APUNTS D'EXAMEN

1. Desenvolupament

1.1. Què és SPRING BOOT?

És una extensió del framework SPRING que ha arribat a ser l'estàndard de la indústria per la seva estabilitat, la base comunitària i la seva extensibilitat.

1.2. Característiques principals de SPRING BOOT

- Està basat en JAVA.
- És molt robust.
- És autoconfigurable, la qual cosa simplifica el seu desplegament.
- Starter dependencies: són paquets pre-configurats amb llibreries comunes.
- Servidor embegut com Tomcat o Jetty, que permet executar-lo de forma stand-alone.
- Production ready: proporciona observabilitat i health check "out of the box".
- Injecció de dependències.

1.3. Elements principals d’arquitectura de SPRING BOOT

- S’empaqueten en JAR autocontinguts que es poden executar de manera separada o en un servidor Spring.
- A nivell lògic utilitza una arquitectura: Controller → Service → DAO → DB.
- Es basa en els components principals Spring Core, SPRING MVC, SPRING DATA JPA i SPRING SECURITY.

1.4. Què és QUARKUS?

És un framework per a la implementació d'APIs optimitzat per a temps de resposta i consum de memòria.

1.5. Característiques principals de QUARKUS

- Cloud first.
- Arrenca en pocs mil·lisegons i consumeix poca memòria.
- Compilació nativa amb GraalVM amb configuració embedded.

- Live coding: permet recompilar i executar en viu.
- Reactive programming: enfocat en esdeveniments.

1.6. Elements principals d’arquitectura de QUARKUS

- GraalVM per compilar.
- Injecció de dependències.

1.7. Què és MYSQL?

És un RDBMS obert i gratuït, adquirit per ORACLE.

1.8. Característiques principals de MYSQL

- Open source, fiable amb versions TLS, àmpliament utilitzat.
- Multiplataforma.
- Seguretat robusta amb controls d’accés, gestió d’usuaris i permisos.
- Alt rendiment optimitzat en lectura.
- Llenguatges SQL i suport per a molts llenguatges de programació.
- Ús estès en la web amb LAMP.
- Flexibilitat de motor d’emmagatzematge.
- Comunitat gran.
- Versió Enterprise de pagament amb funcionalitats de load balancing i seguretat estesa.
- MARIADB és un fork open source.

1.9. Elements principals d’arquitectura de MYSQL

- **Servidor**: és el procés principal que gestiona totes les operacions.
- **Motor d’emmagatzematge INNODB**: motor específic per a emmagatzemar i gestionar dades.
- **Sqldump**: eina per fer backups.
- **Mysql**: client per a consulta i gestió de la base de dades.

1.10. Què és ORACLE?

És un RDBMS privatiu molt estable, segur i escalable amb tarificació per cores.

1.11. Característiques principals d’ORACLE

- Escalabilitat: gestiona grans volums de dades i usuaris.
- Alta disponibilitat: replicació de dades, conmutació per error i recuperació de desastres.
- Seguretat avançada: autenticació, xifratge, auditoria automàtica, control granular de rols i permisos.
- Multiplataforma.
- Llenguatges PL/SQL.
- Automatització i gestió autònoma.
- Alt rendiment i disponibilitat.

1.12. Elements principals d’arquitectura d’ORACLE

- **Instance**: el servidor en execució que gestiona la comunicació i processos (similar al server en MySQL).
- **Database**: fitxers de dades (datafile en MySQL).
- **Schema**: conjunt d’objectes de la base de dades d’un usuari (concepte similar a database en MySQL).
- **Tablespace**: unitat lògica d’emmagatzematge que agrupa fitxers de dades.

1.13. Què és APACHE2?

És un servidor web molt popular, open source, modular i segur.

1.14. Característiques principals d’APACHE2

- Arquitectura modular: estructura extensible mitjançant mòduls.
- Virtual hosts: permet gestionar múltiples servidors virtuals en un mateix servidor.
- Suport SSL/TLS per a connexions segures.
- Load balancing per distribució de càrrega.
- Multiplataforma.
- Suport per a contingut estàtic i dinàmic.

1.15. Elements principals d’arquitectura d’APACHE2

- **Nucli (procés)**: gestiona les operacions bàsiques del servidor.
- **Mòduls**: unitats funcionals independents que proporcionen serveis específics i extensen la funcionalitat del servidor.

- **Directoris i fitxers de configuració**: contenen i organitzen la configuració del servidor.
- **Virtual hosts**: permeten allotjar múltiples dominis o webs en un mateix servidor físic.

1.16. Què és NGINX?

És un servidor web lleuger, open source, enfocat a contenidors i alta concurrència.

1.17. Característiques principals de NGINX

- Container first: dissenyat per a entorns amb contenidors, arrencada ràpida.
- Observabilitat i health check integrats.
- Cloud first: preparat per desplegar-se en entorns Cloud de tipus IaaS.
- Reverse proxy (proxy invers) per gestionar connexions i redireccions.
- Arquitectura orientada a esdeveniments (event driven) per eficiència.
- Load balancing distribuït.
- Caching per millorar rendiment.

1.18. Elements principals d'arquitectura de NGINX

- **Procés Master**: gestor central de configuració i supervisió.
- **Processos Worker**: executen les peticions de forma asíncrona i multiplexada.
- Arquitectura event driven: gestiona múltiples connexions de forma eficient.
- Bloc de configuració: defineix regles i comportaments mitjançant fitxers textuals.
- Gestió de continguts: serveix contingut estàtic i redirigeix peticions a contingut dinàmic.

1.19. Què és APACHE TOMCAT?

És un servidor d'aplicacions Java J2EE, open source i estàndard de la indústria.

1.20. Característiques principals d'APACHE TOMCAT

- Implementa Java Servlet i JavaServer Pages (JSP).
- Servidor web autònom o integrat amb Apache2.
- Alta disponibilitat amb suport a clustering.
- Suport per a aplicacions empresarials amb alta concurrència, autenticació i seguretat.
- Arquitectura modular dissenyada per components.

1.21. Elements principals d’arquitectura d’APACHE TOMCAT

- **Server**: procés d’alt nivell que administra serveis.
- **Service**: enllaça connectors amb un engine i gestiona la comunicació HTTP o AJP.
- **Engine**: processa sol·licituds i determina el host virtual corresponent.
- **Host**: domini virtual que conté les aplicacions web.
- **Servlet Container**: component que executa aplicacions web Java.
- **Connector**: gestiona la comunicació client-servidor, per exemple Coyote (HTTP) i JK2 (AJP).
- **Context**: representa una aplicació web desplegada dins d’un host.
- **JSP Engine**: processa pàgines JSP.
- **JNDI Support**: suport a Java Naming and Directory Interface.
- **Connection Pooling**: gestiona les connexions a bases de dades.

1.22. Què és JBOSS EAP?

És un servidor d’aplicacions Java J2EE open source desenvolupat per Red Hat, robust per aplicacions empresarials.

1.23. Característiques principals de JBOSS EAP

- Compliment Java EE.
- Clustering per alta disponibilitat.
- Consola de gestió per administrar el servidor.
- Arquitectura modular amb mòduls carregables dinàmicament.
- Seguretat integrada i gestió avançada de sessions.

1.24. Elements principals d’arquitectura de JBOSS EAP

- **Contenidor modular (Modules)**: sistema que permet la càrrega selectiva i modular de serveis.
- **Subsistemes**: gestionen funcions com transaccions, persistència i seguretat.
- **Servidor d’aplicacions**: executa Enterprise Java Beans (EJB) i components Java EE.
- **CDI (Contexts and Dependency Injection)**: motor per a la injecció de dependències dins d'aplicacions.
- **ORM/Hibernate**: framework per a la gestió d’accés i persistència de dades en bases SQL.
- **Servidor web integrat**: utilitza Apache Tomcat.

- **Clustering i alta disponibilitat**: suporta cache distribuït i gestió de sessions persistents.
- **Gestió i monitoratge**: consola d'administració i APIs JMX per monitoritzar el servidor.
- **Mòduls de seguretat**: proporcionen autenticació i autorització robusta.

1.25. Què és GIT?

És un sistema de control de versions distribuït (VCDS) utilitzat per gestionar canvis en codi font.

1.26. Característiques principals de GIT

- Distribuït i descentralitzat: cada desenvolupador té una còpia completa del codi.
- Rastreig precís de canvis: registre detallat de cada commit.
- Ramificació (branches) eficient i lleugera per desenvolupar noves funcions paral·lelament.
- Fusió de branques simplificada.
- Alt rendiment i eficiència en la gestió d'històrics.
- Segur i fiable.
- Suport per a col·laboració distribuïda.
- Versàtil per a diferents tipus de projectes.

1.27. Elements principals d'arquitectura de GIT

- **Repositori**: emmagatzema fitxers i historial del projecte.
- **Arbre de treball (Working Directory)**: còpia del codi per modificar localment.
- **Àrea d'indexació (Staging Area)**: lloc per preparar canvis abans de confirmar (commit).
- **Commit**: instantània de l'estat del projecte amb canvis registrats.
- **Objectes Git**: estructures internes com blob, tree, commit i tag que emmagatzemen contingut i informació.
- **Branches**: punters a commits que permeten desenvolupaments independents.
- **Referències (Refs)**: apunten a diferents punts dins del repositori i faciliten l'accés.

1.28. Què és GITHUB?

És una plataforma web per allotjar repositoris GIT que facilita la col·laboració i la gestió de projectes de programari.

1.29. Característiques principals de GITHUB

- Suporta repositoris públics i privats.
- Facilita la col·laboració mitjançant pull requests i revisió de codi.
- Integració CI/CD mitjançant GitHub Actions.
- Sistema de seguiment de problemes (issues).
- Comunitat molt activa i projectes open source massius.
- Seguretat avançada amb permisos detallats i autenticació de doble factor.
- GitHub Pages per allotjament de pàgines i wikis.

1.30. Elements principals d’arquitectura de GITHUB

- Plataforma SaaS (Software as a Service).
- Connectors per integrar amb solucions corporatives CI/CD.

1.31. Què és GITFLOW?

És un model estricte de flux de treball per a GIT que organitza el desenvolupament amb ramificacions ben definides per facilitar la col·laboració i el control de versions.

1.32. Característiques principals de GITFLOW

- Dues branques principals: **main** i **develop**.
- **Feature branches**: per noves funcionalitats, creades a partir de develop.
- **Release branches**: per fer proves i corregir errors abans d’integrar les noves versions a main.
- **Hotfix branches**: per correccions crítiques directament sobre main.
- Proporciona un aïllament clar del treball en curs.
- Ideal per a projectes amb ciclicitat complexa i equips paral·lels.
- Permet un cicle d’entrega controlat i segur.

---

2. Arquitectures

2.1 Què és AWS?

Amazon Web Services és una plataforma de cloud computing més utilitzada.

2.2 Característiques principals de AWS

- EC2 (Elastic Compute Cloud): màquines virtuals escalables.
- Lambda: computació serverless per a funcions.
- ECS/EKS: contenidors i kubernetes gestionats.
- Elastic Beanstalk: PaaS multillenguatge.
- S3 (Simple Storage Service): emmagatzematge d'objectes en buckets.
- EBS (Elastic Block Store): emmagatzematge en blocs per a EC2.
- EFS (Elastic File System): sistema de fitxers compartit.
- RDS (Relational Database Service): bases de dades relacionals.
- DynamoDB: base de dades NoSQL.
- ElasticCache: cache en memòria (Redis/Memcached).
- VPC (Virtual Private Cloud): xarxa virtual aïllada.
- CloudFront: CDN (Content Delivery Network) proxy cache.
- Route 53: DNS gestionat.
- ELB (Elastic Load Balancer): balancejadors de càrrega.

2.3 Què és GCP?

Google Cloud Platform és el servei cloud de Google.

2.4 Característiques de GCP

- Compute Engine: màquines virtuals.
- Cloud Functions: computació serverless.
- GKE (Google Kubernetes Engine): kubernetes gestionats.
- App Engine: PaaS.
- Cloud Storage: emmagatzematge d'objectes.
- Persistent Disk: discos bloc per a VMs.
- Cloud Filestorage: sistema de fitxers NFS.
- Cloud SQL: bases de dades relacionals gestionades.
- Cloud Spanner: base de dades distribuïda.
- Firestore: bases de dades NoSQL.
- VPC: xarxa privada virtual.
- Cloud CDN (Content Delivery Network).
- Cloud DNS: DNS.
- Cloud Load Balancing: balancejador de càrrega.

2.5 Què és Docker?

És una plataforma de contenidorització efímera que permet empaquetar aplicacions i les seves dependències en contenidors lleugers.

2.6 Característiques de Docker

- Contenidors lleugers i aïllats: són més lleugers que les VM perquè es basen en versions simplificades de SO.
- Portabilitat: com contenen totes les dependències, es poden executar en qualsevol plataforma.
- Imatges basades en capes: que es poden reutilitzar, estalviant espai i facilitant la distribució incremental.
- Gestió i execució eficient: Docker daemon s'encarrega de parar, iniciar i destruir els contenidors.
- Volums de dades: que es connecten amb els contenidors efímers per mantenir la persistència.
- Ecosistema i integració: dins de kubernetes.

2.7 Elements principals d'arquitectura de Docker

- Host Docker: és la màquina física on s'executa Docker Engine.
- Docker Engine: el procés nucli de Docker format per tres components:
  - dockerd: el daemon que executa les ordres i gestiona imatges, contenidors, xarxes, etc.
  - API REST: que comunica el daemon amb el client de Docker.
  - Docker Client (CLI): interfície d'usuari.
- Imatge: plantilles readonly construïdes en capes amb el sistema operatiu, dependències, la configuració i l'aplicació.
- Contenidor: instància lleugera, aïllada i efímera de la imatge.
- Docker Registry: repositori d'imatges Docker.

2.8 Què és Kubernetes?

És un sistema d'orquestració de contenidors que automatitza el desplegament, l'escalat i la gestió de contenidors.

2.9 Característiques de Kubernetes

- Orquestració: organitza i distribueix els contenidors entre els treballadors.

- Escalabilitat automàtica: del nombre de contenidors en execució en funció de mètriques.
- Alta disponibilitat: detecta caigudes i recupera automàticament.
- Actualització i rollback en calent: actualitza versions de contenidors sense temps de parada.
- Autodetecció de serveis i balanceig: proporciona un DNS i IP interna, i balanceja la càrrega entre contenidors.
- Portabilitat: multiplataforma.
- Gestió de secrets: evita guardar informació en imatges.
- Ecosistema i comunitat.

2.10 Elements principals d'arquitectura de Kubernetes

- kube-apiserver: punt d'entrada API REST que gestiona ordres d'entrada.
- etcd: base de dades distribuïda amb la configuració i l'estat del clúster.
- kube-scheduler: decideix en quin node s'executarà un pod segons disponibilitat.
- kube-controller-manager: gestiona els controladors que mantenen l'estat del clúster.
- Cloud Controller Manager: interacciona amb els serveis del proveïdor cloud.
- kubelet: agent que s'executa en cada node i monitoritza els pods.
- kube-proxy: administra la xarxa i el balanç de càrrega dins del node, gestiona el trànsit cap als serveis i els pods.

2.11 Què és WSO2?

És una plataforma open source que gestiona tot el cicle de vida de les APIs: crear, publicar, governar, assegurar, monetitzar, observar les APIs.

2.12 Característiques de WSO2

- Disseny i creació API-First: APIs amb OpenAPI (Swagger).
- Publicació i portal de desenvolupadors: facilita la publicació amb un portal interactiu per descobrir, provar i subscriure una API.
- Seguretat i control d'accés: implementa OAuth2, JWT, API keys i control de permisos a nivell granular.
- Gestió del trànsit i qualitat del servei: inclou limitació de rate, polítiques de priorització i gestió del rendiment.
- Monetització i anàlisi: permet crear models de pagament, supervisar l'ús de les APIs i dades analítiques per la presa de decisions.

- Governança i lifecycle management: controla tota la vida útil, versions, actualitzacions, retirada i continuous delivery de les APIs.

2.13 Elements principals d’arquitectura WSO2

- API Publisher: crea, publica i gestiona les APIs; definició de recursos i paràmetres; pòlitiques de seguretat; gestió de versions.
- API Developer Portal: catàleg d’APIs, descobrir, provar APIs, documentació i monitorització d’ús.
- API Gateway: executa les APIs, autentica i autoritza, aplica rate limit, transformació de missatges, logging i anàlisi.
- Key Manager: gestió de tokens d’autenticació i autorització (OAuth, JWT), valida l’accés a les APIs i proporciona endpoints de renovació de tokens.

2.14 Què és microservei?

És un enfocament d’arquitectura de desenvolupament d’aplicacions com a conjunt de serveis petits i independents.

2.15 Característiques de microserveis

- Single responsibility: cada microservei té una única responsabilitat, Domain Driven Design.
- Descentralització: cada servei és independent amb bases de dades diferents, no tenen per què córrer en el mateix servidor o cloud.
- Comunicació via APIs: normalment HTTP REST, GraphQL o gRPC binària de gran rendiment.
- Deployment independent: es poden desplegar de manera autònoma sense que afecti els altres microserveis.
- Technology diversity: cadascun pot estar fet amb una tecnologia diferent.
- Escalabilitat independent segons necessitat.
- Desenvolupament i desplegament àgils.
- Resiliència a fallades.
- Equips petits, problemes simplificats.
- Complexitat de xarxa i comunicació.
- Gestió de dades més distribuïda.
- Monitorització i debugging més complex.

2.16 Què és Apache Kafka?

És una plataforma de streaming distribuïda que permet publicar, subscriure, emmagatzemar i processar streams de dades en temps real.

2.17 Característiques d'Apache Kafka

- Alta escalabilitat: horitzontal afegint brokers per gestionar TB de dades sense degradació de servei.
- Alta disponibilitat i resiliència: amb la replicació de particions entre múltiples brokers per si un d'ells falla.
- Baixa latència: de pocs mil·lisegons per missatge, ideal per a processament en temps real.
- Persistent storage: de missatges en disc en ordre dins de cada partició.
- Processament en streaming: amb la seva API integrada que transforma i processa.
- Model publicació-suscripció: de diversos productors a diversos consumidors.
- Integracions i ampli ecosistema: Apache Spark, Flink, Apache NiFi.
- Autogestió amb KRaft: redueix latències eliminant la necessitat d'Apache ZooKeeper.

2.18 Principals elements d'arquitectura d'Apache Kafka

- Brokers: servidors que formen el clúster, emmagatzemen i gestionen les dades dels topics i la comunicació amb els clients.
- Topics: canals o categories lògiques dels missatges; es divideixen en particions per paral·lelitzar i escalar.
- Particions: subdivisons d'un topic que garanteixen l'ordre dels missatges dins de cada partició i gestionen la tolerància a fallades.
- Replicació: de particions en els brokers; una és líder i les altres són sincronitzades per seguretat.
- Producers: aplicacions que publiquen missatges en un topic.
- Consumers: aplicacions que es subscriuen i consumeixen missatges d'un topic.
- ZooKeeper: servei extern que coordina i gestiona la configuració entre els brokers.

2.19 Què és Apache Spark?

És una plataforma opensource per al processament distribuït i àgil de grans volums de dades, ràpida i eficaç, que processa tant dades estructurades com no estructurades, permetent executar anàlisis, machine learning i processament en temps real.

2.20 Característiques d'Apache Spark

- Processament en memòria: emmagatzema dades en RAM per un processament eficient, no com Hadoop MapReduce.
- Multimòdul i versàtil: ja integrats com Spark SQL, Spark Streaming, MLlib i GraphX per processament de gràfics.
- Escalabilitat: amb arquitectura distribuïda, per paralel·litzar tasques en clústers.
- Multi plataforma: executable de manera independent, dins d'Apache Hadoop, Kubernetes, pot accedir a AWS S3, Apache Cassandra, Kafka, etc.
- Multillenguatge de programació.

2.21 Principals elements d'arquitectura d'Apache Spark

- Spark Driver: controlador nucli, coordina les tasques als executors, la comunicació amb el cluster manager, construeix el gràfic acíclic dirigit (DAG).
- Executors: nodes treballadors del clúster amb una JVM pròpia, calculen les dades, mantenen les dades en memòria i comuniquen resultats al driver.
- Cluster Manager: gestiona recursos per als executors.
- Spark Context: punt d'entrada d'una aplicació Spark.

3 Metodologies i eines de gestió

3.1 Què és ELK?

És un conjunt de tres eines opensource: Elasticsearch, Logstash i Kibana, per a la gestió, anàlisi i visualització de grans volums de dades.

3.2 Principals elements d'arquitectura d'ELK

- Elasticsearch: motor de cerca i anàlisi distribuït. Permet buscar i analitzar grans quantitats de dades en temps real.
- Logstash: eina d'ingesta i processament de dades que recopila, filtra, transforma i envia les dades a Elasticsearch. Pot connectar diverses fonts, formats i aplicar lògiques complexes.
- Kibana: plataforma de visualització i exploració de dades, gràfics interactius, mapes, dashboards, per facilitar la interpretació.
- Beats: són els agents que recopilen informació als servidors, com Filebeat o Eventbeat.

3.3 Què és Splunk?

És el mateix que ELK però de pagament segons volum de dades, més intuïtiu, requereix menys configuració i té una corba d'aprenentatge menys pronunciada, amb suport empresarial.

3.4 Què és AGILE?

És una metodologia de desenvolupament de programari basada en iteracions incrementals.

3.5 Característiques de AGILE

- Individus i interaccions per sobre de processos i eines
- Programari funcionant per sobre de documentació exhaustiva
- Col·laboració amb el client per sobre de negociació contractual
- Resposta al canvi per sobre de seguir un pla
- Valor de transparència: visibilitat total del procés de treball
- Valor d'inspecció: revisió freqüent del progrés i els artefactes
- Valor d'adaptació: ajustament basat en els resultats de la inspecció

3.7 Què és SCRUM?

És un marc de treball (framework) d'AGILE basat en equips autogestionats, enfocat a la flexibilitat i a entorns complexos i canviant. Prioritza el valor per al client.

3.8 Característiques de SCRUM

- Control empíric amb tres pilars: transparència, inspecció i adaptació
- Organitzat en SPRINTS: iteracions de treball
- Prioritats: comunicació oberta, compromís de l'equip, adaptació a necessitats canviants

3.9 Principals elements de SCRUM

- **Rol Product Owner**: Defineix i prioritza el PRODUCT BACKLOG, representa els interessos dels STAKEHOLDERS, accepta o rebutja el treball completat, maximitza el valor del producte.
- **Rol SCRUM MASTER**: Facilita el procés, elimina impediments, és un "agile evangelist" i coach de pràctiques AGILE, protegeix l'equip de distraccions externes.
- **Rol DEVELOPER TEAM**: Auto-organitzat i multifuncional, responsable d'entregar el producte; idealment entre 3 i 9 membres, amb totes les habilitats necessàries per completar el treball.
- **PRODUCT BACKLOG**: Llista prioritzada d'històries pendents, refinada contínuament, amb estimació d'esforç i temps pel dev team.
- **SPRINT BACKLOG**: Històries seleccionades per al sprint actual, pla detallat per assolir l'objectiu del sprint, gestionat pel dev team.

- **INCREMENT**: Suma dels elements del product backlog completats, potencialment entregable, que compleix el DoD (Definition of Done).
- **SPRINT**: Període de temps fix durant el qual es crea l'increment; no es pot alterar sense posar en perill l'objectiu del sprint (SPRINT GOAL).
- **SPRINT PLANNING**: Reunió on es planifica el treball del sprint, es defineix l'objectiu del sprint (SPRINT GOAL), i es seleccionen elements del product backlog per passar-los al sprint backlog.
- **DAILY SCRUM**: Reunió diària de 15 minuts per a la sincronització de l'equip: "Què vaig fer ahir?", "Què faré avui?", "Quins impediments tinc?".
- **SPRINT REVIEW**: Reunió de demostració del treball entregat, recollida de retroacció dels stakeholders i actualització del product backlog.
- **SPRINT RETROSPECTIVE**: Reunió de reflexió sobre el procés, identificació de millores i creació d'un pla d'acció per al futur.
- **Mètrica Velocity**: Punts d'història completats per sprint.
- **Burndown chart**: Treball restant vs. temps.
- **Burnup chart**: Treball completat vs. temps.
- **Cycle Time**: Temps des de l'inici fins a la finalització.

3.10 Què és DevOps?

És una cultura que combina el desenvolupament de programari i les operacions d'IT.

3.11 Característiques de DevOps

- **Col·laboració**: Eliminació de silos entre desenvolupament i operacions
- **Automatització**: De processos repetitius
- **Integració contínua**: Integració freqüent dels canvis
- **Monitoratge continu**: D'aplicacions i infraestructura
- **Recuperació ràpida**: De fallades
- Millor qualitat del programari
- Reducció del temps de desenvolupament
- Major col·laboració entre equips
- Resposta més ràpida a les necessitats del negoci

3.13 Què és CI/CD?

És la pràctica de fusionar freqüentment els canvis de codi a la branca principal i desplegar de forma contínua i automàtica el codi fusionat.

3.14 Característiques de CI/CD

CI (Integració Contínua):

1. **Source**: Un membre de l'equip fa push del codi a la branca principal.
2. **Build**: Construcció automàtica del codi.
3. **Test**: Execució de proves unitàries i d'integració.
4. **Quality Gates**: Anàlisi de la qualitat del codi.
5. **Package**: Empaquetat d'artefactes desplegables.
6. **Publish**: Publicació en un repositori d'artefactes.

CD (Entrega/Desplegament Continu):
7. **Deploy to staging**: Desplegament a un entorn de proves.
8. **Integration tests**: Proves d'integració i d'acceptació.
9. **Deploy to prod**: Desplegament automàtic a producció un cop acceptat.
10. **Monitoring**: Monitoratge post-desplegament.

3.15 Principals elements d'arquitectura de CI/CD

- **GIT**: Sistema de control de versions per a la integració del codi.
- **Jenkins**: Servidor d'automatització open source, amb un gran ecosistema de plugins i suport per "pipeline as code" mitjançant Jenkinsfile.
- **GitHub Actions**: Plataforma CI/CD integrada a GitHub, amb workflows en YAML, un ampli mercat d'"actions" i execució en "runners" cloud o self-hosted.

3.16 Què és la gestió de projectes WATERFALL?

És una metodologia tradicional de gestió de projectes amb tasques i fases seqüencials i ben definides.

3.17 Característiques de WATERFALL

- Seqüencial i lineal
- Fases ben definides: iniciació, planificació, execució, monitoratge i tancament
- Documentació exhaustiva
- Menys flexibilitat davant dels canvis
- Adequat per a projectes amb requisits estables

3.18 Què és ITIL?

És un marc de bones pràctiques per alinear els serveis d'TI amb els objectius del negoci, optimitzar processos, millorar la qualitat del suport i augmentar l'eficiència operativa.

3.19 Característiques d'ITIL

- **Enfocament en el client**: Prioritza la seva satisfacció i alinea les TI amb els seus objectius.
- **Cicle de vida complet del servei**: Des de l'estratègia i disseny fins a l'operació i la millora contínua.
- **Processos estandarditzats**: Amb definició clara d'entrades, sortides i resultats mesurables.
- **Millora contínua**: Revisió constant i aprenentatge.
- **Gestió integral de serveis**: Inclou processos de gestió del canvi, incidències, problemes i coneixement.
- **Rols clars**: Dins dels processos.
- **Orientació a resultats**: Es focalitza en què els resultats facilitin als clients assolir els seus objectius.

3.20 Què és la DIRECCIÓ de PROJECTES?

La direcció de projectes permet definir la visió global, l'abast i els objectius del projecte, alineats amb l'estratègia de l'organització, establint prioritats i orientacions per a les fases posteriors. La planificació és el procés de determinar "què cal fer?", "com?", "quan?" i "qui ho farà?", estructurant les activitats en un pla. La valoració econòmica és la previsió i el seguiment dels costos i recursos financers del projecte.

3.21 Principals elements de DIRECCIÓ de PROJECTES

Planificació:

- Definició d'objectius i resultats esperats
- Establiment de fites i calendaris
- Distribució del treball i assignació de recursos
- Elaboració d'un pla de projecte que fa de guia i control
- Mecanismes de control d'abast, calendaris, riscos i canvis

Tècniques d'estimació:

- **Analogous**: Basada en projectes similars
- **Paramètrica**: Utilitzant paràmetres i models matemàtics

- **Three-point**: Optimista, pesimista i més probable
- **Planning Poker**: Estimació col·laborativa en equips àgils

Valoració econòmica:

- Estimació inicial del pressupost
- Planificació i control de costos
- Anàlisi de desviacions
- Avaluació de l'eficiència

Altres elements clau:

- **Pla de projecte**: Document estructurat amb activitats, recursos, calendaris i costos
- **EDT (Estructura de descomposició del treball)**: Desglossament de tasques necessàries, també anomenada Work Breakdown Structure (WBS)
- **Mecanismes i eines de control**: De l'abast, cost, temps i qualitat
- **Assignació de recursos**: Definició de personal, materials i habilitats necessàries
- **Indicadors i informes**: Per avaluar l'estat i el rendiment del projecte

Anàlisi cost-benefici:

- **ROI** (Return on Investment)
- **VAN** (Valor Actual Net / NPV – Net Present Value)
- **TIR** (Taxa Interna de Retorn / IRR – Internal Rate of Return)
- **Període de retorn**: Temps d'amortització de la inversió

---

4. SEGURETAT

4.1 Què és OAUTH2?

És un estàndard obert d'autorització que permet que aplicacions de tercers puguin obtenir accés limitat a serveis web.

4.2 Característiques de OAUTH2

- **Seguretat i delegació**: No comparteix les credencials de l'usuari; utilitza tokens.
- **Arquitectura basada en rols**: Per a cada element del flux.
- **Múltiples fluxos d'autorització**: Tipus de concessió (grant types) adaptats a diferents tipus d'aplicacions i casos d'ús (ex: codi d'autorització, autorització implícita, credencials

del propietari del recurs, credencials del client).
- **Tokens d'accés i actualització**: Tokens d'accés de curta durada i tokens de refresc (refresh tokens) per renovar l'accés.
- **Simplicitat d'implementació**: No utilitza signatures criptogràfiques, només tokens.
- **Control d'abast (scope)**: Per limitar on pot accedir una aplicació.

4.3 Principals elements d'arquitectura OAUTH2

- **Resource owner**: Propietari del recurs que autoritza l'accés.
- **Client**: Aplicació que vol accedir al recurs protegit.
- **Resource server**: Servidor que allotja els recursos protegits.
- **Authorization server**: Servidor que autentica l'usuari i emet els tokens.

4.4 Què és OPENID CONNECT?

És una capa d'autenticació construïda sobre OAUTH2 que proporciona informació sobre la identitat de l'usuari.

4.5 Característiques de OIDC

- Permet a les aplicacions verificar la identitat dels usuaris i obtenir informació dels perfils mitjançant ID tokens basats en JSON Web Tokens (JWT).
- Facilita l'inici de sessió únic (SSO).
- Suporta diferents tipus de clients: web, aplicacions mòbils i d'una sola pàgina (SPA).
- Gestiona el consentiment explícit de l'usuari.
- Model clar de rols.
- Adopta estàndards com JWT i HTTP.
- Integra opcionalment autenticació multifactor (MFA).
- És una alternativa moderna i més senzilla a SAML 2.0.

4.6 Principals elements d'arquitectura OIDC

- **Usuari final**: Persona que vol accedir a l'aplicació.
- **Client o Relying Party (RP)**: Aplicació que sol·licita l'autenticació de l'usuari i confia en el proveïdor d'identitat.
- **Proveïdor d'identitat (Identity Provider, IdP)**: Sistema responsable d'autenticar l'usuari i emetre ID tokens i access tokens (d'OAuth2).
- **Tokens**:

    - **ID Token (JWT)**: Conté informació signada de l'usuari.
    - **Access Token**: Permet accedir a recursos protegits.
    - **Refresh Token**: Permet renovar tokens.
- **Endpoints**: Proporcionats pel IdP per processar sol·licituds d'autenticació, revocació de tokens, etc.

4.7 Què és OWASP?

L'Open Web Application Security Project (OWASP) és una organització sense ànim de lucre dedicada a millorar la seguretat de les aplicacions web. Manté el llistat de vulnerabilitats crítiques conegut com a **OWASP Top 10**.

4.8 Característiques d'OWASP (Top 10 2021)

- **A01: Broken Access Control**: Control d'accés inadequat o inexistent, escalada de privilegis, accés no autoritzat.
  Solució: Implementar control d'accés, denegar per defecte, registrar intents fallits.
- **A02: Cryptographic Failures**: Dades sensibles mal xifrades, algorismes febles, gestió inadequada de claus.
  Solució: Xifratge robust, gestió segura de claus.
- **A03: Injection**: Injeccions SQL, NoSQL, de comandaments, etc., degut a dades no validades.
  Solució: Consultes parametritzades, validació d'entrades.
- **A04: Insecure Design**: Manca de controls, arquitectura insegura, modelatge inadequat d'amenaces.
  Solució: Seguretat des del disseny, modelatge d'amenaces.
- **A05: Security Misconfiguration**: Configuracions per defecte, funcionalitats innecessàries activades, gestió inadequada d'errors.
  Solució: Hardening de sistemes, automatització de configuracions segures.
- **A06: Vulnerable and Outdated Components**: Vulnerabilitats conegudes, components obsolets, dependències insegures.
  Solució: Inventari de components, actualitzacions regulars, monitoratge de vulnerabilitats.
- **A07: Identification and Authentication Failures**: Autenticació feble, gestió inadequada de sessions, contrasenyes febles.
  Solució: Autenticació multifactor, polítiques de contrasenya robustes, gestió segura de sessions.
- **A08: Software and Data Integrity Failures**: Pipelines CI/CD insegurs, deserialització insegura, plugins/libreries no verificades.
  Solució: Logging adequat, CI/CD segur, validació d'integritat.

- **A09: Security Logging and Monitoring Failures**: Registre (logging) inadequat, monitoratge insuficient, resposta inadequada a incidents.
  Solució: Logging comprehensiu, monitoratge en temps real, alertes automatitzades.
- **A10: Server-Side Request Forgery (SSRF)**: L'aplicació fa peticions a URLs proporcionades per l'usuari, permetent accés a recursos interns o eludir tallafocs.
  Solució: Validació rigorosa de URLs, llistes blanques de destinacions, segregació de xarxa.

4.9 Principals elements d'arquitectura OWASP

- **OWASP ZAP**: Eina d'auditoria i escaneig per detectar riscos i vulnerabilitats de seguretat.
- **OWASP Dependency-Check**: Escaneja el codi per detectar biblioteques i dependències vulnerables.
- **OWASP Web Security Testing Guide (WSTG)**: Guia exhaustiva per realitzar proves de seguretat.
- **OWASP Juice Shop**: Aplicació web intencionadament vulnerable per a l'aprenentatge.
- **OWASP Amass**: Eina per recollir informació i cartografiar l'atac a la infraestructura d'una organització.
- **OWASP ModSecurity Core Rule Set (CRS)**: Conjunt de regles per a WAF (Web Application Firewall).
- **OWASP Cheat Sheet Series**: Col·lecció de bones pràctiques i recomanacions de seguretat.