Universidad del País Vasco
Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

MÁSTER UNIVERSITARIO EN INGENIERÍA DE TELECOMUNICACIÓN

TRABAJO FIN DE MASTER

ANEXO I: DESCRIPCIÓN DE LA SOLUCIÓN REALIZADA: DISEÑO DE ALTO NIVEL

profesiolan

Empresa apoyada por:

Estudiante: Hospital Gómez, Ander

Director: Prieto Agujeta, Gorka

Curso: 2023-2024

Fecha: Bilbao, a 23 de enero del 2024

Intencionadamente en blanco.

Introducción

El presente anexo está englobado en el Trabajo Fin de Máster Dimensionamiento, desarrollo, despliegue y análisis de rendimiento de una api graphql con una arquitectura de microservicios y cluster de base de datos altamente escalable en aws para producción.

Además, el presente Anexo es el primero de una serie de dos, a través de los cuales se presentan los aspectos técnicos de todos y cada uno de los componentes que componen el proyecto realizado.

En concreto, este *Anexo I. Descripción de la solución realizada: Diseño de alto nivel* se centra en la exposición de alto nivel de abstracción.

Contenido

Lista de ilustraciones

1. Diseño de arquitectura propuesto

En esencia, la solución Profesiolan se trata de un Marketplace basado en un cliente web que consume un backend servido mediante una API. El presente trabajo recoge el proyecto de ingeniería relacionado con el desarrollo de la parte back-end.

Por supuesto, la arquitectura de alto nivel se ha desarrollado en el marco de cumplimiento de los objetivos indicados y desarrollados en el *Apartado 1.3: Objetivos y alcance del trabajo* del *Documento Principal*, siguiendo las mejores prácticas en cuanto a paradigmas de desarrollo de producto de estas características.

El backend se compone a su vez de tres capas de abstracción. La capa de datos o *Data Layer*, la capa de negocio o *Core Layer*, y la capa API o *API Layer*. El diseño, desarrollo, despliegue y gestión de cada una de las capas supone per sé tres proyectos diferentes, con tecnologías, lenguajes de programación, entornos, y riesgos o estándares de seguridad a aplicar diferentes en cada caso.

Como se ha desarrollado a lo largo del *Apartado 1.6 Análisis de alternativas* del *Documento Principal*, para la capa de base de datos se utilizará un motor SQL provisto por MySQL, en la capa de negocio, se empleará Java Spring Boot como stack principal, y en la capa API, un paradigma GraphQL. Asimismo, el entorno común a todo el proyecto es Amazon Web Services.

Para facilitar la presentación de la solución realizada, se ha separado la descripción de alto nivel en dos apartados. Por un lado, se presentan los componentes lógicos en los que se basa la solución para, por otro, presentar la arquitectura aplicada y basada en servicios AWS.

1.1. Arquitectura lógica

La arquitectura en la que se basa el proyecto backend desarrollado se describe en la *Imagen I: Arquitectura de alto nivel.*

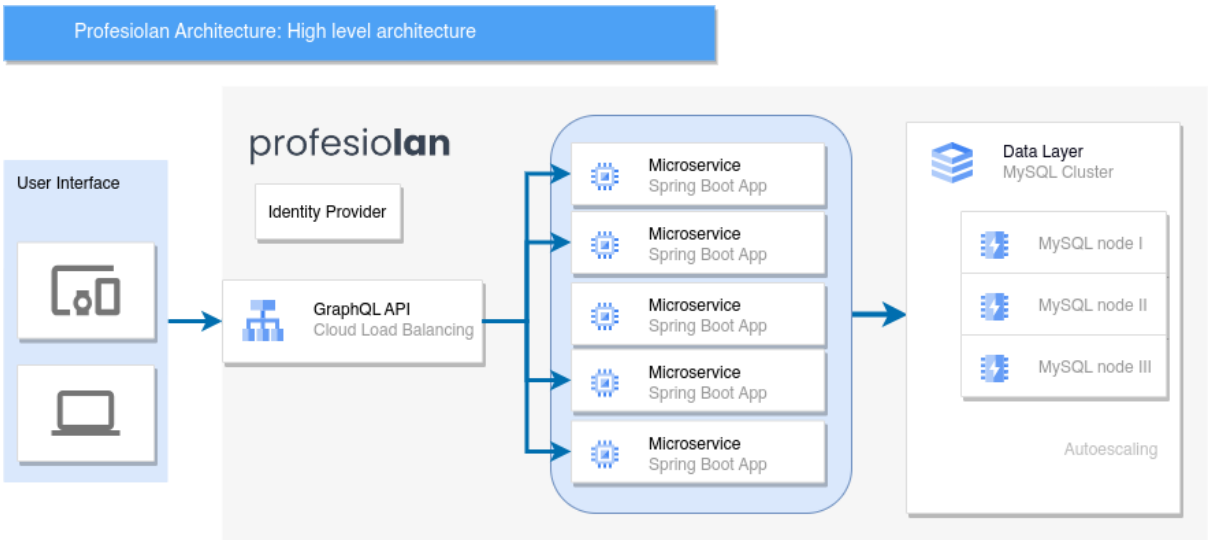

Imagen I: Arquitectura de alto nivel.

En esencia, se observa, sin entrar en la tecnología, la arquitectura modular del diseño. A la izquierda se presentan los clientes, indistintamente de que se trate de aplicación móvil o web.

Toda interacción de los clientes se realiza, de forma unificada, con la API. Es precisamente esta la que implementa las reglas de seguridad, la que realiza balanceo de cargas, la que protege los recursos a nivel de petición en función de los permisos del solicitante y, por supuesto, la que transmite la consulta al microservicio concreto en cada caso. Además, recoge toda la información de todas y cada una de las tareas que realiza para la obtención de métricas de salud. Finalmente, cuando el microservicio en concreto haya terminado la consulta, es la API la encargada de organizar la devolución de la información, en una forma estandarizada y desacoplada.

Como se ha introducido, cada microservicio tiene una función particular y acotada, y es totalmente independiente del resto. En su conjunto, alberga la totalidad de la lógica de negocio pero, de forma unitaria, es una caja negra dos únicos puntos de entrada y salida: la interfaz hacia la API y la interfaz hacia la base de datos.

Finalmente, tal y como se puede ver en la *Imagen I: Arquitectura de alto nivel,* todos los microservicios convergen, de forma lógica, en una única interfaz con la base de datos. Cabe subrayar el concepto de convergencia en este caso, y es que, mientras que la interfaz de entrada y salida de la base de datos es única, en la aplicación se establecen decenas (incluso cientas) de conexiones, cada una con su correspondiente sesión.

Se aprecia que es un diseño interesante, complejo, y que juega con el concepto del autoescalado y la ingeniería del aprovisionamiento de recursos para su beneficio. En la realidad, se ha testeado que es un sistema resiliente, robusto, rápido, escalable y diseñado para el largo plazo.

Para una mayor comprensión de la solución, se propone la lectura del *Anexo II. Descripción de la solución realizada: Diseño de bajo nivel* y el Anexo III. Análisis de rendimiento.

1.2. Arquitectura basada en Servicios AWS

Una vez contextualizado el alcance tecnológico de la solución y el diagrama lógico de la arquitectura, se procede a presentar la arquitectura aplicada en servicios AWS.

AWS, al igual que sus competidores en materia de proveedores Cloud, proponen una forma de desarrollo basada en servicios PaaS. De esta forma, cada proveedor dispone al equipo desarrollador una serie de servicios orientados a cumplir tareas específicas en la arquitectura.

Por tanto, en un proyecto real desplegado en algún proveedor cloud, la definición de tres capas pasa a ser una abstracción, pues en realidad se presenta como una armonía de servicios asíncronos interconectados por medio de interfaces y que están a su vez desplegados en entornos distribuídos en una o más redes.

Para completar el estudio de la arquitectura, en este apartado se procede a definir cada uno de los servicios referenciados en la *Imagen II: Arquitectura conceptual resumida de la solución backend con terminología de servicios AWS.*

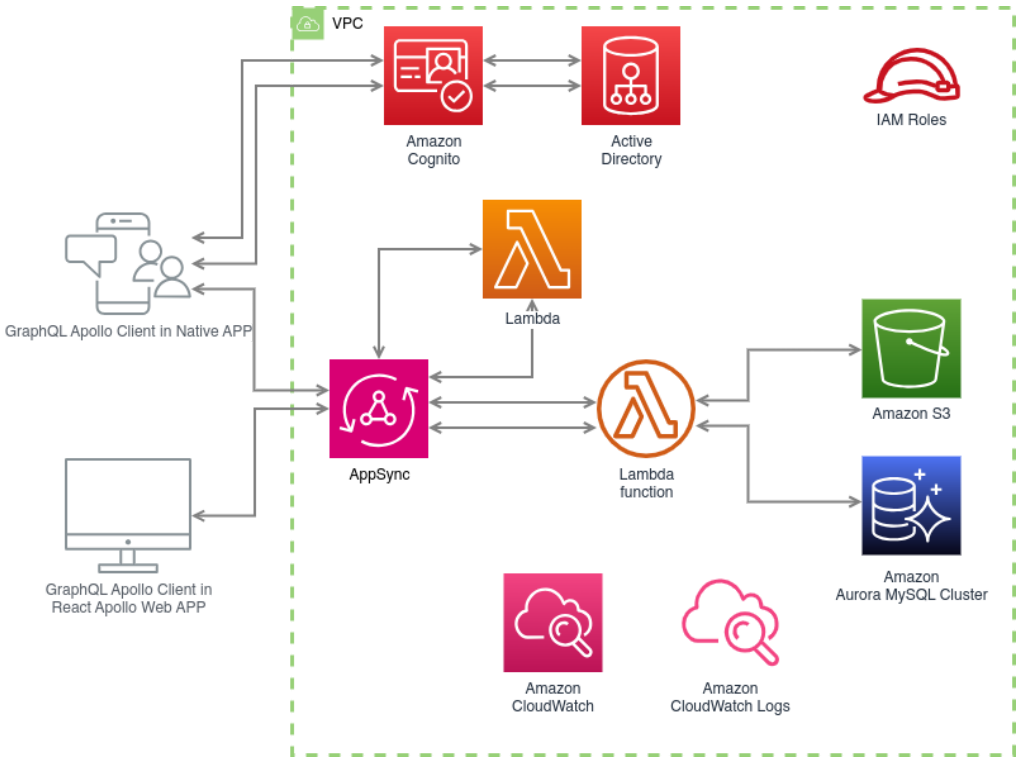

Imagen II: Arquitectura conceptual resumida de la solución backend con terminología de servicios AWS.

1.2.1. Amazon Aurora

El servicio Aurora, es la principal solución para la administración de bases de datos relacionales, sin servidores y altamente escalable de AWS. Se trata de una solución en la que se garantiza la alta disponibilidad, el autoescalado en función del consumo, la capacidad para generar réplicas de forma distribuida físicamente, segura y de muy baja latencia.

AWS Aurora es el servicio de base de datos relacional utilizado por grandes compañías como Samsung o Nintendo para sus productos desplegados.

1.2.2. Amazon AppSync

El servicio AppSync, es la principal solución para la administración y el despliegue de una API basada en GraphQL, por parte de AWS.

1.2.3. Amazon Clouwatch y Amazon Cloudwatch Logs

El servicio CloudWatch, es una de las principales soluciones para la recopilación de información y métricas de eventos en tiempo real para el correcto mantenimiento de las aplicaciones desplegadas, por parte de AWS.

Se trata de una solución integral para detectar cuellos de botella, identificar orígenes de problemas, y en general supervisar el estado de salúd de la aplicación.

1.2.4. Amazon Cognito

El servicio Cognito, es una herramienta de autenticación y autorización de AWS. Además, provee de funciones de registro e inicio de sesión para los usuarios, permitiendo una segura y confiable integración para la lógica de negocio de la aplicación.

AWS Cognito permite un almacenamiento de identidades escalable a millones de usuarios, que además incorpora la posibilidad de federación de identidades social y empresarial (inicio de sesión con Microsoft y Google entre otros).

En resumen, AWS Cognito ofrece una perfecta integración a la lógica de negocio para realizar el control de acceso a funciones de la API, en función de una serie de roles otorgados de forma ordenada por grupos de usuarios.

Todo esto es posible mediante el uso de Tokens JWT expedidos tras autenticación y autorización de los usuarios.

1.2.5. Amazon Lambda

Lambda es un servicio para el despliegue de pequeñas aplicaciones, altamente escalable, y sin servidores de AWS.

En la solución desarrollada, AWS Lambda ofrece contenedores para el despliegue de aplicaciones Java Spring Boot, que incorporan grupos de funcionalidades, y cuya separación tiene un impacto directo positivo en materia de modularidad y optimización de latencias.

1.2.6. Amazon IAM

AWS Identity and Access Management (IAM) es la solución de AWS para la gestión de identidades y permisos, integral, para todos los servicios y usuarios. IAM define quién y qué servicios tienen acceso a realizar qué operaciones sobre qué otros servicios en AWS.

AWS IAM es una solución ultra-granular, que permite, mediante código, otorgar y especificar permisos de la forma más precisa posible, y basada en una política de denegación por defecto.

En la solución desarrollada, todos los servicios desplegados incorporan un archivo en el que se especifica qué servicios y qué usuarios tienen acceso a operar sobre el mencionado servicio, además de los otros servicios terceros sobre los que el servicio en cuestión tiene el derecho de operar.

1.2.7. Amazon VPC

Amazon Virtual Private Cloud (VPC) es la solución principal para la generación y gestión de redes privadas para la arquitectura de red sobre la que se encuentra desplegada la aplicación.

AWS VPC permite diseñar, crear, desplegar y mantener redes seguras de todo tipo, mediante tecnologías de networking como NATs, y configuraciones totalmente granulares.

2. Componentes

En este momento, se exponen de forma resumida cada uno de los tres componentes. Este apartado promete ser la antesala de la lectura de los Apartados 2, 3 y 4 respectivamente del *Anexo II. Descripción de la solución realizada: Diseño de bajo nivel.*

2.1. Data Layer

Como se ha referenciado durante el documento, la capa de datos del proyecto Profesiolan está basada en un motor MySQL desplegado sobre el servicio AWS Aurora. A continuación, se manifiesta tanto la configuración desarrollada para Aurora, como el diseño de la base de datos. Además, se incluye información sobre la población de datos *dummy* inicial.

2.1.1. Configuración AWS Aurora

A continuación, se presenta la configuración establecida para el cluster de base de datos que conforma la totalidad de la llamada Data Layer.

- Versión: MySQL 5.7.2.11.3.
- SLA: https://aws.amazon.com/es/rds/aurora/sla/
- Capacidad Mínima de cada nodo del cluster: 2GiB RAM.
- Capacidad Máxima a provisionar por cada nodo del cluster: 32GiB RAM.
- Frecuencia de generación de backup: 24 horas.
- Despliegue físico: EU-WEST-2 (Londres).
- Despliegue físico multi AZ: Habilitación pospuesta a expansión territorial de la solución a otros países europeos.
- Despliegue lógico: Subred privada segura VPC.
- Entrada en hibernación tras inactividad: 5 minutos en dev., deshabilitado en producción.
- Encriptación: Habilitada.
- Acceso público: No.
    - Accesible mediante entorno de desarrollo.
    - Accesible por los microservicios, capa core.

- Escalado automático: Autogestionado por AWS en función del tráfico.
- Acceso unificado de alta disponibilidad: HTTP API habilitado para el rápido acceso a la interfaz unificada del cluster. Esto permite poder esconder la totalidad del cluster en una red privada remota, con un único acceso seguro basado en una API HTTP con un sistema de renovación de credenciales, localizado en otra red privada.

2.2. Core Layer

Como se ha referenciado durante el documento, la capa core del proyecto Profesiolan se basa en una arquitectura distribuida y modular de microservicios basados en aplicaciones Java Spring Boot, desplegadas en contenedores AWS Lambda.

En este apartado se describe, de forma resumida, el alcance de todos los microservicios desarrollados. Además, se manifiesta la configuración de los contenedores Lambda donde están desplegadas las aplicaciones.

2.2.1. Configuración por defecto de los contenedores

A continuación, se presenta la configuración establecida para el servicio PaaS AWS Lambda donde están desplegadas las aplicaciones o microservicios. Esta configuración es la desarrollada, por defecto, para todas ellas. No obstante, cada instancia está desacoplada en su totalidad de las otras, y por tanto esta configuración es modificable en función de las necesidades puntuales de cada una.

- Version: Lambda.
- SLA: https://aws.amazon.com/es/lambda/sla/.
- Runtime: Java 11.
- Arquitectura: x86_64.
- Cloudwatch Log: /aws/lambda/MgmtService.
- Memoria RAM: 512 MB. Expandible hasta 10240 MB.
- Memoria en /tmp: 512 MB
- Snapstart: Habilitado. Permite la reducción del cold time o tiempo asociado al levantamiento de un servicio tras estado en hibernación.
- Despliegue físico multi AZ: Habilitación pospuesta a expansión territorial de la solución a otros países europeos.
- Despliegue lógico: Subred privada segura VPC.
- Entrada en hibernación tras inactividad: 5 minutos en dev., deshabilitado en producción.
- Localización de claves: Contenedor seguro gestionado en Secrets Manager.
- Acceso público: No. Únicamente accesible a través de la API.

2.2.2. Presentación inicial de los servicios

La capa *Core* está compuesto por cuatro microservicios, el servicio Search, User, Stock y Mgmt.

El primer microservicio a presentar de la solución desarrollada es el denominado Servicio MGMT. En él, se centraliza gran parte de las tareas de mantenimiento y gestión del producto. El servicio de Mgmt resuelve acciones de modificación y mantenimiento relativas a la gestión de categorías y de filtros, principalmente. Por supuesto y tal y como se verá a lo largo de este apartado, se tratan de gestiones exclusivas a usuarios con permisos de administración.

En segundo lugar, se encuentra el microservicio Search, módulo encargado de realizar las tareas de búsqueda, filtrado y navegación principal de usuario. Como parece lógico, se trata del microservicio con más uso, debido a que es el que asume la total responsabilidad del filtrado y navegación por el marketplace.

A continuación, se encuentra el microservicio User. Este es el encargado de la gestión de cuentas de usuario, actualización de la información, creación y borrado de cuentas, etc. También es el utilizado por parte de la administración del negocio para la modificación de planes o asignación de ofertas.

En último lugar, el microservicio Stock es el encargado de la gestión de los activos. Módulo que posibilita la publicación, actualización y borrado de anuncios por parte de los usuarios propietarios de ese stock. Además, es el encargado de ejecutar tareas de sincronización de stock con proveedores que mantienen un gran volumen de activos en cartera.

Una vez más, se propone la lectura detenida del *Apartado 2: Core Layer,* del *Anexo II. Descripción de la solución realizada: Diseño de bajo nivel,* para una mayor profundización sobre el alcance de cada una de las funcionalidades, además de una breve explicación de la estructura de cada proyecto Java Spring Boot.

2.3. API Layer

Como se ha referenciado durante el documento, la capa API del proyecto Profesiolan está basada en la tecnología GraphQL, y ha sido desplegada sobre el servicio AWS AppSync. A continuación, se expone la configuración desarrollada para la API.

El esquema público de la API, junto con la definición de todas y cada una de las operaciones posibles, la descripción detallada de todos los objetos de entrada y salida admitidos, y la política de accesos exigida para la ejecución de cada una de las consultas se define en el *Apartado 3: API Layer* del *Anexo II. Descripción de la solución realizada: Diseño de bajo nivel.*

2.3.1. Configuración AWS AppSync

- Versión: GraphQL.
- SLA: https://aws.amazon.com/es/appsync/sla/
- Endpoint público: https://lg2uz3spjjczzpalaaanio7ndi.appsync-api.eu-west-2.amazonaws.com/graphql
- Autenticación y autorización: Habilitada.
- Sistemas de auth. adicionales: Habilitados mediante API Keys.
- Despliegue físico: EU-WEST-2 (Londres).
- Despliegue físico multi AZ: Habilitación directa.
- Logging: Habilitado. Generación de log completa a AWS Cloudwatch.
- Gestión de salud y forense: Habilitado el trazado total de las peticiones mediante AWS X-Ray, con el fin de identificación de cuellos de botella y puntos de mejora.
- Entrada en hibernación tras inactividad: No. Alta disponibilidad 24/7.
- Caché: Granularidad de aplicación excelente. Análisis desarrollado en el *Apartado 5.1.5: Gestión de la Caché.*
- Resolvers: En GraphQL, la conexión a la capa de negocio, en este caso los microservicios SpringBoot.
    - PostService: Data source al microservicio de publicaciones para renderizar los activos del marketplace, ejecutar filtrado y que, de forma global, recoge toda la funcionalidad necesaria para poder navegar en el marketplace.

- UserService: Data source al microservicio de usuarios para la gestión de estos en todos los sentidos. Desde la actualización de datos de contacto por parte del usuario autenticado como el borrado de su cuenta Profesiolan.
- StockService: Data source al microservicio de stock para la gestión de activos, donde se gestiona la publicación de activos, actualización y el borrado.