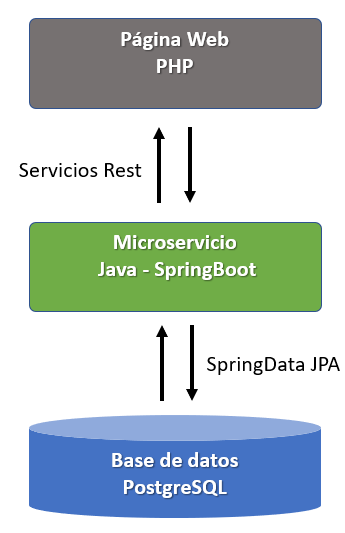
**TALLER 1.**

Se realizó la validación del historial médico de una mascota, se especificó la siguiente estructura:



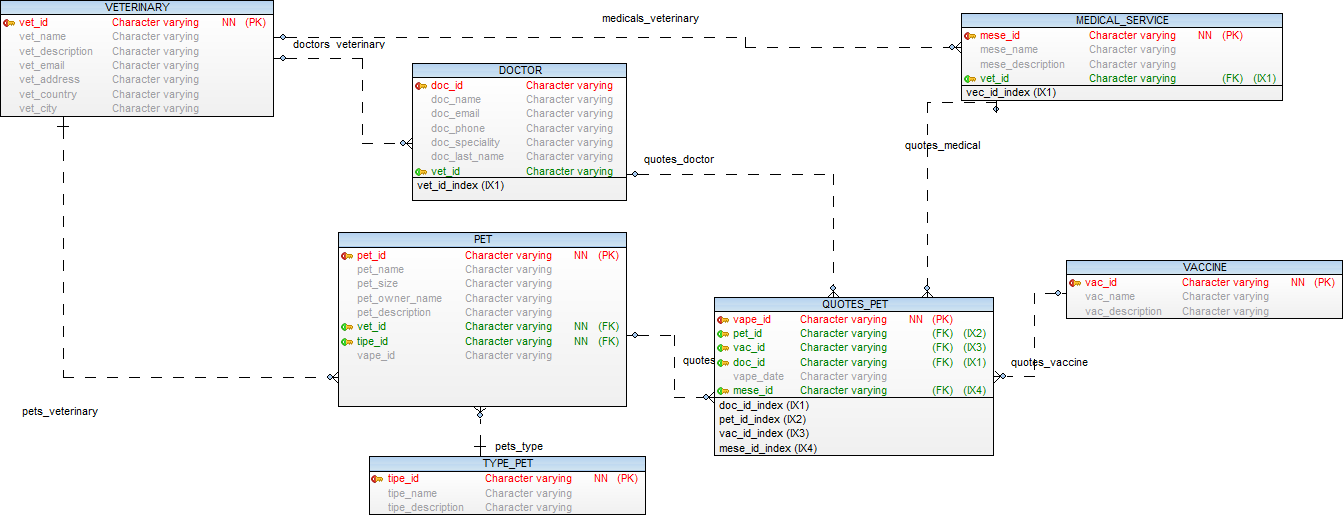
A lo anterior se realizó como microservicios, en donde el back va a ser independiente de quien lo utilice, en este caso el Front, siendo la comunicación por servicios REST.

También se optó por este modelo, ya que la página web, no solo cuenta con el *Historial de mascotas*, sino con otras opciones que no son dependientes entre sí, por ejemplo, la *Tienda de mascotas*, en este caso este sería otro microservicio que en caso de fallar no va a afectar las demás funcionalidades de la página web.

Esto también nos permite poder realizar un despliegue independiente de cada componente, siendo útil para una implementación en la nube.

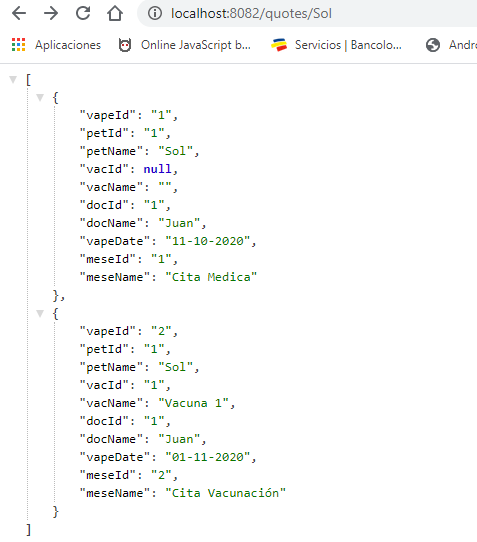
**Base de datos.**

Se realizó un modelo relacional como se pedía en la documentación y se implementó en PostgreSQL.



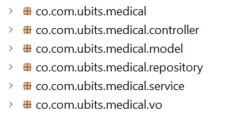
**Microservicio.**

Este se implementó en java – springboot, la idea es poder tener aquí toda la lógica de negocio y se encarga de exponer la información por medio de servicios REST y obtener la información en la base de datos comentada anteriormente.



*Servicio REST que consulta el historial de citas por nombre de mascota.*

La estructura de desarrollo dentro del microservicio es la siguiente:



Los paquetes más relevantes son:

*Controller*: Donde se exponen los servicios REST.

*Model*: Entidades de la base de datos.

*Repository*: Comunicación con la base de datos.

*Servicio*: Contiene la lógica de negocio requerida.

**Página web.**

Este se implementó en HTML y PHP, el primero para construir la estructura web y el segundo para realizar las peticiones al microservicio.

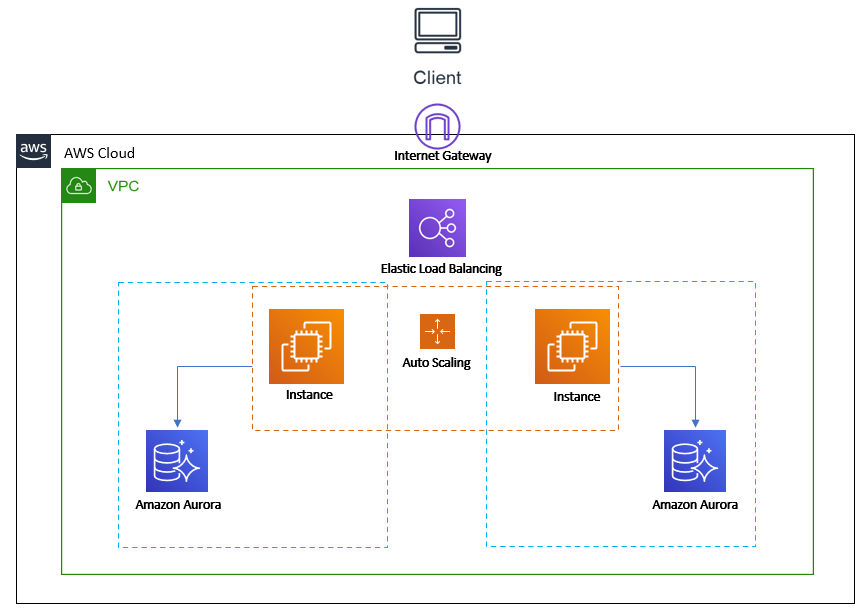
Para lo anterior solo se logró construir lo siguiente:

*consulta.html:* Consulta el nombre de la mascota.

*pets.php:* Realiza las peticiones REST al microservicio, obtiene y muestra la información de la mascota y su historial de citas.

*default.css:* Archivo de estilos.

**TALLER 2.**



La anterior es una propuesta que se establece cuando se maneja instancias EC2.

1. Las 3 principales funcionalidades que ofrece actualmente el LXP que usted considera son las más importantes.

**Facilidad para encontrar el conocimiento relevante por parte del usuario y así obtener un aprendizaje más personalizado.**

**La utilización de Bussiness Intelligence, como transformador de datos en información e información en conocimiento.**

**Entendimiento del aprendizaje como una red usuarios conectados entre sí, obteniendo una mayor capacidad de colaboración y respuestas.**

2. Los tres escenarios de atributos de calidad principales que considera son los más importantes.

**Facilidad de navegación, intuitiva.**

**Escalabilidad.**

**Seguridad.**

3. Las tres restricciones principales (relacionadas con el negocio y técnicas) que considera son las más importantes.

**Alojamiento y configuración del LXP en una instancia EC2.**

4. Los 3 problemas de diseño principales que identifica como los más importantes para resolver este desafío

**Definición de la ubicación de cada instancia EC2 a la que va a acceder el balanceador de cargas, por ejemplo, si se van a colocar en diferentes regiones.**

**Validar si la base de datos también se duplicará y si es así como se va a realizar la actualización y consistencias de los datos en cada una.**

5. Propuesta de su arquitectura AWS para solucionar los problemas de diseño que identificó previamente

**VPC: Conexión segura entre servidores y base de datos.**

**ELASTIC LOAD BALANCING: Se propone un balanceador de cargas de aplicaciones para poder dirigir el tráfico dentro de Amazon Virtual Private Cloud (VPC), proporcionando equilibrio de carga en varias instancias EC2.**

**AUTO SCALING: Poder definir una escala de recursos adicionales dependiendo de la demanda y poder mantener una disponibilidad.**

**INSTANCIA EC2: Va a contener todo lo relacionado con el negocio como la plataforma de aprendizaje, la cual a contener auto-scaling dependiendo de la demanda y un balanceador dependiendo de la cantidad de peticiones.**

**AMAZON AURORA: Base de datos que contendrá la información necesaria que provenga de las instancias EC2, además de que es una base de datos relacional compatible con MySQL y PostgreSQL.**

6. La razón de ser de cada decisión de diseño que tomó al proponer su diseño arquitectónico

**Aunque no estoy seguro si el diseño haya quedado correcto, lo hice pensando en algunos atributos de calidad como Robustez, Flexibilidad y Performance, ya que este producto (Ubits), en lo que he entendido va a seguir creciendo y es muy importante que los cambios que se sigan realizando no vayan a afectar en producción.**

**Otra decisión fue la parte económica, ya que los servicios que se encuentran en el diseño son “Se cobra lo que se usa”.**

**Por último, la mayoría de los servicios plasmados en el diseño, son de fácil configuración a través de la consola de AWS, permitiendo tener una curva de aprendizaje no tan alta.**