



## BugBusters

Email: [bugbusters.unipd@gmail.com](mailto:bugbusters.unipd@gmail.com)

Gruppo: 4

Università degli Studi di Padova

Laurea in Informatica

Corso: Ingegneria del Software

Anno Accademico: 2025/2026

# Glossario

**Versione 0.0.2**

<b>Stato</b>	In redazione
<b>Redattori</b>	Alberto Autiero
<b>Verificatori</b>	[Nome Cognome]
<b>Uso</b>	Interno ed esterno
<b>Destinatari</b>	Prof. Tullio Vardanega, Prof. Riccardo Cardin, BugBusters, Eggon

## Indice

Introduzione	8
A	9
AI (Artificial Intelligence) . . . . .	9
AI Co-Pilot . . . . .	9
AI generativa . . . . .	9
Accoppiamento . . . . .	9
Accoppiamento (Coupling) . . . . .	9
Affidabilità (Reliability) . . . . .	9
Agile . . . . .	9
Aggregazione . . . . .	9
Amministratore . . . . .	10
Analisi dei Requisiti . . . . .	10
Analista . . . . .	10
API (Application Programming Interface) . . . . .	10
Approvazione . . . . .	10
Architettura a livelli . . . . .	10
Architettura dei microservizi . . . . .	10
Architettura multilivello . . . . .	10
Architettura Pipe-and-Filter . . . . .	10
Architettura Three-Tier . . . . .	11
Associazione . . . . .	11
Attore . . . . .	11
Attore principale . . . . .	11
Attore secondario . . . . .	11
Availability (Disponibilità) . . . . .	11
AWS (Amazon Web Services) . . . . .	11
B	12
Back-end . . . . .	12
Baseline . . . . .	12
Best Practice . . . . .	12
C	13
Capitolato . . . . .	13
Casi d'uso . . . . .	13
Cedolini . . . . .	13
Cedolini Massivi . . . . .	13
Cerimonia . . . . .	13
Committente . . . . .	13
Composizione . . . . .	13

Comprensibilità . . . . .	13
Constructor Injection . . . . .	13
Cruscotto/Dashboard . . . . .	14
<b>D</b>	15
Decomposizione funzionale . . . . .	15
Decisione esterna . . . . .	15
Decisione interna . . . . .	15
Dependency Injection . . . . .	15
Design pattern architetturali . . . . .	15
Diagramma dei componenti . . . . .	15
Diagramma dei casi d'uso . . . . .	15
Diagramma delle classi . . . . .	15
Diagramma degli stati . . . . .	15
Diagramma di attività . . . . .	16
Diagramma di deployment . . . . .	16
Diagramma di Gantt . . . . .	16
Diagramma di PERT . . . . .	16
Diagramma di sequenza . . . . .	16
Dipendenza . . . . .	16
Discord . . . . .	16
Dispatch . . . . .	16
Dominio d'uso . . . . .	16
<b>E</b>	17
Economicità . . . . .	17
Efficacia . . . . .	17
Efficienza . . . . .	17
Entity Resolution . . . . .	17
Ereditarietà . . . . .	17
Ereditarietà di classe . . . . .	17
Ereditarietà dell'interfaccia . . . . .	17
<b>F</b>	18
Flessibilità . . . . .	18
Front-end . . . . .	18
Funzionalità . . . . .	18
<b>G</b>	19
GDPR (General Data Protection Regulation) . . . . .	19
Gestione dei rischi . . . . .	19
GitHub . . . . .	19
Glossario . . . . .	19

H	20
Human in the loop . . . . .	20
I	21
Implementazione . . . . .	21
Incapsulamento . . . . .	21
Information Hiding . . . . .	21
Issue . . . . .	21
K	22
KPI (Key Performance Indicator) . . . . .	22
L	23
Latex . . . . .	23
Livelli aperti . . . . .	23
Livelli chiusi . . . . .	23
LLM (Large Language Model) . . . . .	23
Load balancer . . . . .	23
M	24
Microservizio . . . . .	24
Modularità . . . . .	24
O	25
OCR (Optical Character Recognition) . . . . .	25
P	26
Polimorfismo . . . . .	26
Post-condizione . . . . .	26
Pre-condizione . . . . .	26
Preventivo . . . . .	26
Procedimento agile . . . . .	26
Procedimento bottom-up . . . . .	26
Procedimento top-down . . . . .	26
Progettazione di dettaglio . . . . .	26
Progettazione logica . . . . .	26
Progettista . . . . .	27
Programmatore . . . . .	27
Progetto . . . . .	27
Prompt . . . . .	27
Proponente . . . . .	27
Processi organizzativi . . . . .	27
Processi primari . . . . .	27
Processi di supporto . . . . .	27
PWA (Progressive Web App) . . . . .	27

Q		28
	Qualità architetturale . . . . .	28
R		29
	Rating System . . . . .	29
	RBAC (Role-Based Access Control) . . . . .	29
	Redattore . . . . .	29
	Repository . . . . .	29
	Requisito . . . . .	29
	Requisiti desiderabili . . . . .	29
	Requisiti funzionali . . . . .	29
	Requisiti non funzionali . . . . .	29
	Requisiti obbligatori . . . . .	29
	Requisiti opzionali . . . . .	30
	Requirements Baseline . . . . .	30
	Requisito software . . . . .	30
	Requisito utente . . . . .	30
	Responsabile . . . . .	30
	Riusabilità . . . . .	30
	Robustezza . . . . .	30
S		31
	Sandbox di Sviluppo . . . . .	31
	Scalabilità . . . . .	31
	Scenario . . . . .	31
	Scenario alternativo . . . . .	31
	Scenario principale . . . . .	31
	Scope . . . . .	31
	SCRUM . . . . .	31
	SEMAT . . . . .	31
	Sottotipizzazione . . . . .	31
	Specification Tecnica . . . . .	32
	Sprint . . . . .	32
	Stakeholder . . . . .	32
	Stima dei costi . . . . .	32
T		33
	Technology Baseline . . . . .	33
	Test . . . . .	33
	Transazioni distribuite . . . . .	33
U		34
	Unità architetturali . . . . .	34

V	35
Validatore . . . . .	35
Validazione . . . . .	35
Verifica . . . . .	35
Verificatore . . . . .	35
Versionamento/versioning . . . . .	35
W	36
Way of Working . . . . .	36
Workflow . . . . .	36

## Versioni del documento

Versione	Data	Descrizione	Redatto	Verificatori	Approvato
0.0.2	17/11/2025	Aggiunti nuovi termini.	Alberto Autiero	-	-
0.0.1	06/11/2025	Prima stesura del documento con i termini del cappitolato aggiudicato (5) e diapositive dei docenti.	Alberto Autiero	-	-

## Introduzione

Questo documento è stato redatto per garantire chiarezza e uniformità nell'interpretazione della terminologia utilizzata nell'intera documentazione del progetto. Al fine di eliminare possibili fraintendimenti, vengono presentate qui le definizioni dei termini tecnici, degli acronimi e delle espressioni che potrebbero risultare ambigue. L'obiettivo del glossario consiste nell'assicurare che tutti i membri del gruppo di lavoro e i destinatari della documentazione condividano un'interpretazione univoca e coerente della terminologia impiegata.

La nomenclatura utilizzata per evidenziare che una parola, con annessa la sua definizione, è presente all'interno del Glossario viene indicata come segue: **parolaG**.

## A

### **AI (Artificial Intelligence)**

Campo dell'informatica che si occupa di sviluppare sistemi in grado di svolgere compiti che normalmente richiedono l'intelligenza umana, come il riconoscimento vocale, la visione artificiale, l'elaborazione del linguaggio naturale e il processo decisionale.

### **AI Co-Pilot**

Modulo di supporto automatizzato per gli studi dei Consulenti del Lavoro (CdL), che assiste nei processi di gestione, riconoscimento e distribuzione documentale attraverso tecniche di intelligenza artificiale.

### **AI generativa**

Tipo di intelligenza artificiale in grado di creare autonomamente nuovi contenuti (testo, immagini, audio, codice) basandosi su pattern appresi dai dati di addestramento.

### **Accoppiamento**

Grado di interdipendenza tra componenti software. Un accoppiamento stretto indica forte dipendenza, mentre un accoppiamento debole minimizza le dipendenze, favorendo manutenibilità e riutilizzo del codice.

### **Accoppiamento (Coupling)**

Grado di interdipendenza tra componenti software. Un accoppiamento stretto indica forte dipendenza, mentre un accoppiamento debole minimizza le dipendenze, favorendo manutenibilità e riutilizzo del codice.

### **Affidabilità (Reliability)**

Capacità di un sistema software di mantenere il proprio livello di prestazioni in condizioni specificate per un determinato periodo di tempo.

### **Agile**

Metodologia di sviluppo del software iterativa e incrementale che si basa su principi come la collaborazione continua con il cliente, la consegna frequente di software funzionante e la capacità di rispondere ai cambiamenti dei requisiti.

### **Aggregazione**

Relazione strutturale in OOP che rappresenta un legame "parte-tutto" dove gli componenti possono esistere indipendentemente dall'oggetto che li contiene. È una forma di associazione con condivisione dei riferimenti.

## **Amministratore**

Figura responsabile della gestione dell'infrastruttura, degli strumenti di sviluppo e dei processi del progetto. Si occupa di configurare e mantenere gli ambienti di lavoro e garantire che il team abbia a disposizione gli strumenti necessari.

## **Analisi dei Requisiti**

Processo sistematico volto a identificare, documentare e validare i bisogni, i vincoli e le esigenze del progetto, trasformandoli in requisiti formali che guideranno lo sviluppo del software.

## **Analista**

Figura professionale specializzata nell'analisi dei requisiti e nella specifica delle funzionalità del sistema. Collabora con gli stakeholder per comprendere le esigenze e tradurle in specifiche tecniche.

## **API (Application Programming Interface)**

Insieme di definizioni, protocolli e strumenti per la costruzione e l'integrazione di software applicativo. Consente a diversi sistemi di comunicare tra loro.

## **Approvazione**

Processo formale attraverso il quale un documento o una funzionalità viene accettata e considerata completa dopo aver superato le verifiche e validazioni necessarie.

## **Architettura a livelli**

Pattern architettonico che organizza il sistema in livelli gerarchici, dove ogni livello fornisce servizi al livello superiore e utilizza servizi del livello inferiore.

## **Architettura dei microservizi**

Approccio architettonico che struttura un'applicazione come una collezione di servizi piccoli, autonomi e indipendenti, ciascuno responsabile di una specifica funzionalità di business.

## **Architettura multilivello**

Pattern architettonico che separa le responsabilità dell'applicazione in livelli logici distinti, tipicamente presentazione, logica di business e persistenza dati.

## **Architettura Pipe-and-Filter**

Pattern architettonico in cui i componenti (filtri) elaborano i dati in sequenza, passandoli attraverso connessioni (pipe).

## **Architettura Three-Tier**

Architettura software che divide l'applicazione in tre livelli: presentazione, logica di business e dati.

## **Associazione**

Relazione strutturale in OOP che rappresenta una connessione duratura tra oggetti di classi diverse, dove gli oggetti interagiscono per un periodo di tempo prolungato condividendo riferimenti.

## **Attore**

Ruolo svolto da un utente o sistema esterno che interagisce con il sistema software per raggiungere obiettivi specifici. Gli attori possono essere primari (beneficiari diretti) o secondari (fornitori di servizi).

### **Attore principale**

Utente o sistema esterno che interagisce direttamente con il sistema software per raggiungere un obiettivo specifico nel caso d'uso. È il protagonista dell'interazione e trae beneficio diretto dall'esecuzione del caso d'uso.

### **Attore secondario**

Utente o sistema esterno che fornisce servizi o supporto all'attore principale durante l'esecuzione di un caso d'uso. Partecipa all'interazione ma non è il beneficiario principale del risultato, spesso fornendo funzionalità accessorie o di supporto.

## **Availability (Disponibilità)**

Misura della percentuale di tempo in cui un sistema è operativo e accessibile agli utenti.

## **AWS (Amazon Web Services)**

Piattaforma di cloud computing che offre servizi di calcolo, storage, database e altre funzionalità per supportare lo sviluppo e il deployment di applicazioni.

## B

### **Back-end**

Parte di un'applicazione software che gestisce la logica di business, l'elaborazione dei dati e la comunicazione con il database. Opera sul server ed è inaccessibile direttamente all'utente finale.

### **Baseline**

Versione approvata e formalmente controllata di un documento o di un componente software che serve come riferimento per sviluppi successivi. Le modifiche successive richiedono procedures formali di controllo.

### **Best Practice**

Insieme di tecniche, metodi e procedures che sono state riconosciute come le più efficaci ed efficienti per raggiungere un obiettivo specifico in un determinato contesto.

## C

### **Capitolato**

Documento contrattuale che specifica i requisiti, le caratteristiche tecniche e le condizioni di un progetto software proposto da un'azienda proponente per il corso di Ingegneria del Software.

### **Casi d'uso**

Tecnica di specifica dei requisiti che descrive le interazioni tra gli attori (utenti o sistemi esterni) e il sistema software per raggiungere un obiettivo specifico.

### **Cedolini**

Documenti retributivi che attestano la retribuzione corrisposta al dipendente per un determinato periodo di paga.

### **Cedolini Massivi**

File contenenti più documenti retributivi aggregati, da suddividere e assegnare ai singoli destinatari tramite riconoscimento automatico.

### **Cerimonia**

Evento formale o informale nel framework Scrum che segue un agenda prestabilita e ha uno scopo specifico, come la Pianificazione dello Sprint, il Daily Stand-up, la Revisione dello Sprint o la Retrospettiva.

### **Committente**

Soggetto che commissiona il progetto, definendone gli obiettivi, i vincoli e i requisiti, e che ricepisce il prodotto finale. Nel contesto del corso di Ingegneria del Software, può essere un'azienda proponente o un docente.

### **Composizione**

Relazione strutturale in OOP che rappresenta un legame "parte-tutto" forte dove gli componenti non possono esistere indipendentemente dall'oggetto che li contiene. Implica ownership esclusiva e distruzione concomitante.

### **Comprensibilità**

Capacità di un sistema software di essere facilmente compreso dai suoi utilizzatori, riducendo lo sforzo cognitivo necessario per il suo utilizzo.

### **Constructor Injection**

Tecnica di Dependency Injection in cui le dipendenze vengono fornite a un componente attraverso il suo costruttore.

**Cruscotto/Dashboard**

Interfaccia utente che presenta in forma grafica e sintetica le metriche, gli indicatori di performance e lo stato corrente del progetto o dell'applicazione.

## D

### **Decomposizione funzionale**

Processo di scomposizione di un sistema complesso in funzioni o componenti più piccoli e gestibili.

### **Decisione esterna**

Scelta presa da entità esterne al team di progetto (come il committente o il proponente) che vincola le attività del progetto e che il team deve rispettare.

### **Decisione interna**

Scelta presa dal team di progetto riguardante aspetti tecnici, organizzativi o metodologici, documentata per garantire tracciabilità e coerenza nelle attività successive.

### **Dependency Injection**

Pattern architettonale in cui le dipendenze di un componente vengono fornite dall'esterno, anziché essere create internamente al componente stesso.

### **Design pattern architetturali**

Soluzioni progettuali ricorrenti e collaudate per problemi architetturali comuni nei sistemi software.

### **Diagramma dei componenti**

Diagramma UML che mostra l'organizzazione e le dipendenze tra i componenti software di un sistema.

### **Diagramma dei casi d'uso**

Diagramma UML che descrive le interazioni tra gli attori e il sistema, mostrando i diversi scenari possibili per raggiungere un obiettivo specifico.

### **Diagramma delle classi**

Diagramma UML che mostrano le classi del sistema, i loro attributi, metodi e le relazioni tra di esse (associazioni, ereditarietà, dipendenze, ecc.).

### **Diagramma degli stati**

Diagramma UML che descrive il comportamento di un oggetto in risposta a eventi, mostrando i diversi stati in cui può trovarsi e le transizioni tra essi.

## **Diagramma di attività**

Diagramma UML che modellano il flusso di controllo o il flusso di dati tra attività, utilizzati per descrivere la logica di procedura, di business o di caso d'uso.

## **Diagramma di deployment**

Diagramma UML che mostra la configurazione fisica dei nodi di elaborazione e dei componenti software eseguiti su di essi.

## **Diagramma di Gantt**

Strumento di pianificazione progettuale che visualizza le attività su un asse temporale, mostrando durate, sequenzialità, parallelismo e progressi rispetto alle pianificazioni.

## **Diagramma di PERT**

(Program Evaluation and Review Technique) Strumento di analisi delle dipendenze temporali entre attività di progetto, utilizzato per identificare cammini critici e margini temporali (slack time).

## **Diagramma di sequenza**

Diagramma UML che mostra le interazioni tra oggetti in sequenza temporale, evidenziando l'ordine dei messaggi scambiati.

## **Dipendenza**

Relazione tra componenti software per cui un componente (dipendente) richiede un altro componente (dipenduto) per funzionare correttamente. Le modifiche al componente dipenduto possono influenzare il componente dipendente.

## **Discord**

Piattaforma di comunicazione tramite chat vocale, testuale e video, utilizzata dal gruppo per la comunicazione interna e la collaborazione quotidiana.

## **Dispatch**

Processo di distribuzione automatizzata dei documenti verso i destinatari finali attraverso diversi canali (es. app, portale, email, PEC).

## **Dominio d'uso**

Contesto specifico o ambiente operativo in cui il sistema software sarà impiegato, comprendente le caratteristiche degli utenti finali, le condizioni operative, i vincoli tecnologici e le regole di business che definiscono l'ambito di applicazione del prodotto.

## E

### **Economicità**

Principio gestionale che combina efficienza ed efficacia, misurando la capacità di raggiungere obiettivi prefissati (efficacia) impiegando le risorse minime indispensabili (efficienza).

### **Efficacia**

Capacità di raggiungere gli obiettivi prefissati e produrre i risultati attesi, indipendentemente dalle risorse impiegate.

### **Efficienza**

Rapporto entre i risultati ottenuti e le risorse impiegate per conseguirli. Un processo è efficiente quando raggiunge i suoi obiettivi utilizzando il minimo di risorse necessarie.

### **Entity Resolution**

Processo di identificazione e associazione di entità (es. persone o aziende) a partire da dati parziali o duplicati, tramite algoritmi di matching e disambiguazione.

### **Ereditarietà**

Meccanismo della programmazione orientata agli oggetti che permette a una classe (sottoclasse) di acquisire attributi e metodi di un'altra classe (superclasse), favorendo il riutilizzo del codice e le relazioni di generalizzazione.

### **Ereditarietà di classe**

Meccanismo di ereditarietà in cui una classe eredita da un'altra classe (sia classe concreta che astratta).

### **Ereditarietà dell'interfaccia**

Meccanismo di ereditarietà in cui una interfaccia eredita da un'altra interfaccia, oppure una classe implementa un'interfaccia.

**F****Flessibilità**

Capacità di un sistema software di adattarsi a cambiamenti nei requisiti o nell'ambiente operativo con modifiche minime.

**Front-end**

Parte di un'applicazione software con cui l'utente interagisce direttamente, responsabile della presentazione dei dati e dell'acquisizione dell'input dell'utente.

**Funzionalità**

Caratteristica o capacità specifica che un sistema software deve possedere per soddisfare i bisogni degli utenti e gli obiettivi del progetto.

## G

### **GDPR (General Data Protection Regulation)**

Regolamento generale sulla protezione dei dati dell'Unione Europea che stabilisce norme per la protezione e la libera circolazione dei dati personali.

### **Gestione dei rischi**

Processo sistematico di identificazione, analisi, pianificazione e controllo dei rischi di progetto, volto a minimizzare la probabilità di occorrenza e l'impatto degli eventi negativi.

### **GitHub**

Piattaforma di hosting per repository Git che offre strumenti per il version control, la collaborazione e la gestione del ciclo di vita del software.

### **Glossario**

Documento che raccoglie e definisce i termini specifici, gli acronimi e le parole ambigue utilizzati nella documentazione di progetto, con lo scopo di garantire una comprensione univoca della terminologia da parte di tutti i membri del team e dei destinatari.

## H

### **Human in the loop**

Approccio in cui l'intelligenza artificiale e gli esseri umani collaborano, con l'uomo che supervisiona, corregge o fornisce feedback al sistema AI, particolarmente utile quando la confidenza del sistema è bassa.

# I

## Implementazione

Processo di traduzione di un design in codice eseguibile, realizzando le specifiche funzionali e non funzionali.

## Incapsulamento

Principio della programmazione orientata agli oggetti che consiste nel racchiudere in un'unica entità (classe) dati e metodi che operano su di essi, nascondendo i dettagli implementativi all'esterno.

## Information Hiding

Principio di progettazione software che consiste nel nascondere i dettagli implementativi di un modulo, esponendo solo le interfacce necessarie, per ridurre l'accoppiamento e aumentare la manutenibilità.

## Issue

Segnalazione di un problema, un bug o una richiesta di miglioramento nel sistema di tracking del progetto. Ogni issue viene tracciata, assegnata e gestita fino alla risoluzione.

## K

### **KPI (Key Performance Indicator)**

Indicatore chiave di performance che misura l'efficacia di un processo, un'attività o un'organizzazione nel raggiungere i propri obiettivi.

## L

### **Latex**

Sistema di composizione tipografica utilizzato per la produzione di documentazione tecnica e scientifica di alta qualità, particolarmente adatto per documenti complessi con formule matematiche.

### **Livelli aperti**

Architettura a livelli in cui un livello può utilizzare servizi di qualsiasi livello sottostante, non solo di quello immediatamente inferiore.

### **Livelli chiusi**

Architettura a livelli in cui un livello può utilizzare solo i servizi del livello immediatamente inferiore.

### **LLM (Large Language Model)**

Modello di linguaggio di grandi dimensioni addestrato su vasti corpus di testo, in grado di generare, comprendere e elaborare linguaggio naturale in modo sofisticato.

### **Load balancer**

Componente che distribuisce il carico di lavoro tra più server per migliorare le prestazioni e l'affidabilità del sistema.

## M

### **Microservizio**

Approccio architetturale in cui un'applicazione è composta da piccoli servizi indipendenti, ciascuno eseguibile autonomamente e comunicante tramite API.

### **Modularità**

Grado in cui un sistema è composto da componenti separati che possono essere sviluppati, testati e mantenuti indipendentemente.

## O

### **OCR (Optical Character Recognition)**

Tecnologia che converte immagini di testo scritto o stampato in testo digitale machine-readable.

## P

### **Polimorfismo**

Principio della programmazione orientata agli oggetti che permette a oggetti di classi diverse di rispondere allo stesso messaggio (metodo) in modo specifico per la propria classe, favorendo flessibilità ed estensibilità del codice.

### **Post-condizione**

Condizione o stato del sistema che deve essere vero dopo il completamento di un caso d'uso. Definisce il risultato atteso e le garanzie che il sistema fornisce al termine dell'esecuzione del caso d'uso.

### **Pre-condizione**

Condizione o stato del sistema che deve essere vero prima che un caso d'uso possa iniziare. Definisce i prerequisiti necessari per l'esecuzione corretta del caso d'uso.

### **Preventivo**

Documento di pianificazione che stima i costi e le risorse necessarie per lo svolgimento del progetto, basato sulle attività pianificate e sulle risorse disponibili.

### **Procedimento agile**

Approccio iterativo e incrementale allo sviluppo software che enfatizza la flessibilità e la collaborazione con il cliente.

### **Procedimento bottom-up**

Approccio allo sviluppo software che parte dai componenti di basso livello per costruire gradualmente sistemi più complessi.

### **Procedimento top-down**

Approccio allo sviluppo software che parte da una visione d'insieme per scomporre gradualmente il sistema in componenti più piccoli.

### **Progettazione di dettaglio**

Fase della progettazione software in cui si specificano nel dettaglio i componenti, le interfacce e le relazioni tra di essi.

### **Progettazione logica**

Fase della progettazione software in cui si definisce l'architettura del sistema senza considerare i dettagli implementativi specifici.

**Progettista**

Figura responsabile della progettazione dell'architettura software e delle soluzioni tecniche, garantendo che soddisfino i requisiti e siano realizzabili efficientemente.

**Programmatore**

Figura che implementa il codice sorgente secondo le specifiche tecniche, seguendo le best practice e gli standard di qualità definiti nel progetto.

**Progetto**

Insieme di attività che devono raggiungere obiettivi specifici a partire da date specifiche, con un inizio e una fine fissate in calendario, disponendo di risorse limitate (persone, tempo, denaro, strumenti) e consumando risorse nel loro svolgersi.

**Prompt**

Input (testo e/o altri dati) fornito a un modello di intelligenza artificiale per ottenere una risposta o un output specifico.

**Proponente**

Azienda o organizzazione che propone un capitolato d'appalto per il progetto del corso di Ingegneria del Software, definendone requisiti e obiettivi.

**Processi organizzativi**

Processi trasversali rispetto ai singoli progetti che riguardano la gestione dei processi, delle infrastrutture, del miglioramento continuo e della formazione del personale nell'organizzazione.

**Processi primari**

Processi fondamentali che agiscono direttamente sul ciclo di vita del prodotto software, includendo acquisizione, fornitura, sviluppo, operazione e manutenzione.

**Processi di supporto**

Processi che supportano i processi primari, includendo documentazione, gestione della configurazione, accertamento della qualità, verifica, validazione e risoluzione dei problemi.

**PWA (Progressive Web App)**

Applicazione web che utilizza tecnologie web moderne per offrire un'esperienza simile a quella di un'app nativa, funzionando offline e potendo essere installata sul dispositivo.

## Q

### **Qualità architetturale**

Insieme delle caratteristiche che definiscono la bontà di un'architettura software in termini di modularità, riusabilità, manutenibilità e altre proprietà desiderabili.

## R

### **Rating System**

Sistema di valutazione che assegna un punteggio o un giudizio a un'entità (es. prodotto, servizio, contenuto) in base a criteri prestabiliti.

### **RBAC (Role-Based Access Control)**

Modello di controllo degli accessi in cui i permessi sono assegnati a ruoli specifici, e gli utenti ottengono i permessi attraverso l'assegnazione a questi ruoli.

### **Redattore**

Membro del team responsabile della stesura e della produzione dei documenti di progetto, garantendo chiarezza, completezza e conformità alle norme stabilite.

### **Repository**

Archivio centrale in cui vengono memorizzati e versionati i file sorgente, la documentazione e le risorse del progetto utilizzando un sistema di controllo versione.

### **Requisito**

Condizione o capacità che deve essere posseduta da un sistema o componente software per soddisfare un contratto, standard, specifica o altro documento formalmente imposto.

### **Requisiti desiderabili**

Requisiti che sono importanti ma non essenziali per il funzionamento base del sistema. La loro implementazione apporta valore aggiunto ma la loro assenza non compromette il progetto.

### **Requisiti funzionali**

Specificano cosa il sistema deve fare, descrivendo le funzionalità, i comportamenti e le interazioni che il software deve supportare.

### **Requisiti non funzionali**

Definiscono come il sistema deve comportarsi in termini di prestazioni, sicurezza, affidabilità, usabilità e altri attributi di qualità, senza riguardo alle funzionalità specifiche.

### **Requisiti obbligatori**

Requisiti che devono essere necessariamente soddisfatti e la cui mancata implementazione comporterebbe il fallimento del progetto. Sono critici per il successo del sistema.

## **Requisiti opzionali**

Requisiti che sono utili ma non necessari, e la cui implementazione dipende dalla disponibilità di risorse e tempo. Possono essere considerati per versioni future del prodotto.

## **Requirements Baseline**

Insieme dei requisiti concordati e formalmente approvati che costituisce il riferimento per lo sviluppo del progetto. Una volta stabilita, qualsiasi modifica alla baseline dei requisiti deve seguire un processo formale di controllo delle modifiche.

## **Requisito software**

Requisito specificato in termini tecnici, destinato agli sviluppatori, che descrive in modo dettagliato e misurabile una funzionalità o un vincolo del sistema software.

## **Requisito utente**

Requisito espresso dal punto di vista dell'utente finale, descritto in linguaggio naturale e senza dettagli tecnici, focalizzato su ciò che l'utente si aspetta che il sistema faccia.

## **Responsabile**

Figura di riferimento del progetto con compiti di coordinamento, pianificazione, gestione delle risorse e comunicazione con docenti e proponenti.

## **Riusabilità**

Capacità di un componente software di essere utilizzato in diversi contesti o applicazioni senza modifiche sostanziali.

## **Robustezza**

Capacità di un sistema software di funzionare correttamente in condizioni anomale o in presenza di input non validi.

## S

### **Sandbox di Sviluppo**

Ambiente isolato per testare e validare funzionalità senza influenzare i sistemi di produzione, utilizzato per sviluppare e verificare nuove feature in sicurezza.

### **Scalabilità**

Capacità di un sistema di gestire un aumento del carico di lavoro aggiungendo risorse, senza modifiche all'architettura.

### **Scenario**

Sequenza di interazioni tra attori e sistema che descrive un percorso specifico attraverso un caso d'uso. Può essere principale (percorso di successo) o alternativo (variazioni ed eccezioni).

### **Scenario alternativo**

Sequenza di interazioni nel caso d'uso che rappresenta un percorso diverso da quello principale, tipicamente gestendo condizioni eccezionali, errori o scelte alternative dell'utente. Descrive come il sistema reagisce in situazioni non standard.

### **Scenario principale**

Sequenza di interazioni entre l'attore principale e il sistema che descrive il percorso di successo del caso d'uso, dove l'obiettivo viene raggiunto senza intoppi o condizioni eccezionali. Rappresenta il flusso ideale e più frequente di esecuzione.

### **Scope**

Ambito di un progetto, che definisce i confini, i deliverables, gli obiettivi e i compiti che sono inclusi nel progetto.

### **SCRUM**

Framework agile per la gestione dello sviluppo software che enfatizza lo sviluppo iterativo, l'adattamento ai cambiamenti e la consegna incrementale di valore.

### **SEMAT**

(Software Engineering Method and Theory) Iniziativa internazionale per rifondare l'ingegneria del software come disciplina rigorosa, basata su un kernel di elementi essenziali comuni a tutti i metodi di sviluppo software.

### **Sottotipizzazione**

Relazione entre tipi per cui un tipo (sottotipo) può essere utilizzato in ogni contesto in cui è atteso un altro tipo (supertipo), in accordo con il principio di sostituzione di Liskov.

**Specification Tecnica**

Documento che descrive in dettaglio l'architettura, o design e le scelte implementative del sistema software, guidando le attività di sviluppo.

**Sprint**

Periodo di tempo fisso (tipicamente 2-4 settimane) in Scrum durante il quale il team sviluppa e consegna un incremento di prodotto potenzialmente rilasciabile.

**Stakeholder**

Portatore di interesse, ovvero qualsiasi individuo, gruppo o organizzazione che può influenzare o essere influenzato da un progetto, un'azienda o un sistema.

**Stima dei costi**

Processo di previsione dei costi associati alle attività di progetto, considerando risorse umane, strumenti, infrastrutture e altri fattori che influenzano il budget complessivo.

## T

### **Technology Baseline**

Insieme delle tecnologie, framework, librerie e strumenti di sviluppo selezionati e approvati per il progetto. Definisce lo stack tecnologico di riferimento e costituisce la base per le scelte implementative, garantendo coerenza e standardizzazione nell'architettura software.

### **Test**

Processo sistematico di verifica che il software soddisfi i requisiti specificati e identifichi difetti, attraverso l'esecuzione controllata di casi di test.

### **Transazioni distribuite**

Transazioni che coinvolgono multiple risorse distribuite in diversi nodi di un sistema, richiedendo meccanismi di coordinamento per garantire atomicità e consistenza.

## U

### **Unità architetturali**

Componenti fondamentali che costituiscono l'architettura di un sistema software, come moduli, componenti, connettori e dati.

## V

### **Validatore**

Membro del team responsabile di eseguire attività de validazione, accertando che il prodotto software soddisfi le effettive esigenze del cliente e gli obiettivi di business.

### **Validazione**

Processo che accerta che il prodotto software sviluppato soddisfi le effettive esigenze del cliente e gli obiettivi di business per i quali è stato realizzato. Risponde alla domanda "Stiamo costruendo il prodotto giusto?" e viene tipicamente effettuata attraverso test di accettazione con il cliente.

### **Verifica**

Processo sistematico che determina se i prodotti di lavoro (documenti, codice, componenti) soddisfano i requisiti e le specifiche definite per loro. Risponde alla domanda "Stiamo costruendo il prodotto nel modo giusto?" e include attività come revisioni, ispezioni e test.

### **Verificatore**

Membro del team responsabile di controllare che documenti, codice e altri prodotti di lavoro rispettino gli standard de qualità definiti, le norme di progetto e siano privi di errori, incoerenze o ambiguità.

### **Versionamento/versioning**

Sistema per gestire le diverse versioni di file, codice sorgente o documenti, permettendo di tracciare le modifiche, collaborare in team e revertire a versioni precedenti.

## W

### **Way of Working**

Insieme di processi, metodologie, strumenti e pratiche adottati dal team per organizzare e svolgere le attività di progetto in modo coordinato ed efficiente. Definisce come il team collabora, comunica e gestisce il lavoro quotidiano.

### **Workflow**

Sequenza di attività che definiscono un processo di business, dove compiti, informazioni o documenti passano da un partecipante all’altro secondo regole prestabilite.