firmar el **JWT** es gestiona des de la mateixa plataforma de **EACAT3.0 .** Cada clau privada te associada uns **ENS,** de tal forma, que tots el consums dels serveis estan restringits al **ENS** en concret que ve donat per el **JWT**.

8.2. PICA

Les peticions entrants per **PICA** segueixen aquest esquema:



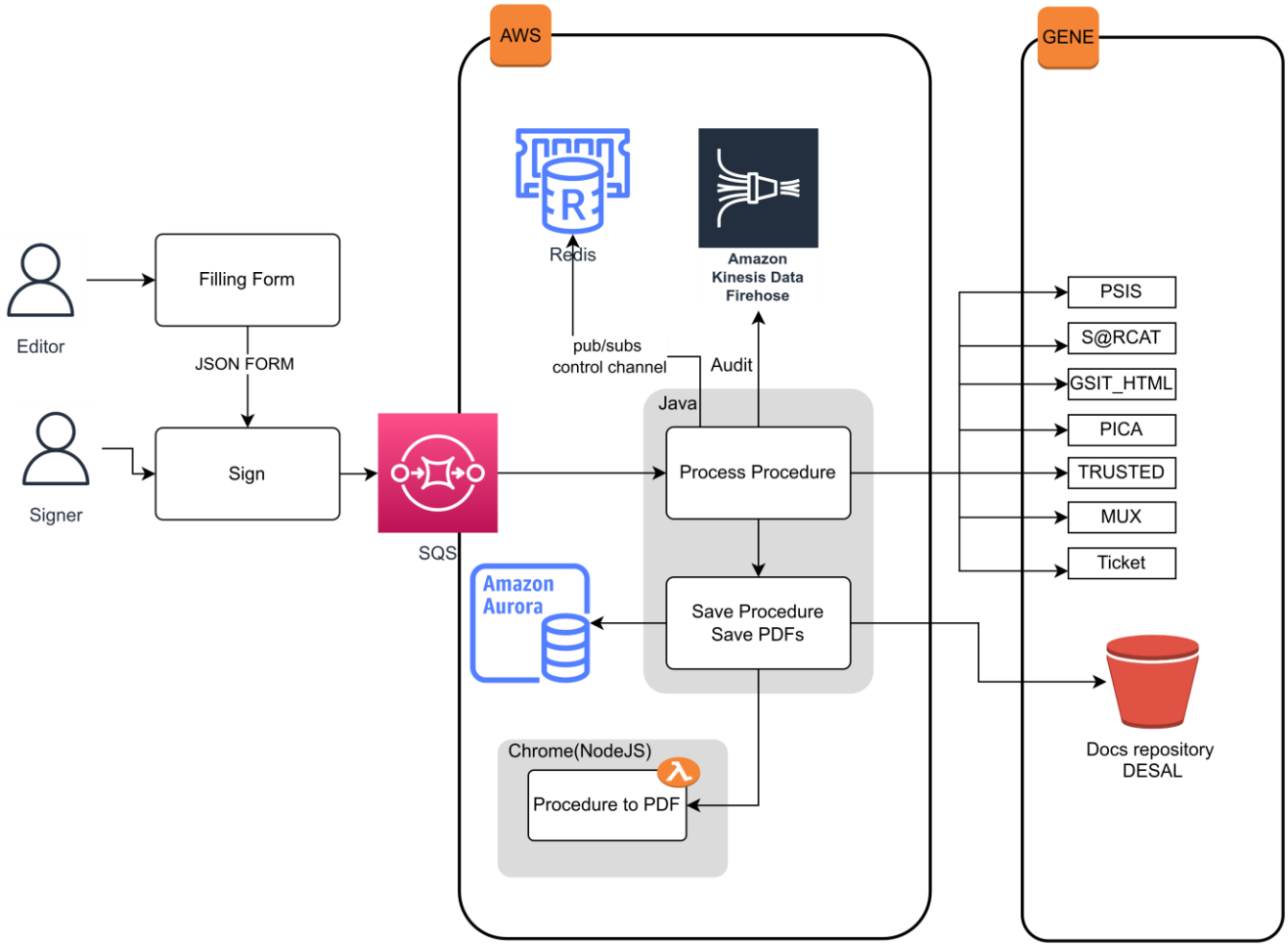
El connector amb **EACAT 3.0** es un servei **SOAP**. El **WS Adapter** el transforma en un **JSON** i l'introdueix en una cua per a ser tractat de formar asíncrona si es tracta d'una tramitació. Si es tracta d'una consulta de dades la fa directament contra la base de dades i retorna el resultat.

El **Async Task Executor** es la tramitació normal de la resta de tràmits.

El servei de **PICA** es tranversal a tots els **ENS** i només pot ser executat per **PICA**, per tant autenticació del client de pica es podrà fer per rang de **IPs** per a no modificar el client ja fet.

9. Async Task, execució d'un tràmit

L'execució d'una tramitació por ser costa en temps, més de 20 segons en alguns dels casos. També a vegades hi ha problemes en les aplicacions externes com Signar, MUX, etc i aquestes estan caigudes. En aquest casos es necessita una política de reintents fins que l'aplicació torna a la normalitat.
Per això es proposa el següent esquema d'execució de tramitació per fer aquesta tasca en diferit:



Es proposa una execució de la tramitació de forma asíncrona, de tal forma que l'usuari no estigui esperant fins al final de la tramitació per a veure si anat be o no. L'usuari pot consultar l'estat del tràmit des de la bústia que el va crear a la safata de en Procés.

Des de la safata de en Procés es mostraran aquells tràmits, que després de fer tots els reintents, no han aconseguit fer la tramitació junt amb una explicació del error que s'ha produït. Els usuaris podran re-intentar la tramitació una altra vegada quan el error s'hagi arreglat.

Hi ha circumstancies on es sap que els serveis externs estaran parats o trigaran a estar operatius. Per això es permet des de la consola d'administració parar el consum de la cua de tramitació de forma arbitraria. Per aquest tipus de control es fa servir la capacitat que te el **Redis** de publicar i subscriure a events. Amb dos events es tindria suficient per a para la el consum de la cua i posar-la en marxa.

El flux de la tramitació seria el següent:

- L'usuari omple el formulari i el deixa llest per signar.

- L'usuari signador el signa o deixa que la signatura es faci al servidor. La tasca de tramitar es posa en una cua de tramitacions.
- De forma asíncrona es van agafant les tasques de tramitació i es van executant.
    - Es contacta amb els diferents sistemes per signar, registrar etc
    - S'imprimeix amb PDF.
    - El registres
    - Etc

10. Sistema de bústies

10.1. Context

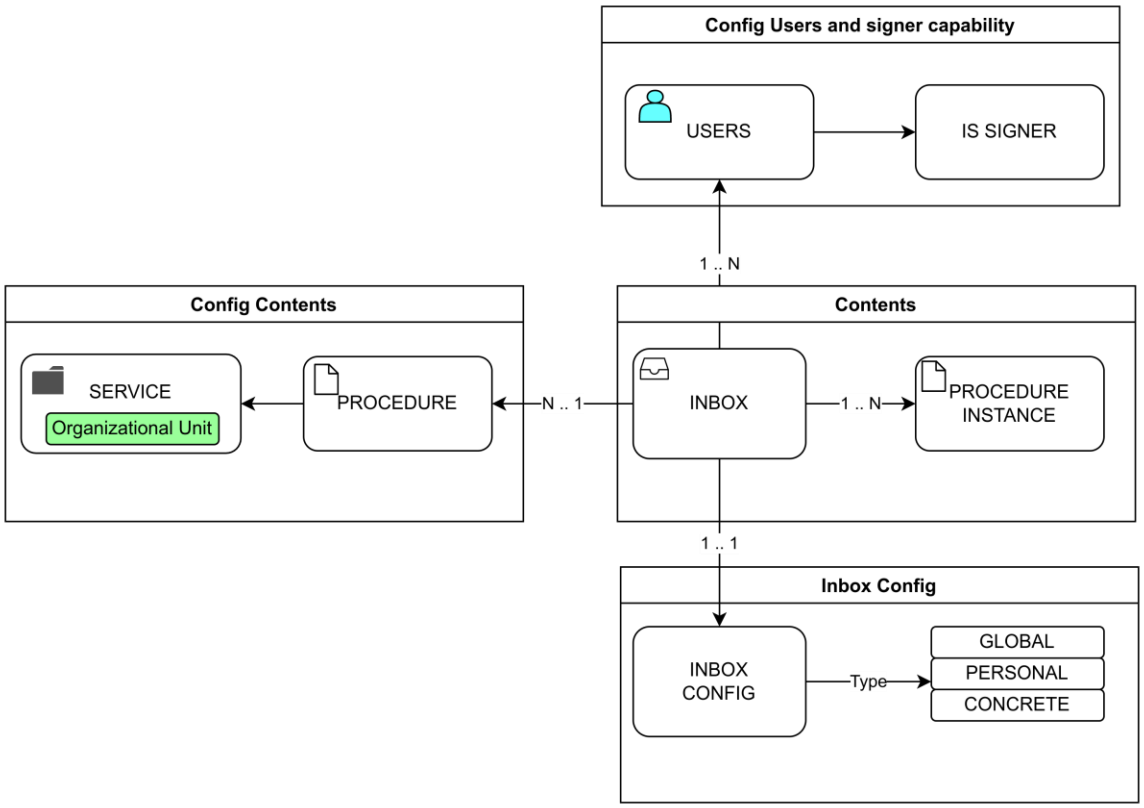
En el **EACAT 2.0** es hi ha producte cartesià entre usuaris, serveis, tràmits i rols per a calcular la visualització dels tràmits. Es un càlcul costós que comporta moltes taules i que si li sumes molts usuaris concurrents, dona com a resultat, problemes de rendiment en la visualització dels tràmits.

10.2. Funcionament de les bústies

Aquesta funcionalitat té dos finalitats:

- Simplificar la gestió dels permisos dels usuaris enfront els serveis que pot veure, etc.
- Simplificar l'accés als tràmits que pot veure l'usuari, traslladant el cost de decidir quin tràmit pot o no pot veure un usuari de la consulta a la creació del tràmit.

Una bústia conté les instàncies tràmits d'un conjunt de serveis configurats:



A l'usuari se li assigna una bústia, idealment 1 o 2, i a la bústia se li assigna els tràmits que contindrà.

Cada cop que es tramita, es decideix en quines bústies del **Prestador** i **Receptor** s'ha de ficar el tràmit en qüestió. La visualització es torna molt senzilla ja que només has de veure els tràmits que estan a la bústia seleccionada.

Els permisos i l'accés queden simplificats, ja que només es pot veure els tràmits de les bústies i la "select" només ha de tenir en compte la bústia que estàs visualitzant.

El preu que es paga es que s'han de mantenir les referencies de les bústies a nivell de base de dades. Que son petites una, "tupla" de identificadors, el de la bústia i el del tràmit. Tenint en compte que les bústies es podran historificar, es a dir, esborrar registres cada X temps, no hauries de créixer en excés.

11. Cercador de Catàleg

El cercador de catàleg cerca per contingut en el títol del tràmit i del servei. A la vista del meu **ENS** ha de tenir en compte que ha de filtrar aquells serveis que estiguin oferts únicament al meu **ENS.**

11.1. Sincronització

Amb la funcionalitat del Catàleg el principal problema es la sincronització entre el la base de dades **Aurora Serverless** i el **openserach**. Aquesta sincronització es pot portar a terme en el moment d'aprovar els serveis i en el moment de guardar les modificacions fetes a un servei o un tràmit. Si surt un error en la sincronització l'usuari seria informat per pantalla i la operació quedaria cancel·lada. Aquesta sincronització es síncrona ja que modificar / crear una entitat al **opensearch** no es una operació costosa en temps.

El administrador de l'aplicació per part de l'**AOC** en qualsevol moment podrà llençar la sincronització d'un servei o de tots els serveis (aquesta acció es de forma asíncrona).

12. Motor de formularis

El motor de formularis te les següents característiques:

- Els formularis es descriuen amb un **JSON**
- Els valors dels camps es guarden amb un altre **JSON**
- Es guarda la versió del formulari.
- Es guarda el tipus de formulari per identificar l'stack tecnològic.

Per a poder suportar qualsevol stack tecnològic es dona la possibilitat de fer servir **iframes** i mitjançant el protocol de **HTML5 Post Message** permetre la comunicació entre els diferents iframes. Al ser al mateix domini es solucionen molts problemes de "**cross-domain**".

Limitacions dels formularis

- S'han de poder mostrar en una sola pàgina per a que el navegador chrome pugui imprimir-los completament.
- Tots els camps s'han de veure completament amb d'impressió, es a dir, no pot haver scroll als texts àrees, etc.

Comportaments

- Responsive
- Col·lapsar regions del formulari
- Poder mostrar uns camps o uns altres en funció de camps anteriors
- Poder html arbitrari (sempre que no sigui un camp del formulari)
- Repetir seccions del formulari (també mostrar array de components)
- Netejar els camps

- Poder configurar el tooltip del camp que surt (caption)
- Camps emplenats de només lectura (Ex: Nom del ENS, el seu CIF)
- Amagar seccions o components quan s'imprimeixi.
- Han de complir uns **CSS** donats per **EACAT**
- Poder mostrar els camps en 1 , 2 o 3 columnes
- S'ha de poder mostrar el formulari en "read only"
- S'han de poder fer "disable" camps

Components dels formularis

- Cercadors predictius.
- Botons de validació personalitzats per a poder validar des de servidor.
- Crides a BackOffice externs per auto completar, validar, etc.
- Components bàsics de :
    - Input text.
    - Text Area.
    - Date picker.
    - Slider.
    - Radio button.
    - Check box.
    - Labels.
    - Select.
    - Input file
- Selects dependents.
- Ajuda del formulari.

Motor de referencia

La primera implementació del motor de referencia es fa servir: https://formly.dev/
El qual es un framework que permet:

- Descriure els formularis en JSON
- S'ajusta al Stack tecnològic inicial del projecte: Angular
- S'ajusta el tema CSS (Material) fet servir a la plataforma
- Conté la majoria de components i comportaments que es necessiten d'entrada.
- Es poden crear components a mida reutilitzables

Aquesta implementació de referencia no fa servir iframes

Implementació dels formularis

Per tal de deslligar la selecció d'una llibreria amb la funcionalitat que esperem d'ella, es defineix un llenguatge de "formularis" expressats amb JSON el qual representa totes les necessitats dels formularis, Ex:

```
{
  "element": [
    {
      "name": "Section",
      "type": "section",
      "icon": "ballot",
      "properties": {
        "commonProperties": {
          "id": "id512295"
        }
      },
      "items": [
        {
          "name": "layout",
          "type": "layout",
          "icon": "view_quilt",
          "properties": {
            "commonProperties": {
              "id": "id867749"
            }
          },
          "columns": [
            {
              "name": "TextArea",
              "type": "textarea",
              "icon": "text_fields",
              "properties": {
                "commonProperties": {
                  "id": "id582119"
                }
              }
            }
          ]
        },
        {
          "name": "layout",
          "type": "layout",
          "icon": "view_quilt",
          "properties": {
            "commonProperties": {
              "id": "id178005"
            }
          },
          "columns": [
            {
```

```
          "name": "Text",
          "type": "input",
          "icon": "title",
          "properties": {
            "commonProperties": {
              "id": "id483465"
            }
          }
        }
      ]
    },
    {
      "name": "Afegir",
      "type": "adder",
      "icon": "",
      "properties": {
        "commonProperties": {
          "id": "id835791"
        }
      }
    }
  ]
 }
]
}
```


Aquest JSON no està pensat per a que sigui creat/modificat a mà. Es proveirà un editor de formularis que doni assistència a aquesta tasca.

13. Auditoria d'accions sobre els tràmits

La auditoria d'accions està pensada per tenir un log d'accions que s'han fet sobre l'aplicació. Ex:

- Tramitació d'un formulari.
- Signatura d'un formulari.
- Error en la tramitació.
- Detecció de virus en un fitxer adjunt.

Es pot auditar tot el que es necessiti.
Per l'auditoria es fa servir el servei **kinesis-Firehose** per a poder emmagatzemar aquests resultats a **S3.**

Tots aquestes dades es poden explotar amb **Athena,** un altre servei.

Per que la comunicació no penalitzi la execució de la aplicació es fa servir el client asíncron de java el qual permet llençar la execució en un altra thread.

14. Particionament de les entitats

El volum de registres en algunes taules de l'aplicació actual creixen a un ritme de 2M de registres anuals i augmentant. Per el tipus de negoci que es manega, els registres creats fa dos o mes anys no son utilitzats o la seva utilització es anecdòtica. Particionar aquestes taules per la data de creació del registre, mantenint en particions petites els registres més utilitzats i en grans els més antics, pot augmentar considerablement la velocitat d'accés i creació dels registres. Amb aquesta tècnica la base de dades pot mantenir un històric molt elevat de dades.

Els candidats per fer servir el particionament de taules son les entitats de instancia de tramis e instancia de serveis.

Existeixen altres entitats que també creixen de forma molt ràpida, com serien:

- Valors dels formularis en format json.
- Missatges d'auditoria.
- Fitxers adjunts.

Aquestes entitats es guarden en:

- Valors dels formularis en format json: **S3**
- Missatges d'auditoria: En format **JSON** en un **S3**
- Fitxers adjunts als tràmits: Aplicació Desa'l. Les referencies retornades per el desa'l també es guardaran en un **S3.**

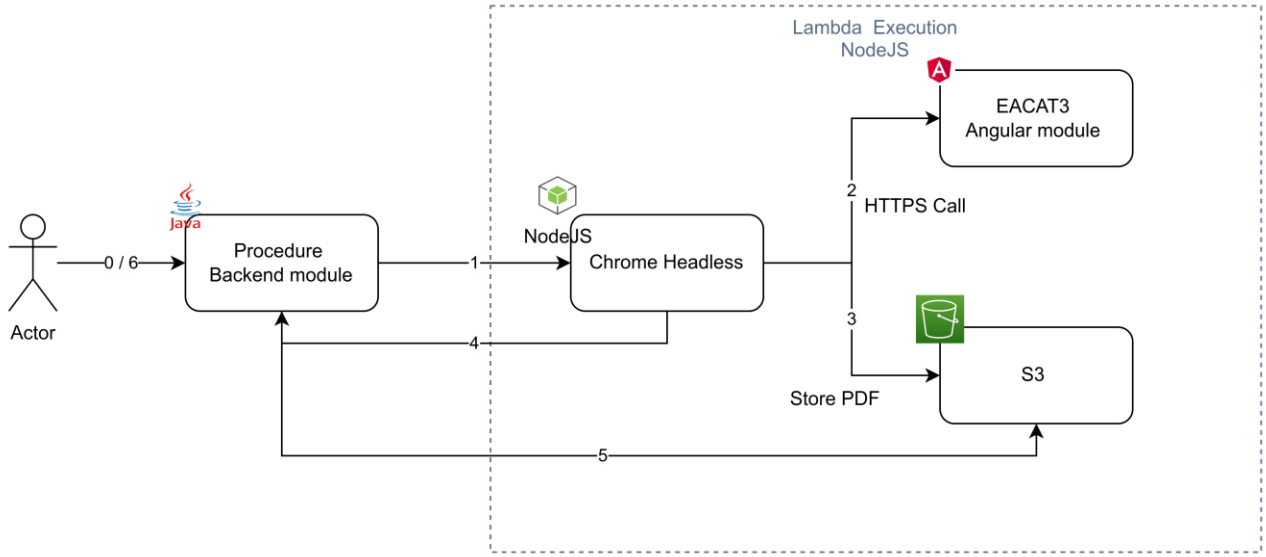
Veure: https://dev.mysql.com/doc/refman/5.7/en/partitioning.html per a més informació sobre el particionament de taules. Aquest concepte es fet servir en altres motors de base de dades com Oracle:
https://docs.oracle.com/cd/E18283_01/server.112/e16541/part_admin001.htm

El particionar taules fa que puguin créixer sense afectar al rendiment. Totes les consultes a la base de dades, a les entitats abans mencionades, aniran amb un filtratge per data per a que sempre es facin servir les particions.

15. Impressió en PDF

La impressió de PDF es fa mitjançant la combinació de Lambdas + Chrome en mode headless.
Google proporciona les llibreries (https://developers.google.com/web/updates/2017/04/headless-chrome#node) en NodeJS.

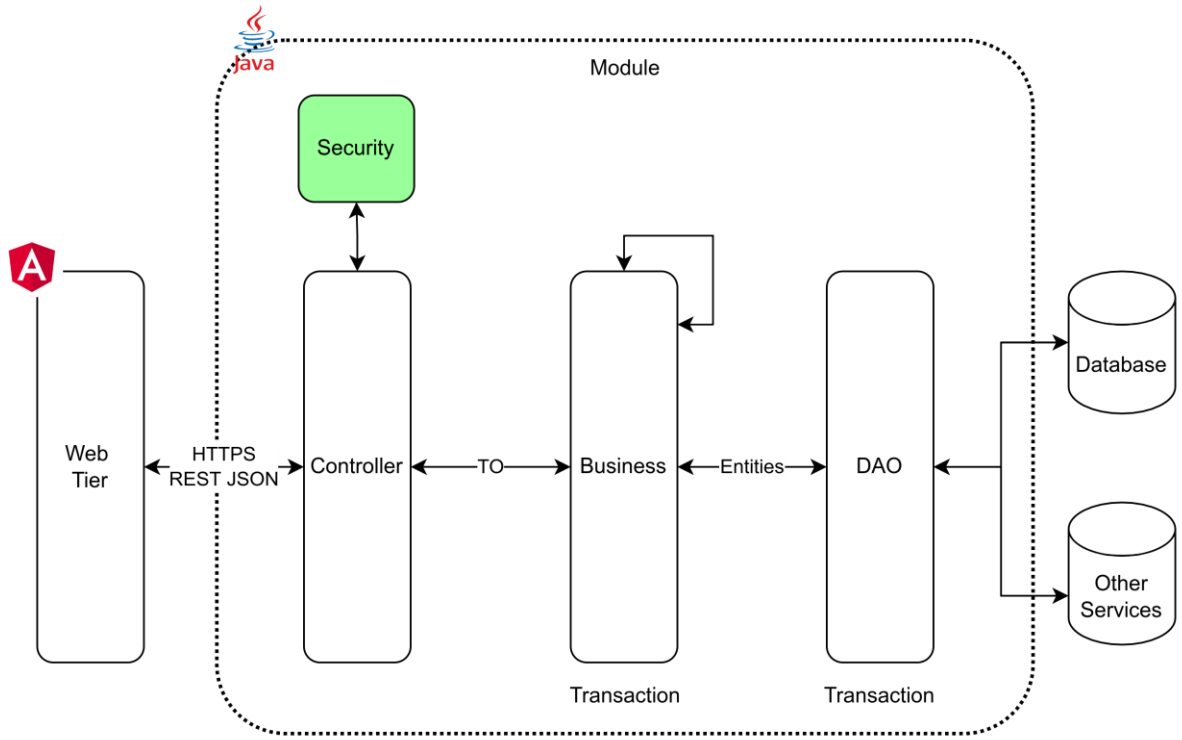
El flux concret per a imprimir un PDF seria:



0. L'usuari fa una petició d'impressió d'un formulari al servidor d'**EACAT**
1. El servidor de **EACAT** fa una petició a un servei lambda encarregat de fer aquesta impressió.
2. La lambda executa codi en NodeJS (https://developers.google.com/web/updates/2017/04/headless-chrome#node) per a la impressió del formulari. Això no es mes que aixecar un navegador chrome sense interfície gràfica i fer una crida a la aplicació web per a mostrar el formulari en format de impressió.
    a. En aquesta comunicació es fa servir **JWT** per validar que el qui demana la impressió es el servidor d'EACAT i no qualsevol usuari. D'aquesta manera evitem que es puguin imprimir formularis als quals l'usuari no te accés.
3. La lambda deixa el **PDF** en un bucket de **S3.**
4. La lambda acaba informant del resultat de la impressió.
5. El servidor d'**EACAT** descarrega el **PDF** del **S3**
6. El **PDF** es retornat al usuari que ha fet la petició

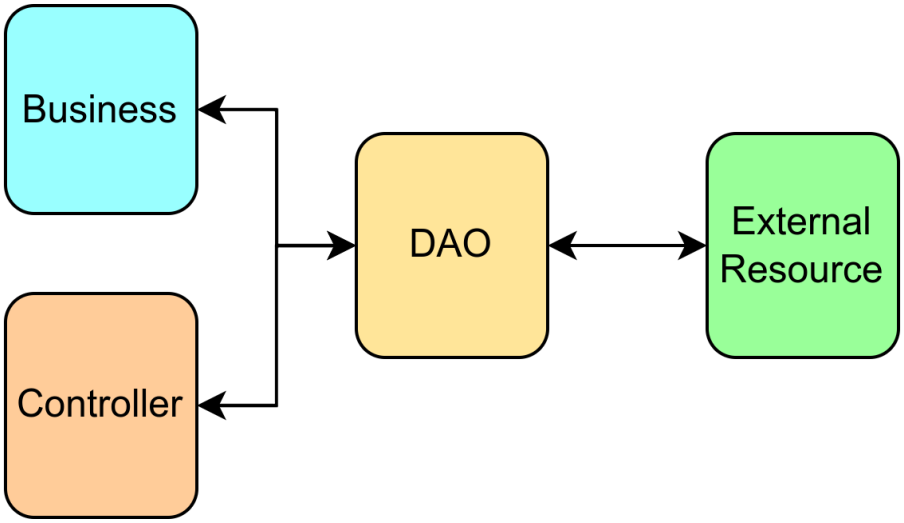
15. Estructura general d'un mòdul (Layers)

La estructura general dels mòduls consisteix en tres capes ben diferenciades:



On:

- **Controller:** Aquesta capa te com a principal tasca redirigir les peticions als diferents processos de negoci definides al business tier. També es comprova que l'usuari que fa la petició te accés els diferents recursos que està sol·licitant. La comunicació amb la capa de negoci es fa mitjançant "**Transfer Objects**".
- **Business:** Aquesta capa te la responsabilitat de resoldre les peticions amb el negoci. Hi ha dos tipus de objectes en aquesta capa, els que parlen amb la capa de control (facades) i els que no es comuniquen entre capes. Els primers retornen **TO** i els segons treballen amb entitats (Procedure, Service, etc). En aquesta capa es comencen les transaccions que calguin. Per defecte les transaccions seran "Read Only" i només es començaran transaccions d'escriptura quan el procés de negoci ho requereixi.
- **DAO:** Aquesta capa serveix per a independitzar l'accés a les dades per part de la capa de negoci. Aquesta capa independitza de la tecnologia feta servir, tal com, queues, opensearch, base de dades, athena, s3, etc. Per accedir a cada un d'aquests recursos sempre es farà amb una interfície per d'acant que et permeti fer una separació clara.



16. Autenticació i Autorització

L'autenticació dels mòduls es farà mitjançant **Oauth2** contra un **IDP** proporcionat per l'**AOC.**
L'autorització es portarà a terme en el mòdul de tramitacions.
Els usuaris es podran crear de dues maneres,

- Des de la administració del **ENS**
- La primera vegada que es connecta a la aplicació.

Les dades mínimes que han de venir des de el **IDP** son:

- Document de identificació de l'usuari. (DNI,)
- Email
- ENS al que pertany

Si l'usuari es crea automàticament per l'aplicació, aquest usuari no tindrà cap bústia assignada ni cap tipus de rol.

Subapartats 1..n

Casos d'us principals

Integracions amb sistemes externs

//TODO
Integracion con desal, MUX3

Vista de desplegament

Vista de dades

Vista de Manteniment

Vista de implementació

Matriu de desenvolupament

La matriu de desenvolupament inicial consta dels següents elements que s'aniran actualitzant a mesura que es necessiti.

Backend

- OpenJDK 13
- Spring Boot 2.6.2
- Hibernate 5.6.3 & JPA 2
- Spring AWS
- AWS for Java SDK 2

Frontend

- Angular ~12.2.0 (https://github.com/angular/angular)
- Material ^12.2.13 (https://github.com/angular/components/tree/12.2.13)
- Nodejs v16.14.0 LTS
- Npm 8.3.1

Vista de seguretat

Riscs principals

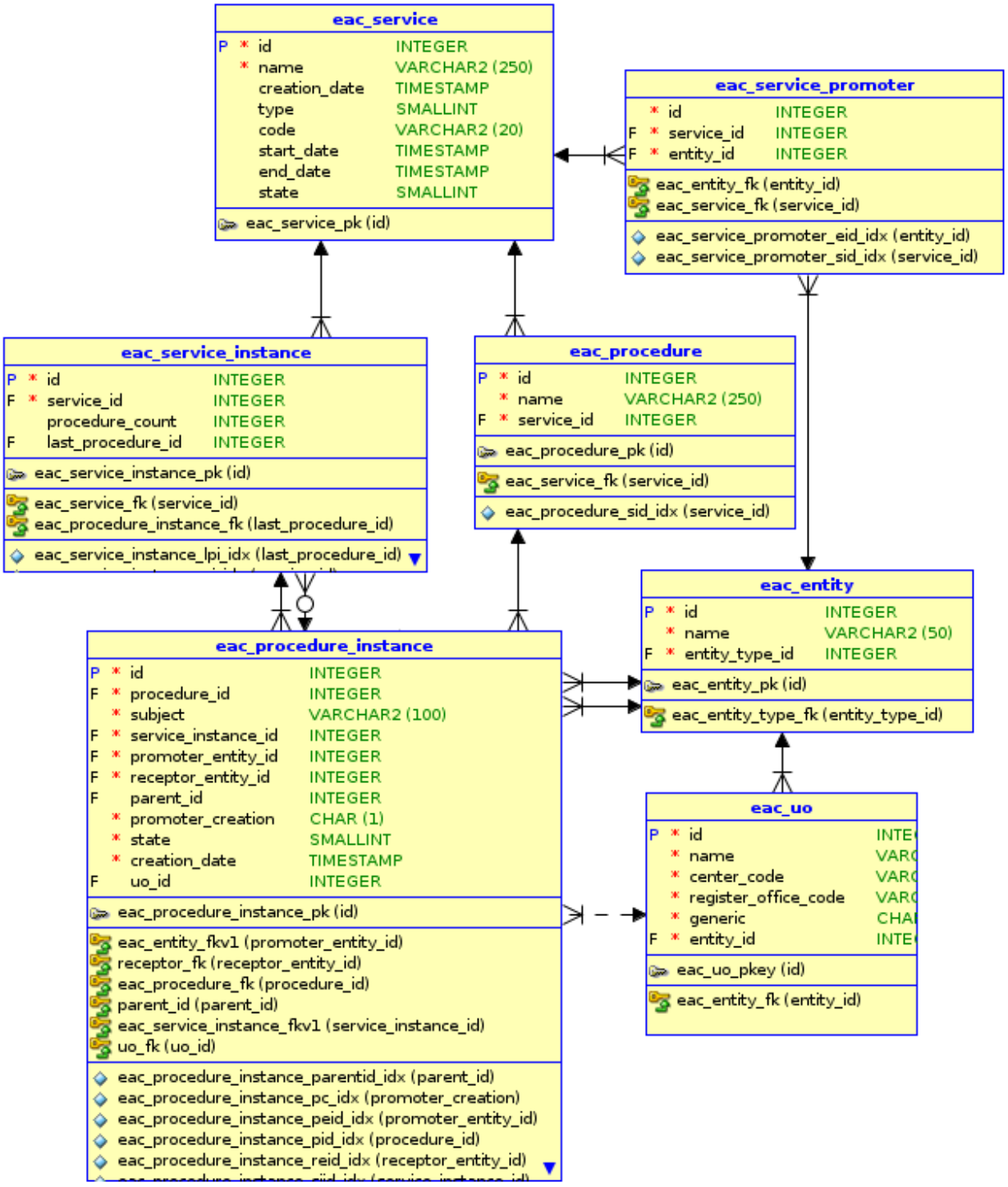
Referencies

1. APENDIX

2. POC viabilitat bústies

No s'han posat tots els camps necessaris, a les taules, només els imprescindibles per entendre la proposta

En aquesta POC es fa una proposta de taules principals



On:

- **eac_service_instance**: Conté les instàncies de serveis.
- **eac_procedure_instance**: Conté les instàncies de tràmits.
  - **uo_id** : Es la referencia a la unitat organitzativa a la qual es va fer al tràmit. Aquesta referencia implica que les unitats organitzatives no es poden

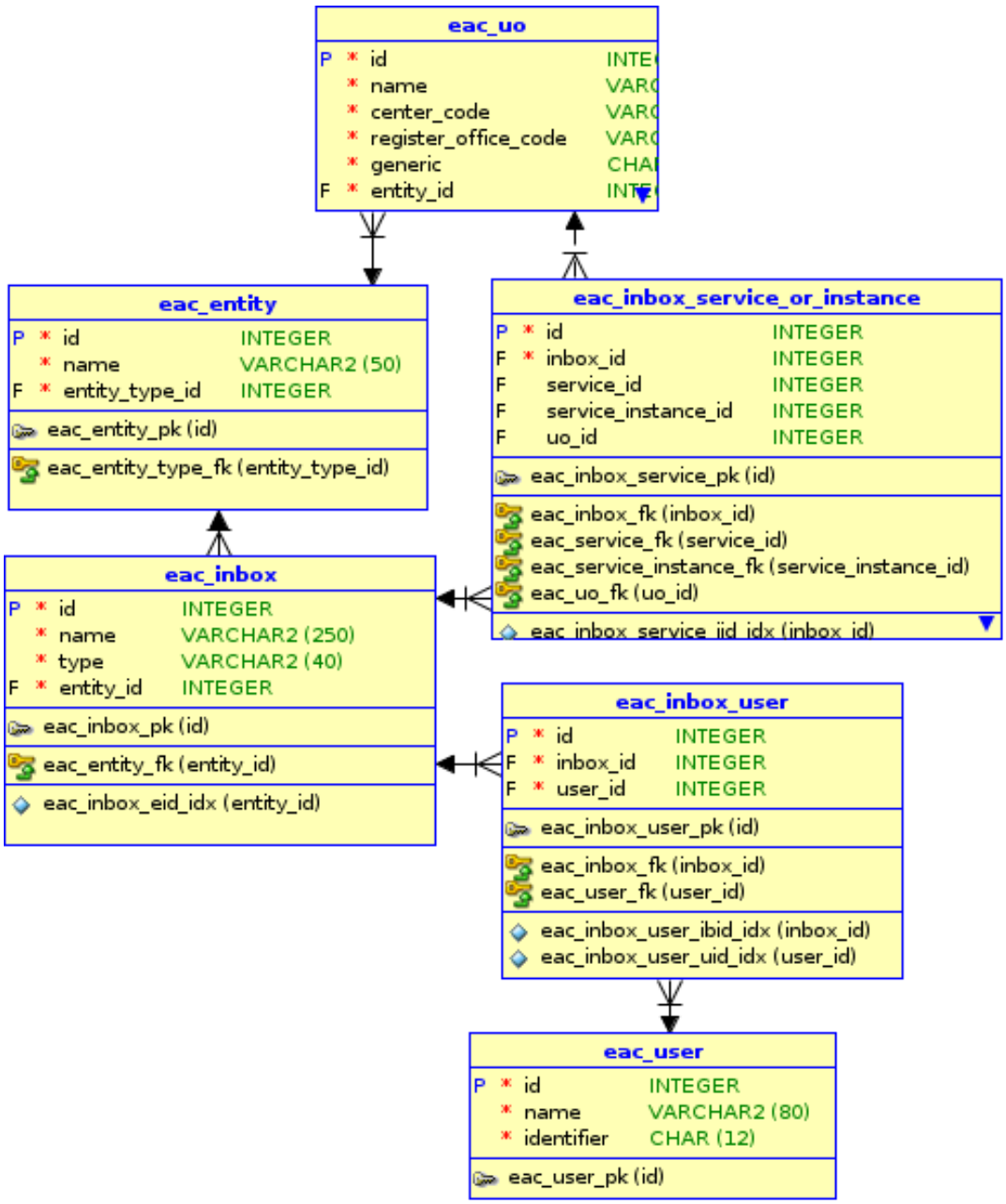
esborrar, només donar de baixa. Aquest camp ha de ser-hi aquí per a poder fer les bústies en funció de la unitat organitzativa i facilitar les cerques.

- **eac_service_promoter**: Conté el promotor del servei. En el cas de que sigui un servei genèric el servei pot tenir més d'un promotor.
- **eac_procedure:** Conté la definició dels tràmits.
- **eac_service:** Conté la definició dels serveis.
- **eac_entity:** Conté la definició dels ens.
- **eac_entity_type:** Conté els tipus de entitats
- **eac_uoc:** Conté les unitats organitzatives de aquelles entitats que pertanyen al SARCAT

Processos associats:

- **Fer canvi de promotor d'un servei**: Només s'ha de canviar el promotor a la taula **eac_service_promoter**. Els tràmits ja executats tenen referencia als promotors on es van fer aquests tràmits.
- **Fer la select dels expedients amb l'últim tràmit nomes:** La taula **eac_service_instance** te els camps **procedure_count** i **last_procedure_id** que fan referencia a la quantitat de tràmits que te el expedient i quin es últim tràmit
- **Conèixer el pare d'un tràmit:** A la taula **eac_procedure_instance** el camp **parent_id** fa referencia a quin es el seu tràmit pare.

Proposta de taules per les bústies:



On:

- **eac_inbox:** Conté la definició de la bústia.
- **eac_inbox_service_or_instance:** Serveis que té la bústia o instancies de expedients que te la bústia assignada. El camp **uo_id** denota que es pot tenir en una bústia una sola o més uo d'un servei i no completament. Però en el moment que s'inclou completament un servei, s'han d'esborrar les referències a les UO.
- **eac_inbox_user:** Usuaris que conte la bústia.
- **eac_entity:** Entitats a les quals estan associades les bústies
- **eac_user:** Son els usuaris de la plataforma
- **eac_uo:** Representen les unitats organitzatives

Les bústies hi de tres tipus:

- **Bústia global**: La qual no te associada cap servei o expedient a **eac_inbox_service_or_instance** perquè en realitat els te **tots** on el ens de la bústia es promotor o receptor del servei.
- **Bústia personal**: Es la bústia d'una persona concreta. Només se li poden assignar expedients, per tant de la taula **eac_inbox_service_or_instance** només la columna **service_instance_id** estarà informada**.**
- **Bústia concreta**: Aquesta pot tenir usuaris, serveis i expedients assignats

Amb aquestes taules s'han fet les següents probes:

Context de la proba:

- La màquina de la base de dades es **db.r5.large**
- S'han carregat a la BD tots els tràmits, serveis, instancies de servei e instancies de tràmits per a totes les entitats del any 2021 (1.5M registres)
- S'ha escollit la entitat idenscode = 9612050006 que es la que mes tràmits a realitzat. (Departament de Drets Socials)
- Amb una usuària anònima
- Per a cada ENS es creen les següents bústies:
    - Una primera amb els primers 235 serveis que hi ha a la aplicació
    - Una segona amb els serveis que van des de el 235 al 470
    - Una tercera amb els serveis a partir del 470 en amunt
    - Una quarta amb dades que son una intersecció de les tres bústies anteriors
    - Una cinquena amb els primers 235 serveis i a mes tots els expedients de la segona bústia.

Consultes a testejar:

- Tots els tràmits d'una persona en totes les seves bústies en la safata d'expedients, on es mostra només el últim tràmit del expedient.

```
select epi.id as epi_id, es.name as service_name,esi.procedure_count, ep.name as
procedure_name,epi.subject,epi.service_instance_id
from eac_procedure_instance epi,
eac_service_instance esi,
eac_procedure ep,
eac_service es
where esi.last_procedure_id = epi.id
and (epi.promoter_entity_id = 4295 or epi.receptor_entity_id = 4295)
and ep.id = epi.procedure_id
and es.id = esi.service_id
and exists (
```

```
select 1 from eac_inbox_service_or_instance eis,eac_inbox_user eiu where
eiu.user_id = 29456
and eis.inbox_id = eiu.inbox_id
and (
(eis.service_instance_id is null and eis.service_id = esi.service_id and eis.uo_id is not
null and eis.uo_id = epi.uo_id)
 or
   (eis.service_instance_id is null and eis.service_id = esi.service_id and eis.uo_id is
null)
 or
   (eis.service_id is null and eis.service_instance_id = epi.service_instance_id)
)
limit 1
)
order by epi.id asc
limit 50
```

El temps en executar la consulta es de mitja: **150 ~ 180 ms**

- **Es limita a 50 resultats ja que seran el resultats per pàgina.**
- **Si es fa servir la sintaxis de paginació que et proporciona el Postgres (offset límit) quan més profunda sigui la pàgina a consular mes es triga. Es recomana afegir a la consulta una condició que "filtri" els registres de les pàgines anteriors. Ex: eac_procedure_instance.id < XXXX . Això fa que per avançar de pàgines s'ha de fer una a una. Te sentit fer-ho d'una altra forma?**

Conclusió:

- No s'ha provat de filtrar per tipus de tràmit (sortida / entrada, actiu, esborrany), etc, però aquests paràmetres ben indexats acotaran més els resultats i no haurien de canviar significativament el temps de resposta.
- Si la bústia que es consulta es la global llavors el tema dels repetits desapareix ja que si l'usuari te aquest tipus de bústia llavors com te accés a tot no es faran servir els filtres de bústia.

Aquesta es la consulta més costosa del llistat del meus tràmits.

3. POC viabilitat assignacions per tipus d'ens i llista blanca

Com implementen? Sempre llista blanca o un mix?
Nomes llista blanca implica manteniment si entra un ens