Microservices

En monolitisk arkitektur kan man se som en enda stor enhet av tjänster och funktioner för en applikation(en och samma kodbas). Medans microservice arkitektur bryter ner allt i mindre oberoende enheter och tjänster. Dessa enheter utför varje ansökningsprocess som en separat tjänst. Så alla enheter kan ha sin egna logik, databas samt utför specifika funktioner.

Fördelar
Förhindra eller minska risken för låsning till t.ex leverantör eller teknik val. Microservices ger flexibiliteten att testa ny teknik på en individuell tjänst, då det inte kommer att finnas så många beroenden samt mindre storlek på kodbasen, och att ändra tillbaka förändringar blir mycket lättare.

Lättare för utvecklare att förstå services/funktioner, då allt är uppdelat.
Mindre och snabbare deployments, just pga mindre kodbas. Därav också fördelen med att kunna implementera Continuous Integration och Continuous Delivery/Deployment.

En annan fördel är skalbarhet, eftersom tjänsterna är separata kan man lättare skala de mest nödvändiga tjänsterna vid lämpliga tidpunkter, i motsats till en monolitisk arkitektur där man behöver skala hela applikationen.

Nackdelar
Kommunikation mellan tjänster kan bli komplexa, eftersom allt nu är en oberoende tjänst måste du noggrant hantera förfrågningar mellan modulerna/enheterna.

Felsökning av problem kan vara svårare, varje tjänst har sina egna loggar.

Distribuerade system och processflöden kan skapa komplicerade tester när du ska testa hela applikationen.

Ett simpelt exempel på en microservice arkitektur är om du ska bygga en e-shop applikation, du delar då upp alla funktioner i olika microservices, t.ex.

- Inloggning
- Sökning av produkter
- Lägg i varukorg
- Köp
- Filtrering
- Recensioner
- Osv.

Integration Patterns

API Gateway Pattern

Om du har olika kanaler som desktop-client, mobil app och hemsida. Alla dessa behöver oftast olika data för att svara från samma backend-tjänst, eftersom t.ex UI kan vara annorlunda på de olika enheterna. Då är en API-Gateway en bra lösning som alla request går igenom för att senare hänvisa till korrekt microservice och hämta korrekt data.

Database Patterns

Database per Service

Det finns olika sätt att implementera så att en microservice har sin egen data.
Man kan t.ex ha en databas som flera microservices kan hämta data från, men de kan bara hämta från specifika tabeller som är låsta till sig. Eller så skapar man en hel databas för en microservice. Private-tables-per-service, Schema-per-service, eller Database-server-per-service