**MarketPlace Prueba MELI**

1) La arquitectura no está definida, entonces te damos total libertad para diseñar lo que creas conveniente. Te pedimos diseñar la arquitectura de la aplicación de forma que soporte el tráfico propuesto, mencionando pros/contras de la solución. Además, definir el backend para el storage que soporte la cantidad y el tráfico especificados.

Diagrama

Descripción generada automáticamente

**Respuesta**: Se realizan las peticiones a la base de datos por medio de Spring Framework utilizando el método de creación de base de datos con CodeFirst se maneja la lógica de negocio del lado del backend, no se manejan procedimientos almacenados, contando con la ventaja de tener versionado el código de persistencia, se utiliza un ORM para facilitar la escalabilidad en caso de necesitar cambiar de motor de base de datos, pero se tiene una desventaja y es que a mayor volumen de datos podría tener menor rendimiento por tratarse de realizar peticiones repetitivas a la base de datos, podría crearse un procedimiento almacenado para la consulta y así disminuir el tiempo de procesamiento. Se utiliza la arquitectura cliente-servidor para facilitar el consumo de la API desde distintos clientes en caso de ser requerido, es usado el patrón de diseño MVC para el proyecto web Client.

-¿Cómo medirías que la solución planteada en el punto anterior soporta los requerimientos planteados?

**RTA:** Se realizaron pruebas con postman para saber si los servicios funcionarían desde cualquier cliente. Se pueden realizar pruebas funcionales con Selenium.

- ¿Qué variables tendrías en cuenta al momento de decidir si tu arquitectura debería ser on-premise o cloud?

**Respuesta:** Las variables que tendría en cuenta serían:

* Presupuesto: En un modelo on-premise es necesario adquirir servidores físicos, también algunas empresas cloud permiten pagar una cuota fija por todos los servicios necesarios.
* Disponibilidad: Es más probable tener un problema con el servicio en una arquitectura on-premise que con cloud.
* Mantenimiento: Para una arquitectura on-premise los servidores están en físico así que deben ser mantenidos por la propria empresa y son costos de manutención, en cloud este costo se elimina ya que el mantenimiento corre por cuenta del proveedor.
* Seguridad: Teniendo servidores propios no habrá fuga de información, en una arquitectura cloud dependeremos de la seguridad del proveedor.

2) Se debe construir un servicio REST para ITEMS que será el encargado de exponer las funcionalidades (endpoints) de Creación, Consulta, Actualización y Eliminación de registros de una base de datos. (Proponga un modelo de datos).

-Describa cómo estructuraría el proyecto del servicio REST (Monolítico, n-capas, cloud, etc), los patrones de diseño que quizá usaría, protocolos de seguridad, logs, etc.

**Respuesta:** Creación de API en el backend para no depender de un proyecto cliente, así cualquier cliente puede consumir los diferentes servicios publicados vía REST. Se creó una capa de controlador para crear los endpoints con su respectiva ruta para el llamado desde cualquier cliente, se creó una capa de modelo para las clases necesarias y así crear la base de datos con ayuda de JPA, y una capa de servicios la cual será la encargada de tener toda la funcionalidad de los endpoints. Se agregaron algunos logs para el proyecto cliente que puedan visualizarse en la web cuando está en ejecución, podrían agregarse también para el lado del backend por cada uno de los endpoints.

- Es libre de escoger el lenguaje de programación donde se pueda ver la estructura del proyecto y al menos un endpoint del CRUD. (Implementar código).

**RTA:** Para el proyecto cliente se utilizó HTML, Typescript y Angular, para el diseño de interfaces de usuario se utilizaron las librerías Boostrap y primeng. Del lado del backend es usado el framework Spring Boo, lenguaje Java, JPA y como base de datos tenemos PostgreSQL.

**Modelo de datos**

Diagrama

Descripción generada automáticamente