Respuesta a las preguntas indicadas

1. ¿Qué ventajas tiene hacer este programa en un modelo de arquitectura de n capas?
   1. Separación de responsabilidades: Permite la separación de responsabilidades, ya que cada capa tiene una sola responsabilidad.
   2. Permite una facilidad en el desarrollo: Este estilo arquitectónico es especialmente fácil de implementar, además de que es muy conocido y una gran mayoría de las aplicaciones la utilizan.
   3. Fácil de probar: Debido a que la aplicación construida por capas es posible ir probando de forma individual cada capa, lo que permite probar por separada cada capa.
   4. Mantenimiento: Debido a que cada capa hace una tarea muy específica, es fácil detectar el origen de un bug para corregirlo, o simplemente se puede identificar donde se debe aplicar un cambio.
   5. Seguridad: La separación de capas permite el aislamiento de los servidores en subredes diferentes, lo que hace más difícil realizar ataques.
2. ¿Qué desventajas tiene hacer este programa en un modelo de microservicios?

Ventajas

* 1. Alta escalabilidad: Los Microservicios es un estilo arquitectónico diseñado para ser escalable, pues permite montar numerosas instancias del mismo componente y balancear la carga entre todas las instancias.
  2. Agilidad: Debido a que cada Microservicios es un proyecto independiente, permite que el componente tenga ciclo de desarrollo diferente del resto, lo que permite que se puedan hacer despliegues rápidos a producción sin afectar al resto de componentes.
  3. Facilidad de despliegue: Las aplicaciones desarrolladas como Microservicios encapsulan todo su entorno de ejecución, lo que les permite ser desplegadas sin necesidad de dependencias externas o requerimientos específicos de Hardware.
  4. Facilidad para realizar Test: Los Microservicios son especialmente fáciles de probar, pues su funcionalidad es tan reducida que no requiere mucho esfuerzo, además, su naturaleza de exponer o brindar servicios hace que sea más fácil de crear casos específicos para probar esos servicios.
  5. Fácil de desarrollar: Debido a que los Microservicios tiene un alcance muy corto, es fácil para un programador comprender el alcance del componente, además, cada Microservicios puede ser desarrollado por una sola persona o un equipo de trabajo muy reducido.
  6. Reusabilidad: La reusabilidad es la médula espinal de la arquitectura de Microservicios, pues se basa en la creación de pequeños componentes que realice una única tarea, lo que hace que sea muy fácil de reutilizar por otras aplicaciones o Microservicios.
  7. Interoperabilidad: Debido a que los Microservicios utilizan estándares abiertos y ligeros para comunicarse, hace que cualquier aplicación o componente pueda comunicarse con ellos, sin importar en que tecnología está desarrollado.

Desventajas

* 1. Performance: La naturaleza distribuida de los Microservicios agrega una latencia significativa que puede ser un impedimento para aplicaciones donde el performance es lo más importante, por otra parte, la comunicación por la red puede llegar a ser incluso más tardado que el proceso en sí.
  2. Múltiples puntos de falla: La arquitectura distribuida de los Microservicios hace que los puntos de falla de una aplicación se multipliquen, pues cada comunicación entre Microservicios tiene una posibilidad de fallar, lo cual hay que gestionar adecuadamente.
  3. Trazabilidad: La naturaleza distribuida de los Microservicios complica recuperar y realizar una traza completa de la ejecución de un proceso, pues cada Microservicio arroja de forma separa su traza o logs que luego deben de ser recopilados y unificados para tener una traza completa.
  4. Madurez del equipo de desarrollo: Una arquitectura de Microservicios debe ser implementada por un equipo maduro de desarrollo y con un tamaño adecuado, pues los Microservicios agregan muchos componentes que deben ser administrados, lo que puede ser muy complicado para equipo poco maduros.

1. ¿Si tuviera que implementar un sistema de autenticación y autorización como lo haría?
   1. Autenticación: Seria utilizando una integración con el sistema de administración de usuarios central. Ejemplo LDAP, para garantizar que los usuarios que puedan ingresar al sistema sean usuarios habilitados para el ingreso
   2. Autorización: Usaría un módulo de roles y perfiles para la definición de cada una de las acciones habilitadas para cada rol. Adicionalmente utilizaría el JWT lo que permitirá al usuario acceder a rutas, servicios y recursos permitidos con ese token
2. ¿En este proyecto que tan importante es la Confidencialidad y la Integridad? ¿Por qué?
   1. Confidencialidad: Seria utilizando una integración con el sistema de administración de usuarios central. Ejemplo LDAP, para garantizar que los usuarios que puedan ingresar al sistema sean usuarios habilitados para el ingreso
   2. Integridad: Usaría un módulo de roles y perfiles para la definición de cada una de las acciones habilitadas para cada rol. Adicionalmente utilizaría el JWT lo que permitirá al usuario acceder a rutas, servicios y recursos permitidos con ese token
3. ¿Cómo calcularía la disponibilidad de la implementación que usted realizó?

Disponibilidad es el porcentaje de tiempo que una carga de trabajo está disponible para su uso (Referencia AWS)

Disponibilidad = Disponibilidad por Tiempo de Uso / Tiempo Total

1. Si tuviera 3 desarrolladores para desarrollar esta aplicación, ¿Cómo les distribuiría las tareas?

Después de realizar el respectivo desglose de las tareas durante el spring planning asignaría cada desarrollador para que trabaje en el desarrollo de los componentes de cada una de las capas de arquitectura. Ejemplo: Un desarrollador para realizar el front, otro desarrollador para construir la parte del backend, y otro desarrollador que se encargue de los componentes de la capa de persistencia. Adicionalmente dividiría las actividades de apoyo como la creación de repositorio, aprobación de pull request y merge. Trabajar en todas las actividades de Devops, etc.

1. Si hay que generar un reporte, ¿Cómo lo haría?

Si es a nivel del trabajo realizado en la metodología, me apoyaría en los dashboard que proveen las diferentes herramientas de trabajo que se utilizan tales como jira, vst, o suite Kanban. Con estos dashboard presentaría el avance del trabajo realizado; con apoyo de los desarrolladores generaría información acerca de impedimentos, logros más importantes.

Si son reportes de datos, se puede utilizar una herramienta como Power BI que permite la integración de diferentes orígenes de datos y permite generar informes dinámicos, atractivos.

1. Hay que cifrar algunos datos en la aplicación. ¿Cómo lo haría?
2. Garantizando el uso de protocolos de seguridad
3. Utilizando en java paquetes de seguridad para el cifrado de datos ya sea RSA o SHA.