ANNO ACCADEMICO 2025/2026

Tecnologie e Architetture Avanzate di Sviluppo Software

Teoria

Altair's Notes

DIPARTIMENTO DI INFORMATICA

Premessa

Licenza

Formato utilizzato

Box di "Concetto sbagliato":

Concetto sbagliato 0.1: Testo del concetto sbagliato

Testo contente il concetto giusto.

Box di "Corollario":

Corollario 0.0.1 Nome del corollario

Testo del corollario. Per corollario si intende una definizione minore, legata a un'altra definizione.

Box di "Definizione":

Definizione 0.0.1: Nome delle definizione

Testo della definizione.

Box di "Domanda":

Domanda 0.1

Testo della domanda. Le domande sono spesso utilizzate per far riflettere sulle definizioni o sui concetti.

Box di "Esempio":

Esempio 0.0.1 (Nome dell'esempio)

Testo dell'esempio. Gli esempi sono tratti dalle slides del corso.

Box di "Note":

Note:-

Testo della nota. Le note sono spesso utilizzate per chiarire concetti o per dare informazioni aggiuntive.

Box di "Osservazioni":

Osservazioni 0.0.1

Testo delle osservazioni. Le osservazioni sono spesso utilizzate per chiarire concetti o per dare informazioni aggiuntive. A differenza delle note le osservazioni sono più specifiche.

1 Introduzione

1.1 Intro al Corso

Parole chiave:

- Web Apps.
- Mission Critical.
- DevOps.
- Cloud Native.

Definizione 1.1.1: Mission Critical Applications

Un'applicazione o sistema le cui operazioni sono fondamentali per una compagnia o un'istituzione.

Osservazioni 1.1.1

- Enfasi sui requisiti non funzionali: i requisiti funzionali sono la baseline, ma ci si aspetta di più per rimanere competitivi.
- Da non confondere con life critical: non muore nessuno.

Definizione 1.1.2: Enterprise Application Integration (EAI)

Tutto l'insieme di pratiche architetturali, tecnologie, patterns, frameworks e strumenti che consentono la comunicazione e la condivisione tra diverse applicazioni nella stessa organizzazione.

Si ha enfasi sull'infrastruttura:

- *Data Integration:* combinare dati da più moduli diversi (coinvolge database).
- *Process Integration:* le interazioni tra più moduli.
- *Functional Integration:* si vuole fornire una nuova funzionalità sfruttando funzionalità già esistenti.

1.1.1 Esempio e Requisiti Non Funzionali

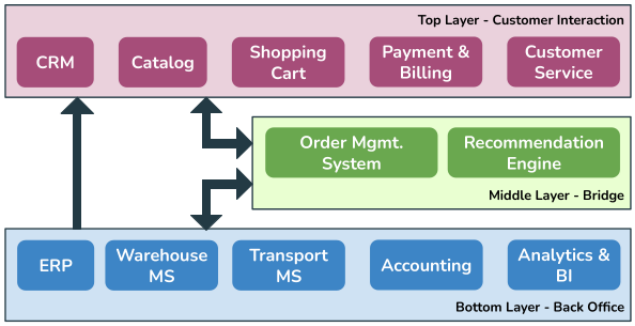

Figure 1.1: Esempio di e-commerce.

Commento dell'esempio:

- Ci sono tre livelli:
  - Top Layer: moduli che si rivolgono al cliente.
  - Middle Layer: gestione della comunicazione tra cliente e azienda.
  - Bottom Layer: moduli interni aziendali.

Requisiti non funzionali:

- High availability/zero downtime: l'applicativo deve essere sempre o quasi sempre disponibile.
- Affidabilità: in caso di interruzione di workflow si deve far sì che non ci siano stati danni (e.g. un'interruzione durante una transazione).
- Consistenza dei dati.
- Integrità dei dati.
- Low latency: per avere una buona performance, tutto deve essere fluido.
- Scalabilità.
- Sicurezza.
- Resilienza: capacità di reagire agli errori.
- Mantenibilità: quanto un pezzo di software sia mantenibile o riutilizzabile.
- Osservabilità: per comprendere eventuali problemi in un sistema distribuito.
- Auditability: le verifiche di qualità fatte su software[1].

1.1.2 Panoramica Storica

Definizione 1.1.3: Waterfall

Le metodologie a cascata[a] sono metodologie in cui ci sono fasi ben distinte e separate tra loro.

[a] Viste a "Sviluppo delle Applicazioni Software".

[1] Meglio visto in "Etica, Società e Privacy".

Note:-

È un modello prevedibile, ma lento a gestire i cambiamenti.

Osservazioni 1.1.2

- Software on the shelf: una volta acquistato è proprio.
- Software custom: prodotto su richiesta, ha bisogno di tutto un servizio di manutenzione.

Definizione 1.1.4: Lean

Metodologie nate negli anni '50 alla Toyota, verranno applicate al software dagli anni '90. Si basa su tre principi:

- Muda[a] (waste): si deve stare sui requisiti, non mettere troppe funzioni non necessarie.
- Mura (unevenness): è necessaria consinstenza per aumentare la prevedibilità.
- Muri (overburden): non sovraccaricare le persone o le macchine. Non progettare software utilizzando strumenti greedy di risorse.

[a] JOJO'S Reference

Note:-

Lo strumento fondamentale è il *kanban*: la lavagna, per organizzare il lavoro.

Definizione 1.1.5: Siloed

Organizzazione aziendale a silos: si comunica poco e male. Ci sono 4 gruppi:

- BA Team: relazioni con gli stakeholders, requisiti, specifiche, documentazione.
- Dev Team: programma e fa un minimo di unit testing.
- Test Team: testa e decide se il sistema è pronto.
- Ops Team: si occupa del deployement.

Note:-

I vari team si parlano in maniera molto limitata.

Definizione 1.1.6: Transaction Processing Monitor

I TP monitor erano il primo esempio di soluzione middleware. Usata nei sistemi di mainframe erano: centralizzati, monolitici, mission critical, con accesso da vari terminali.

Corollario 1.1.1 Middleware

Software nel mezzo tra applicazioni e infrastrutture. Permette alle applicazioni di utilizzare le infrastrutture per farle comunicare tra di loro.

Obiettivi:

- Performance: si occupa di transazioni rispettando le proprietà ACID.
- Scalabilità: se un programma crasha ne avvia un'altra istanza.
- Affidabilità.
- Consistenza dei dati.

Limiti:

- Proprietario.
- Tight coupling.
- Costosi.
- Complessi.

Domanda 1.1

Cosa rimane dei TP monitors?

- *Gestione delle transazioni e coordinazione:*
  - Soluzioni basate su 2PC (2 Phase Commit).
  - Le proprietà ACID, attualmente supportate internamente da molti database.
  - Proprietà BASE:
    - Basically: risposte basiche.
    - Available: si accetta che si possa non avere il dato più aggiornato.
    - State: la consistenza potrebbe non essere rispettata.
    - Eventually: prima o poi si riceverà il dato corretto.
- *Pool di connessioni*.
- *Distribuzione del carico:*
  - Le richieste vengono distribuite su varie istanze.
  - In caso di fallimento l'applicazione riparte.

Definizione 1.1.7: Remote Procedure Call

Si chiama una funzione da una macchina remota come se fosse locale. È indipendente dal linguaggio e a una struttura silos. RIchiede aggiunte sia nello sviluppo che a runtime.

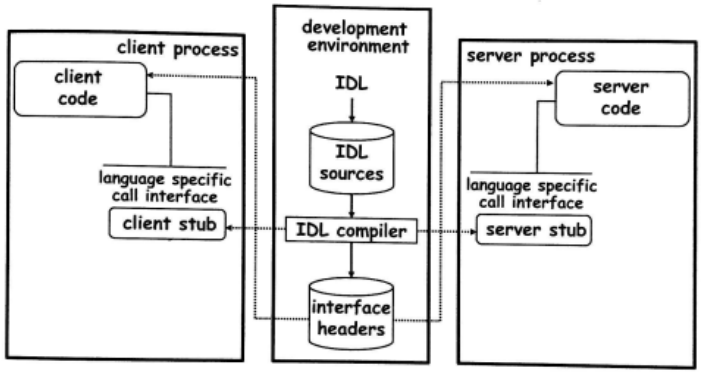

Figure 1.2: Remote Procedure Call - Development.

- Serializzazione: trasformare i dati in qualcosa che può essere comunicato.
- Marshalling: usa la serializzazione e inserisce meta-dati per permettere la ricostruzione della struttura dati.

Definizione 1.1.8: Common Object Request Broker Architecture (CORBA)

Evoluzione di rpc pensata per gli oggetti. Si possono creare oggetti in un server che possono rispondere a chiamate remote.

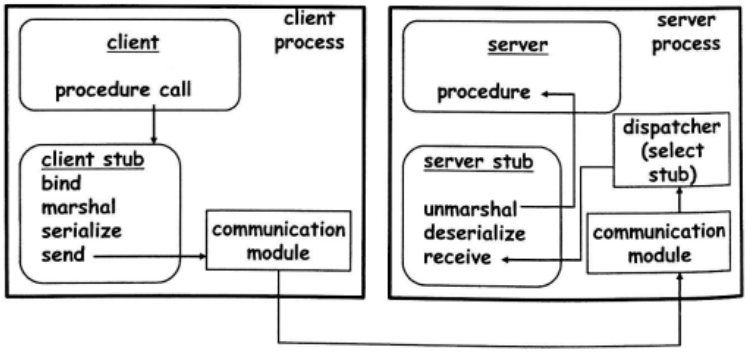

Figure 1.3: Remote Procedure Call - Runtime.

Note:-

Più successo lo ha avuto RMI (Remote Method Invocation) che è CORBA, ma solo con Java.

Limiti:

- Nascondere le cose al programmatore: si ha un falso senso di disaccoppiamento e i programmatori tendono a non vedere la rete.
- La programmazione sembra semplice perché i problemi vengono sottovalutati.

Definizione 1.1.9: Message Oriented Middleware

Invece di chiamarsi a vicenda le applicazioni si inviano messaggi a vicenda:

- Sincronizzazione tra operazioni in applicazioni diverse.
- Notifiche di eventi.
- Non c'è necessità di conoscere il ricevente.

Due modelli di comunicazione:

- Point-to-Point: il mittente manda un messaggio nella coda del middleware, il ricevente lo consuma.
- Publish and Subscribe: c'è una bacheca su cui chiunque può pubblicare un evento.

Definizione 1.1.10: Enterprise Service Bus (ESB)

Un middleware coscente della logica di business. Si occupa di tradurre protocolli e dati.

Note:-

Caduto totalmente in disuso.

Definizione 1.1.11: AGILE

Metodologie fondate su itertività e incrementalità.

Corollario 1.1.2 XP - Xtreme Programming

Si concentra sul codice, lo sviluppo di software si fa in team. Si dà importanza ai feedback sia dai clienti che dagli sviluppatori (small release, test-driven development, on-site customer).

Principi di XP:

- Comunicazione.
- Semplicità.
- Feedback.
- Coraggio.
- Rispetto.

1.2 Ripasso su Spring Boot

Note:-

DISCLAIMER: è il mio primo approccio alla programmazione web (dato che sono specializzata in robe teoriche e/o a basso livello) per cui potrei fare qualche imprecisione, sorry.

1.2.1 Maven

Dato che gli IDE moderni consumano un sacco di batteria e risorse includo anche una mini guida per setuppare un progetto java con Maven (in questo modo potete usare vim, gedit o nano se vi va). Se usate Intell*J, Vs C*de o altro potete saltare[2].

Definizione 1.2.1: Maven

Maven è un tool per creare automaticamente delle build di progetti java. Permette di compilare codice, fare testing, packaging, etc.

Note:-

Maven utilizza il *Project Object Model (POM)* per descrivere la configurazione di un progetto e gestire le dipendenze.

Domanda 1.2

Come si crea un progetto con Maven?

Listing 1.1: Creazione di un progetto Maven

```
mvn archetype:generate \
    -DgroupId=com.example \
    -DartifactId=myapp \
    -DarchetypeArtifactId=maven-archetype-quickstart \
    -DinteractiveMode=false
```

Nello specifico:

- `DgroupID` indica il nome di una compagnia o di un'organizzazione.
- `DartifactId` indica il nome del progetto.
- `DarchetypeArtifactId` indica il template (in questo caso un semplice HelloWorld java.).

[2] questi IDE possono utilizzare anche Maven, ma lo gestiscono loro.

Listing 1.2: Esempio di pom.xml per Spring Boot

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
          http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>3.3.4</version>
        <relativePath/>
    </parent>

    <groupId>com.example</groupId>
    <artifactId>myapp</artifactId>
    <version>1.0.0-SNAPSHOT</version>
    <packaging>jar</packaging>
    <name>myapp</name>

    <properties>
        <java.version>17</java.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>
</project>
```

Spiegazione:

- Il `parent` imposta la versione di Spring Boot e le configurazioni di default.
- Le `dependencies` includono il modulo web e quello per i test.
- Il plugin `spring-boot-maven-plugin` permette di eseguire l'app con `mvn spring-boot:run`.

1.2.2 Gradle

Per alcune persone può essere più facile utilizzare Gradle (inclusa me), quindi aggiungo qualcosa anche per questo.

Definizione 1.2.2: Gradle

Come Maven, Gradle è un tool per creare automaticamente progetti java, C/C++, kotlin, etc. A livello di base ha le stesse funzionalità di Maven, le differenze principali sono il linguaggio utilizzato (Maven è basato su xml, Gradle su Groovy), velocità (Gradle è più veloce per build incrementali), etc.

Domanda 1.3

Come si crea un progetto Spring Boot con Gradle?

Listing 1.3: Creazione di un progetto Spring Boot con Gradle

```
curl https://start.spring.io/starter.tgz \
        -d type=gradle-project \
        -d dependencies=web \
        -d groupId=com.example \
        -d artifactId=test \
        -d name=test \
        -d packageName=com.example.test \
        -o test-gradle.tgz
  tar -xvf test-gradle.tgz
  cd test
```

Alcune osservazioni importanti:

- Di default il progetto creato usa java 17, per cambiarlo basta andare nel file `build.gradle`.
- Inizialmente darà errore perché non si sono definiti endpoint.

Listing 1.4: Avvio del progetto Spring Boot con Gradle

```
# Su Linux/macOS
./gradlew bootRun

# Su Windows
gradlew.bat bootRun
```

Note:-

Al primo avvio Gradle scaricherà tutte le dipendenze necessarie. Una volta completato, l'app sarà disponibile su `http://localhost:8080/`. Se non hai ancora definito controller o endpoint, vedrai la *Whitelabel Error Page*.

1.3 Spring Boot