Qüestionari / Exercici Disseny SI basat en una Arquitectura en 3 Capes

0. Decisions generals d'arquitectura

En primer lloc, cal garantir que cadascuna de les tres capes tingui responsabilitats ben definides: la capa de presentació s'encarregarà de la interacció amb l'usuari, la capa de domini gestionarà la lògica de negoci, i la capa de gestió de dades s'ocuparà de l'emmagatzematge i l'accés a les dades persistents. És crucial definir els mecanismes de comunicació entre capes, utilitzant objectes de transferència de dades (DTOs) per enviar informació i establir una API entre la presentació i el domini. També cal decidir patrons d'integració per a la gestió de dades, i serveis per encapsular la lògica de negoci. És important establir un esquema de versions per mantenir la compatibilitat entre capes durant actualitzacions.

1. Decisions de disseny de la capa de Presentació

En aquesta capa, s'ha de decidir el tipus d'aplicació que es desenvoluparà, ja sigui una aplicació web, mòbil o d'escriptori. També cal escollir els frameworks i tecnologies, com per exemple React, Angular, Vue.js per a aplicacions web, o Flutter per aplicacions mòbils. La comunicació amb el backend ha de ser mitjançant protocols com REST o GraphQL, i s'han d'utilitzar eines com Fetch API per gestionar les crides. Una altra decisió clau és com es gestionarà l'estat del frontend. Pel que fa a la seguretat, s'ha de suportar l'autenticació, per exemple amb OAuth o JWT, i definir com es gestionaran les sessions d'usuari. A més, s'ha de tenir en compte l'accessibilitat, seguint estàndards WCAG, i preveure internacionalització per permetre diversos idiomes.

2. Decisions de disseny de la capa de Domini

En la capa de domini, cal identificar les entitats i agregats que representen la lògica de negoci i encapsular-les seguint patrons com Service Layer. La gestió de transaccions ha de garantir la consistència de les operacions que involucren múltiples entitats, fent servir gestors de transaccions com els de Spring. S'ha de dissenyar la capa per facilitar l'escalabilitat i la modularitat, amb serveis separats si és necessari. També cal assegurar-se que les validacions de dades i regles de negoci estiguin ben definides dins del model. Per garantir la qualitat, calen proves unitaris i d'integració utilitzant eines com JUnit. A més, cal decidir com es gestionarà la comunicació amb sistemes externs mitjançant serveis web o APIs.

3. Decisions de disseny de la capa de Gestió de Dades

En aquesta capa, cal decidir el tipus de base de dades a utilitzar: si serà una base de dades relacional com MySQL o PostgreSQL, o una base NoSQL com MongoDB. També s'ha d'escollir si s'utilitzarà un ORM com Hibernate o accés directe mitjançant SQL. És important dissenyar l'esquema de la base de dades, incloent-hi claus primàries, índexs i relacions entre taules. A més, per millorar el rendiment, es poden implementar mecanismes de caching com Redis. Pel que fa a la seguretat, s'han de protegir dades sensibles mitjançant xifratge i configurar correctament les credencials d'accés.

4. Altres decisions de disseny

Finalment, és important planificar estratègies DevOps que facilitin el desplegament continu, aprofitant eines com Docker per a la contenció de serveis i pipelines mitjançant plataformes com GitHub Actions. També cal implementar solucions per monitoritzar el sistema, com Prometheus, que permetin supervisar el rendiment i detectar anomalies. Per últim, resulta fonamental establir mecanismes d'auditoria i traçabilitat que assegurin el registre detallat de les operacions i possibles incidències.