

UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI MATEMATICA "TULLIO LEVI-CIVITA"

CORSO DI LAUREA TRIENNALE IN

INFORMATICA

VISUALIZZAZIONE TRIDIMENSIONALE PER LA GESTIONE DI
MAGAZZINO

A.A. 2023/2024

RELATORE

Prof. Tullio Vardanega

STUDENTE

Riccardo Carraro

Matricola n. 2042346

“Non puoi risolvere un problema con lo stesso tipo di pensiero che hai usato per crearlo”

Albert Einstein

Sommario

Il presente documento descrive l'esperienza di tirocinio svolta presso l'azienda Sanmarco Informatica S.p.A. del laureando Riccardo Carraro, nel periodo 20 maggio - 12 luglio 2024.

L'obiettivo era lo sviluppo di una visualizzazione tridimensionale per la gestione di magazzino, dando la possibilità di creare ordini di movimentazione della merce in modo intuitivo e veloce mediante un'operazione di *drag & drop* direttamente nell'ambiente 3D. Il lavoro svolto è stato direttamente integrato nel *software* sviluppato dall'azienda, risultando in un'estensione delle funzionalità utilizzabili del prodotto.

Il documento è strutturato in quattro capitoli, quali:

- **L'azienda Sanmarco Informatica:** presenta il contesto organizzativo e produttivo in cui il laureando è stato inserito;
- **Il tirocinio:** descrive il progetto proposto, il rapporto dell'azienda con lo *stage* e le motivazioni che hanno portato alla scelta di questo progetto;
- **Svolgimento del tirocinio:** descrive il metodo di lavoro adottato, le attività svolte e i risultati ottenuti;
- **Valutazione retrospettiva:** riporta le considerazioni finali del laureando sul progetto svolto e sulle competenze acquisite.

Al fine di agevolare la lettura, il documento rispetta le seguenti convenzioni tipografiche:

- i termini in linguaggio diverso dall'italiano sono posti in *corsivo*;
- ogni immagine è corredata da una didascalia e la fonte da cui è stata tratta;
- i termini riportati nel glossario riportano una G posta a pedice.

In appendice è presente il glossario dei termini meno consueti utilizzati, insieme alla lista di abbreviazioni e acronimi e alla bibliografia e sitografia consultata.

Ringraziamenti

Ringraziamenti da definire

Indice dei contenuti

1 L’azienda Sanmarco Informatica	1
1.1 Presentazione dell’azienda	1
1.2 Organizzazione aziendale e i prodotti	1
1.3 I clienti	3
1.4 Processi	3
1.4.1 Modello di sviluppo	3
1.4.2 Ruoli aziendali	4
1.4.3 Processi primari	5
1.4.3.1 Fornitura	5
1.4.3.2 Sviluppo	6
1.4.3.3 Manutenzione	7
1.4.4 Processi di supporto	8
1.4.4.1 Documentazione	8
1.4.4.2 Verifica	9
1.4.5 Processi organizzativi	10
1.4.5.1 Gestione dell’infrastruttura	10
1.4.5.2 Strumenti di tracciamento delle attività	10
1.4.5.3 Strumenti di comunicazione	12
1.4.5.4 Strumenti documentali	13
1.4.5.5 Strumenti di sviluppo	14
1.4.5.6 Integrazione degli strumenti	17
1.4.5.7 Gestione delle risorse umane	18
1.5 Il ruolo dell’innovazione	18
2 Il tirocinio	20
2.1 Il ruolo dello stage per Sanmarco Informatica	20
2.2 Il progetto proposto	20
2.2.1 Descrizione del progetto	20
2.2.2 Obiettivi	21
2.2.2.1 Obiettivi aziendali	21
2.2.2.2 Obiettivi personali	22
2.2.3 Vincoli	24
2.2.3.1 Vincoli temporali	24
2.2.3.2 Vincoli tecnologici	24
2.2.3.3 Vincoli organizzativi	24
2.3 Motivazione della scelta	25
3 Svolgimento del tirocinio	27
3.1 Pianificazione	27

3.2 Metodo di lavoro	30
3.2.1 <i>Way of Working</i>	30
3.2.2 Obiettivi di qualità	32
3.2.3 Obiettivi di qualità di processo	33
3.2.4 Interazione con il referente aziendale	34
3.2.5 Revisioni di progresso	34
3.2.6 Strumenti di verifica	34
3.2.7 Resoconti	37
3.3 Analisi dei requisiti	37
3.3.1 Casi d'uso	37
3.3.2 Tracciamento dei requisiti	37
3.4 Progettazione	37
3.4.1 Tecnologie utilizzate	37
3.4.2 Progettazione dell'ambiente tridimensionale	38
3.4.3 Progettazione della funzionalità <i>drag & drop</i>	38
3.4.4 Architettura del sistema	38
3.4.5 Design pattern	38
3.5 Codifica	38
3.5.1 Visualizzazione tridimensionale	38
3.5.1.1 Classi implementate	38
3.5.1.2 Cambiamenti apportati	38
3.5.2 Drag & Drop e creazione ordini di movimentazione	38
3.5.2.1 Componenti	38
3.5.2.2 Servizi	38
3.5.2.3 Servizi REST	38
3.6 Verifica e validazione	39
3.6.1 Test di unità	39
3.6.2 Test di integrazione	39
3.6.3 Test di <i>performance</i>	39
3.6.4 Test di sistema	39
3.6.5 Test di accettazione	39
3.7 Risultati raggiunti	39
3.7.1 Il prodotto realizzato	39
3.7.2 Copertura dei requisiti	39
3.7.3 Copertura di testing	39
3.7.4 Materiali prodotti	39
4 Valutazione retrospettiva	41
4.1 Soddisfacimento degli obiettivi	41
4.1.1 Obiettivi aziendali	41
4.1.2 Obiettivi personali	41

4.2 Competenze acquisite	41
4.3 Valutazione personale	41
4.4 Università e mondo del lavoro	41
Acronimi e abbreviazioni	42
Glossario	43
Bibliografia e sitografia	48

Indice delle immagini

Immagine 1: Divisione in <i>business unit</i>	2
Immagine 2: Modello di sviluppo <i>Agile</i>	3
Immagine 3: Relazione <i>User Stories</i> , <i>Product Backlog</i> e <i>Sprint Backlog</i>	6
Immagine 4: Manutenzione <i>software</i>	7
Immagine 5: Le tipologie di <i>Software testing</i>	9
Immagine 6: Esempio di <i>board</i> in Jira	11
Immagine 7: Interfaccia di Google Meet	12
Immagine 8: Interfaccia di Scrumlr.io	13
Immagine 9: Interfaccia di <i>Google Sheets</i>	13
Immagine 10: Interfaccia di <i>Confluence</i>	14
Immagine 11: Interfaccia di Bitbucket	15
Immagine 12: Interfaccia di <i>VSCode</i> con il codice <i>frontend</i> del prodotto del tirocinio	15
Immagine 13: Interfaccia di <i>IntelliJ</i> con il codice <i>backend</i> del prodotto del tirocinio	16
Immagine 14: Interfaccia di DBeaver con il <i>database</i> del prodotto del tirocinio	16
Immagine 15: Esempio di chiamata POST ad un servizio REST con Postman ..	17
Immagine 16: Come gli strumenti si integrano nel modello di sviluppo aziendale	17
Immagine 17: Corso di formazione Angular su Udemy	18
Immagine 18: Come le funzionalità sviluppate nel tirocinio si integrano tra loro nel prodotto WMS	21
Immagine 19: Diagramma di Gantt delle attività del primo periodo	28
Immagine 20: Diagramma di Gantt delle attività del secondo periodo	28
Immagine 21: Diagramma di Gantt delle attività del terzo periodo	29
Immagine 22: Diagramma di Gantt delle attività del quarto periodo	30
Immagine 23: Diagramma di Gantt complessivo delle attività svolte durante il tirocinio	30

Immagine 24: L'importanza del <i>Way of Working</i> nel SEMAT	31
Immagine 25: Bacheca personale su Notion	32
Immagine 26: Pipeline di <i>Continuous Integration</i> e <i>Continuous Deployment</i>	35
Immagine 27: <i>Pipeline</i> per l'accettazione di una <i>Pull Request</i>	36

Indice delle tabelle

Tabella 1: Ruoli aziendali	5
Tabella 2: Obiettivi aziendali	22
Tabella 3: Obiettivi personali	23
Tabella 4: Macrosuddivisione del tirocinio	27

1 L’azienda Sanmarco Informatica

1.1 Presentazione dell’azienda

Sanmarco Informatica S.p.A è un’azienda nata nel 1984 specializzata nello sviluppo *software* e nella consulenza informatica.

Con oltre 2500 clienti e più di 650 dipendenti, Sanmarco Informatica opera in uffici distribuiti in molteplici regioni italiane, quali Trentino Alto Adige, Friuli-Venezia Giulia, Lombardia, Piemonte, Emilia-Romagna, Toscana, Campania, Puglia e Veneto, con sede principale a Grisignano di Zocco (VI), poco distante dal Centro Ricerca e Sviluppo in cui ho svolto il tirocinio.

L’obiettivo dell’azienda è l’innovazione delle aziende clienti, agevolandone la trasformazione digitale, progettando e realizzando soluzioni digitali integrate.

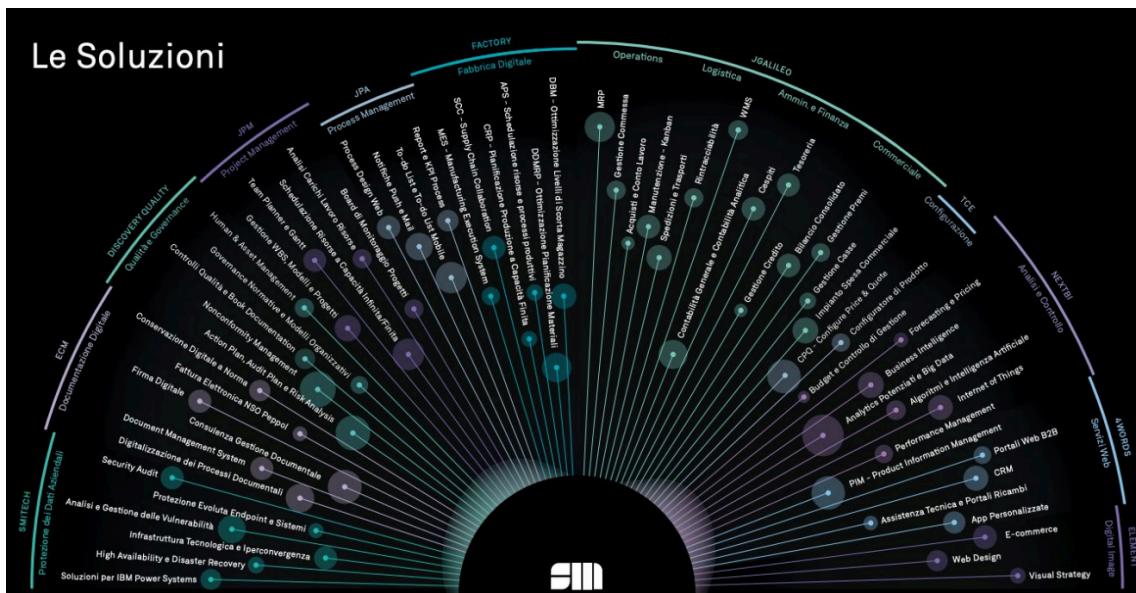
1.2 Organizzazione aziendale e i prodotti

Durante il periodo di tirocinio ho potuto osservare da vicino l’organizzazione che l’azienda segue. Sanmarco Informatica è organizzata in diverse *business unit* (BU), ciascuna in grado di operare in modo autonomo o semi-autonomo, con l’obiettivo di garantire al cliente finale servizi e prodotti di qualità, adattandosi alle diverse esigenze del mercato.

Le BU in cui l’azienda è suddivisa sono undici, ciascuna specializzata in un settore specifico:

- **SMITECH:** specializzata in *Cybersecurity* e protezione dei dati, offre servizi di consulenza, formazione e soluzioni tecnologiche per garantire la sicurezza informatica.
- **ECM:** offre soluzioni di *Enterprise Content Management* (ECM) per una gestione efficiente dei documenti digitali, includendo strumenti per la gestione dei contenuti, la collaborazione e la condivisione dei documenti;
- **DISCOVERY QUALITY:** sviluppa *software* per la *governance* aziendale, il controllo dei processi e la misurazione delle *performance*, con attenzione alle normative e alle metriche di sostenibilità (*Sustainable Development Goals* (SDGs), *Benefit Corporation* (BCorp)), per assicurare la qualità di prodotti e servizi;
- **JPM:** fornisce soluzioni di *Project Management* per la gestione dei progetti, con strumenti per la pianificazione, il monitoraggio e il controllo su commessa o a preventivo;

- **JPA:** sviluppa *software* di *Business Process Management* (BPM) per l’automazione e l’integrazione dei processi aziendali, offrendo una piattaforma completa con un *designer* grafico per la loro modellazione, un motore per l’esecuzione e un’interfaccia grafica per la gestione dei *task* assegnati agli utenti;
 - **FACTORY:** soddisfa le esigenze della *Supply Chain* con soluzioni per la fabbrica del futuro, focalizzate sull’ottimizzazione del servizio clienti, degli asset e dei profitti. Fornisce inoltre soluzioni per la gestione dei magazzini e della produzione. Si tratta della *business unit* in cui ho svolto il tirocinio;
 - **JGALILEO:** sviluppa JGalileo, una soluzione di *Enterprise Resource Planning* (ERP) integrata progettata per ottimizzare i processi aziendali delle imprese, con un focus particolare sulle normative fiscali di carattere internazionale;
 - **TCE:** si impegna a semplificare i processi di preventivazione e acquisizione ordini attraverso il prodotto CPQ, che consente una configurazione rapida e precisa di prodotti e servizi;
 - **NEXTBI:** specializzata in *Information Technology* e consulenza strategica, con competenze specifiche in *marketing*, vendite, retail, innovazione per il cliente, *Business Intelligence* e soluzioni *Internet of Things* (IoT);
 - **4WORDS:** propone soluzioni *Business to Business* (B2B), applicazioni e *Customer Relationship Management* (CRM) per potenziare il business attraverso strumenti digitali, inclusi portali B2B e realtà aumentata;
 - **ELEMENT:** è la divisione creativa specializzata nello sviluppo di siti *web* ed *e-commerce*, con particolare attenzione all’esperienza utente e all’interfaccia grafica.



Imagine 1: Divisione in *business unit*

Fonte: <https://www.udemy.com/course/the-complete-guide-to-angular-2/>

1.3 I clienti

Il portfolio clienti di Sanmarco Informatica vanta più di 2500 aziende, da piccole/medie imprese a grandi aziende internazionali.

DalterFood Group (*leader* nel settore lattiero caseario e della distribuzione internazionale di prodotti alimentari), *Orange1 Holding* (gruppo industriale attivo nel settore della produzione di motori elettrici, con stabilimenti in Italia e all'estero) e *Cigierre S.p.A.* (*leader* nello sviluppo e gestione di ristoranti tematici) sono solo alcuni dei clienti di maggiore rilievo per l'azienda, ma offrono una panoramica della diversità dei settori in cui i clienti di Sanmarco Informatica operano.

Durante il mio periodo di tirocinio, ho avuto modo di assistere al rapporto che l'azienda instaura con i propri clienti, caratterizzato da contatti costanti ed incontri frequenti, sia in presenza che a distanza. Inoltre, per ogni prodotto e servizio che l'azienda offre, è previsto un consulente specializzato che segue il cliente per ogni necessità.

1.4 Processi

1.4.1 Modello di sviluppo

Durante il mio tirocinio, ho osservato da vicino il modello di sviluppo *software* utilizzato dall'azienda: Sanmarco Informatica opera mediante un modello di sviluppo *Agile_G*, implementando nello specifico il *framework Scrum_G*. Per quanto avessi già familiarità con questo modello grazie ai corsi di “Ingegneria del software” e “Metodologie e Tecnologie per lo sviluppo software” frequentati durante il corso di laurea, il tirocinio mi ha permesso di osservare in prima persona come questo modello venga applicato in un contesto aziendale.

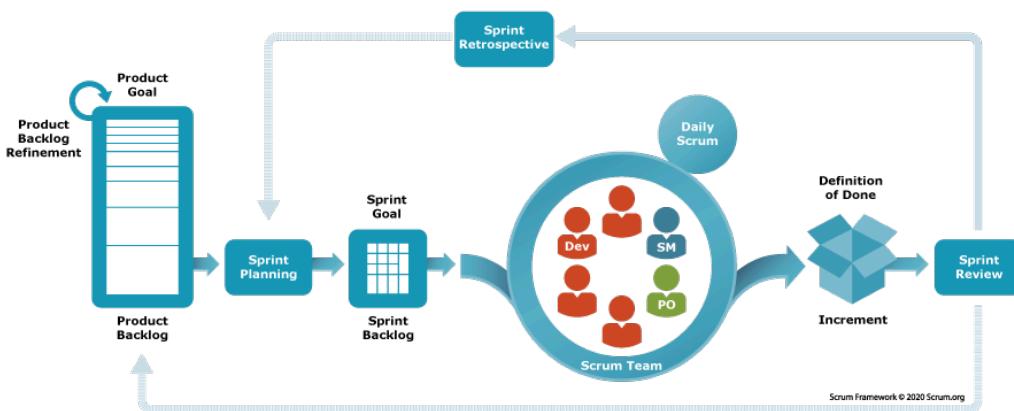


Immagine 2: Modello di sviluppo *Agile_G*

Fonte: https://www.scrum.org/resources/what-is-G-scrum_G

Quanto mostro nell'[immagine 2](#) rappresenta l'insieme di attività e processi che vengono istanziati dall'azienda nella realizzazione di un prodotto *software*.

Il concetto cardine del modello *Agile_G* sono le *User Stories* definite in collaborazione con il cliente, sulla base delle quali si andrà a definire il *Product Backlog_G*, ovvero l'insieme di tutte i *task_G* che il *team* di sviluppo dovrà svolgere al fine di implementare le funzionalità desiderate.

Il modello *Agile_G* suddivide il periodo di realizzazione in *Sprint_G*, ossia iterazioni di sviluppo di durata fissa (nel caso di Sanmarco Informatica di 4 settimane), durante le quali il *team* si impegna a sviluppare l'insieme di funzionalità definite all'interno dello *Sprint_G Backlog_G*.

Per assicurare un allineamento costante tra ogni membro del *team* in merito allo stato di avanzamento, si svolgono *Daily Standup Meeting*, brevi incontri quotidiani durante i quali ogni membro del *team* informa gli altri membri in merito al proprio lavoro svolto e le eventuali difficoltà riscontrate.

Svolgendo questa attività quotidianamente, ho avuto la riprova di quanto sia importante la comunicazione all'interno di un *team* di sviluppo, in quanto permette di mantenere un allineamento costante tra i membri e di risolvere eventuali problemi in modo rapido ed efficace.

Al termine di ogni periodo di sviluppo, si svolge una retrospettiva per valutare i risultati dello *Sprint_G*, denominata *Sprint_G Review_G*, durante la quale il *team* presenta il progresso ottenuto, susseguita successivamente dalla *Sprint_G Retrospective*, che ha l'obiettivo di far riflettere sul lavoro svolto e sulle modalità con cui poter migliorare il processo di sviluppo.

Solo a questo punto, si procede alla pianificazione dello *Sprint_G* successivo e alla definizione del nuovo *Sprint_G Backlog_G*.

Durante il mio tirocinio, ho partecipato attivamente a tutte le attività sopra descritte, concretizzando quanto appreso durante il corso di laurea in un contesto aziendale.

1.4.2 Ruoli aziendali

La corretta implementazione del *framework Scrum_G* richiede l'individuazione di ruoli chiave, ciascuno con compiti e responsabilità ben definite. In azienda, ho avuto modo di osservare i seguenti ruoli:

Ruolo	Mansioni
Product Owner	Responsabile della definizione delle funzionalità del prodotto, in collaborazione con il cliente. Si occupa di definire il <i>Product Backlog</i> _G e di priorizzare le <i>User Stories</i> in base alle esigenze del cliente.
Team leader	Responsabile del coordinamento del <i>team</i> di sviluppo, si occupa di assegnare i compiti e di garantire che il <i>team</i> sia allineato con gli obiettivi dello <i>Sprint</i> _G .
Sviluppatore	Responsabile della realizzazione effettiva delle funzionalità del prodotto.
Tester	Responsabile della verifica del prodotto, si occupa di testare le funzionalità implementate e di segnalare eventuali <i>bug</i> _G al <i>team</i> di sviluppo.
Consulente	Responsabile dell'installazione del prodotto presso il cliente, si occupa di garantire che il prodotto soddisfi le esigenze del cliente.

Tabella 1: Ruoli aziendali

Come ho potuto osservare in azienda, questa suddivisione di compiti e responsabilità, permette di affrontare in modo efficace e organizzato il processo di sviluppo, garantendo che i diversi aspetti del prodotto siano in grado di avanzare in modo parallelo e coordinato.

1.4.3 Processi primari

1.4.3.1 Fornitura

Il processo di fornitura è il processo che si occupa di definire i requisiti del prodotto, di pianificare le attività di sviluppo e di garantire che il prodotto soddisfi le esigenze del cliente. Durante il mio tirocinio ho avuto modo di osservare come questo processo venga attuato in azienda, partendo dalla definizione dei requisiti del prodotto in collaborazione con il cliente, fino alla realizzazione del prodotto stesso.

Tra le peculiarità del modello *Agile*_G infatti, vi è la capacità di adattamento dello sviluppo ai cambiamenti, ottenibile mediante una stretta collaborazione tra il *Product Owner* e il cliente.

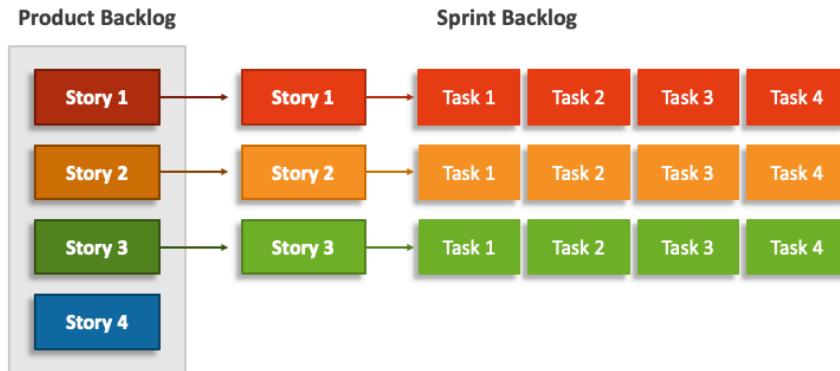


Immagine 3: Relazione *User Stories*, *Product Backlog_G* e *Sprint_G Backlog_G*

Fonte: https://www.collidu.com/presentation-product-backlog_G

Con l'[immagine 3](#) mostro come le *User Stories* siano l'*input* fondamentale per la definizione del *Product Backlog_G* e dello *Sprint_G Backlog_G*, responsabili del delineamento delle funzionalità del prodotto e delle attività da svolgere durante lo *Sprint_G*.

Da quanto ho potuto constatare durante il mio tirocinio, ogni incontro tra il *Product Owner* e il cliente, non solo permetteva di mostrare i risultati fino a quel momento ottenuti dal *team*, ma produceva come risultato un documento di analisi che raccoglieva gli eventuali cambiamenti e le nuove funzionalità richieste dal cliente.

Questa analisi, andava ad integrare la documentazione presente su *Confluence*, la piattaforma utilizzata dall'azienda per la documentazione, e, nel *meeting* di pianificazione dello *Sprint_G* successivo, veniva discussa e valutata insieme al *team* di sviluppo.

1.4.3.2 Sviluppo

Il processo di sviluppo è quello che più da vicino ho potuto osservare durante il mio tirocinio. Questo processo è stato caratterizzato da precise attività, ciascuna con obiettivi e risultati ben definiti.

Il processo di sviluppo si articola nelle seguenti attività principali:

- **Software requirements analysis:** attività di analisi dei requisiti del prodotto. Il suo obiettivo è definire i requisiti del prodotto a partire da quanto emerso dai *meeting* con il cliente e dal documento di analisi prodotto dal *Product Owner* durante il processo di fornitura ([paragrafo 1.4.3.1](#)); I *meeting* di analisi che ho svolto insieme al *team*, hanno avuto durata media ci circa 4 ore, e sono sempre terminati con la rendicontazione delle decisioni prese nella piattaforma *Confluence*.
- **Software detailed design:** attività di progettazione dettagliata del prodotto. Il suo obiettivo è definire l'architettura del prodotto e i dettagli di implemen-

tazione delle funzionalità. Durante il mio tirocinio ho avuto modo di partecipare attivamente a questa attività, in particolare nella progettazione dell’ambiente tridimensionale e della funzionalità di *drag & drop*. Anche in questo caso, le decisioni prese durante i *meeting* di progettazione sono state documentate su *Confluence*, facendo altresì utilizzo di diagrammi UML_G e *mockup* dell’interfaccia.

- **Software coding and testing_G**: attività di codifica e *test* del prodotto. Il suo obiettivo è l’implementazione delle funzionalità e verificare che siano conformi alle aspettative. Il *testing_G* in questo caso si concentra maggiormente sui *test* di unità e di integrazione, con l’obiettivo di garantire che il prodotto sia pronto per il *Software qualification testing_G*.
- **Software qualification testing_G**: attività di *test* di qualifica del prodotto. Il suo obiettivo è verificare che il prodotto soddisfi i requisiti del cliente e che sia pronto per la consegna. In Sanmarco Informatica, questa attività è svolta da una figura specializzata (*tester*) che si occupa di testare le funzionalità implementate e di segnalare eventuali problematiche al *team* di sviluppo.

Questi processi si integrano perfettamente con le pratiche di *continuous integration_G*, dove grazie allo strumento di controllo di versione Bitbucket_G ([paragrafo 1.4.5.5](#)), ad ogni modifica apportata alla *codebase_G* viene attivata una *pipeline* di *build* e *test* automatici.

1.4.3.3 Manutenzione

Lo sviluppo del *software* non termina con la consegna del prodotto al cliente: il processo di manutenzione ricopre un ruolo fondamentale per garantire che il prodotto sia sempre funzionante e allineato alle esigenze del cliente.

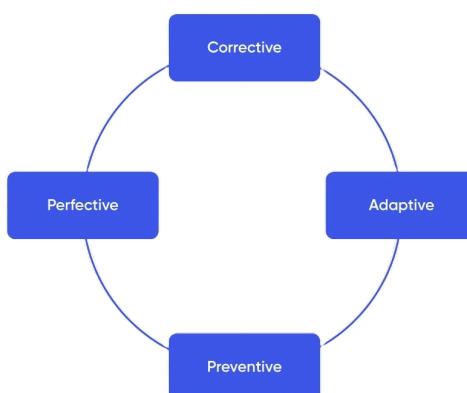


Immagine 4: Manutenzione *software*

Fonte: <https://cleancommit.io/blog/importance-of-software-maintenance-in-software-engineering/>

Come mostro nell'[immagine 4](#), la manutenzione del *software* possiede diversi aspetti, ciascuno con obiettivi ben definiti. Nel mio tirocinio mi è stato possibile notare come l'azienda si preoccupi della manutenzione dei prodotti *software* sviluppati, con l'obiettivo non solo di rispondere alle esigenze del cliente, ma anche di risolvere eventuali problematiche riscontrate.

Ho potuto individuare tre tipologie di manutenzione:

- **Manutenzione correttiva:** attività di correzione di *bug* e problematiche riscontrate nel prodotto. Nasce solitamente da segnalazioni del *tester* o del cliente. Nelle prime settimane del mio percorso, prima di procedere a lavorare alle nuove funzionalità, per approcciarmi al prodotto, ho svolto proprio attività di *bugfixing* su funzionalità già implementate;
- **Manutenzione adattativa:** attività di adattamento del prodotto a nuove esigenze del cliente. Nasce solitamente da nuove funzionalità richieste;
- **Manutenzione evolutiva:** attività di evoluzione del prodotto. Nasce solitamente dall'azienda stessa, con l'obiettivo di migliorare il prodotto e renderlo più competitivo sul mercato.

Un esempio concreto è relativo al *framework* proprietario *Synergy* ([paragrafo 2.2.3.2](#)), il cui sviluppo ed evoluzione è seguito da un *team* dedicato. Questo *framework* infatti si trova alla base di tutti i prodotti *software* sviluppati dall'azienda, e la sua manutenzione è fondamentale affinché questi siano in grado di rispondere non solo alle esigenze del cliente, ma anche alle evoluzioni delle tecnologie con cui si integra.

1.4.4 Processi di supporto

1.4.4.1 Documentazione

La documentazione è un aspetto fondamentale per garantire la qualità del prodotto *software* e la sua manutenibilità. Tra gli obiettivi del mio tirocinio (discussi nel dettaglio nel [paragrafo 2.2.2](#)), vi era infatti anche la produzione di documentazione relativa non solo alle funzionalità implementate, ma anche alla loro analisi e progettazione.

Come risultato di ogni *meeting* il *team* si occupa di documentare le decisioni prese, le funzionalità implementate e le problematiche riscontrate, utilizzando la piattaforma *Confluence*.

Anche l'approccio al *framework* *Synergy*, è stato un'ulteriore conferma in merito all'importanza della documentazione del *software*: trattandosi di un *framework* proprietario, la mia unica fonte di informazioni in merito al suo corretto utilizzo, resideva nella documentazione presente su *Confluence*, e per questo motivo, il

suo aggiornamento costante e la sua completezza erano aspetti fondamentali per permettere a me (e anche ai nuovi colleghi) di utilizzarlo in modo efficace ed efficiente.

Inoltre, anche all'interno del codice mi sono assicurato di seguire le convezioni aziendali in materia di commenti e produzione dei *Javadoc*, in modo da garantire che ogni porzione di codice da me prodotta fosse conforme, documentata e rapidamente comprensibile.

1.4.4.2 Verifica

Il processo di verifica comprende l'insieme di attività necessarie per garantire che il prodotto *software* soddisfi i requisiti del cliente e che sia pronto per la consegna. Durante il mio tirocinio ho avuto modo di osservare come questa attività sia svolta in azienda, partendo dai *test* di unità e di integrazione, fino ai *test* di sistema e di accettazione.

A seguito al processo di progettazione, vengono identificati e definiti i requisiti del prodotto, e per ciascun di questi definiti i test necessari per verificarne il loro soddisfacimento.

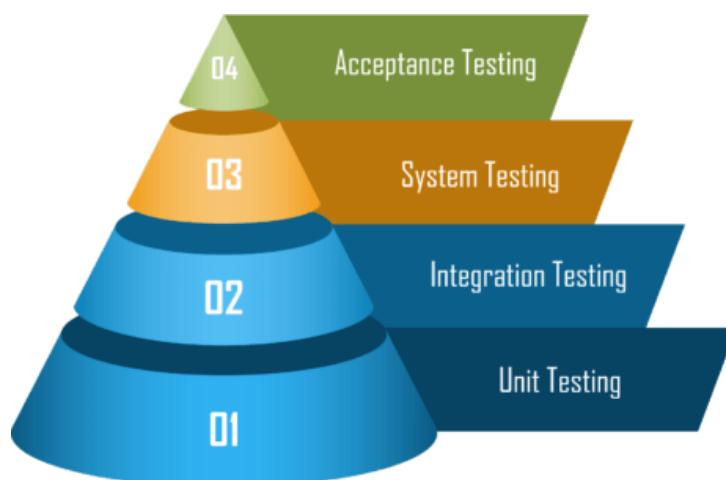


Immagine 5: Le tipologie di *Software testing*_G

Fonte: <https://www.tuleap.org/software-quality-different-types-software-testing>

G

Come mostro nell'[immagine 5](#), il processo di verifica comprende diversi tipi di *test*, ciascuno con obiettivi ben definiti:

- **Test di unità_G**: attività di verifica delle singole unità di codice, dove con unità si intende la minima porzione di codice dotata di comportamento autonomo. Il suo obiettivo è verificare che ciascuna unità funzioni correttamente e che sia conforme alle specifiche. La loro implementazione è predisposta dal *framework Synergy*, e la loro esecuzione è automatica.

-
- **Test di integrazione_G**: attività di verifica dell’integrazione tra le diverse unità di codice. Il suo obiettivo è verificare che le unità funzionino correttamente anche quando integrate tra loro. La loro implementazione è predisposta dal *framework Synergy*, ma sarà poi a cura dello sviluppatore implementare i *test* relativi a logiche e controlli più complessi. La loro esecuzione è automatica.
 - **Test di sistema_G**: attività di verifica del prodotto nel suo complesso. L’obiettivo pertanto è verificare che il prodotto soddisfi quanto emerso dai requisiti e che il suo comportamento sia conforme alle aspettative.
 - **Test di accettazione_G**: attività di verifica del prodotto da parte del cliente. L’obiettivo è verificare che il prodotto soddisfi le esigenze del cliente e che sia pronto per la consegna. Questa tipologia di *test* viene in un primo momento svolta dal *tester* del *team*, sia manualmente che in modo automatico.

In azienda ho partecipato attivamente a queste attività, in particolare ai test di unità_G e di integrazione, con l’obiettivo di garantire che il prodotto fosse pronto per il *Software qualification testing_G* ([paragrafo 1.4.3.2](#)).

Nel mio caso infatti, prima di procedere all’integrazione della *codebase_G* con il mio lavoro svolto, un automatismo si occupava di verificare che tutte le *suite* di *test* predisposte fossero eseguite con esito positivo, in modo da non compromettere il funzionamento del prodotto.

1.4.5 Processi organizzativi

1.4.5.1 Gestione dell’infrastruttura

Al fine di gestire in modo efficiente ed efficace i processi istanziati, l’azienda si avvale di strumenti e tecnologie che possano coprire i diversi aspetti dello sviluppo. Comprendere il loro corretto utilizzo e funzionamento è stato per me un aspetto fondamentale per poter svolgere il mio tirocinio.

Nei successivi paragrafi descriverò l’infrastruttura che ho avuto modo di osservare, presentando le tecnologie utilizzate e come queste siano state integrate nei processi aziendali.

1.4.5.2 Strumenti di tracciamento delle attività

Jira_G

Jira_G è uno strumento di *issue tracking system_G* (ITS_G) utilizzato dall’azienda per la gestione delle attività di sviluppo. Lo strumento permette al *team leader* ad ogni *Sprint_G planning*, di strutturare la *board* con i diversi *task_G* (o *issue_G*) da svolgere

entro la fine dello *Sprint_G*, assegnando a ciascun membro del *team* i compiti da svolgere.

Il tracciamento delle attività è fondamentale per garantire che il *team* sia allineato con gli obiettivi, permettendo di avere sempre una visione di insieme dello stato di avanzamento dei lavori.

Come mostrato nell'[immagine 6](#), Jira_G permette di strutturare la *board* in modo da avere una visione di insieme delle attività da svolgere, con la possibilità di organizzare i *task_G* in colonne in base allo stato di avanzamento.

Durante il mio tirocinio ho utilizzato lo strumento secondo le convenzioni aziendali, lavorando su *task_G* di due tipologie principali:

- **Bug_G**: attività di correzione di *bug_G* e problematiche riscontrate nel prodotto;
- **User story_G**: attività di implementazione di nuove funzionalità.

Lo svolgimento di queste attività seguiva una *pipeline* di stati ben definita:

- **To do**: il *task_G* è stato creato;
- **In progress**: il *task_G* è in corso di svolgimento: questo stato è sinonimo della presenza di un *branch_G* di sviluppo attivo, e che uno o più membri del *team* stanno lavorando al *task_G*;
- **Ready for test**: il *task_G* è stato completato e il lavoro prodotto è pronto per essere sottoposto al *software qualification test* ([paragrafo 1.4.3.2](#)). Il *task_G* viene ora assegnato al *tester* del *team* e, a seconda del risultato dei *test* condotti, il *task_G* può tornare in *In progress* o essere spostato in *Done*;
- **Done**: il *task_G* è stato completato con successo.

Le integrazioni con strumenti come Bitbucket_G ([paragrafo 1.4.5.5](#)) rendono Jira_G uno strumento estremamente versatile e in grado di adattarsi alle diverse esigenze dell'azienda.

The screenshot shows a Jira board titled "BREW board". The board has three main columns: "TO DO", "IN PROGRESS", and "DONE".

- TO DO:**
 - [Task] Test new Board 1 (vanilla extract) - **READY FOR TEST** (BREW-12)
 - [Task] Social content: Week 1 - **PY23 LAUNCH PLAN** (BREW-6)
 - [Task] Taste test: Latte glasses - **PY23 LAUNCH PLAN** (BREW-22)
- IN PROGRESS:**
 - [Task] Content with **BLOCK KAVIS** (BREW-1)
 - [Task] Update project plan with key milestones - **PY23 LAUNCH PLAN** (BREW-10)
 - [Task] Journey mapping workshop w/ Beanz - **PY23 LAUNCH PLAN** (BREW-11)
 - [Task] Explore personas: Geoff - **PY23 LAUNCH PLAN** (BREW-15)
 - [Task] Taste test: Fibre white cups - **READY FOR TEST** (BREW-23)
- DONE:**
 - [Task] Comp. analysis - Food delivery - **PY23 LAUNCH PLAN** (BREW-2)
 - [Task] Comp. analysis - Custom menus - **PY23 LAUNCH PLAN** (BREW-3)
 - [Task] Something's up with the load screen - **PY23 LAUNCH PLAN** (BREW-4)
 - [Task] PY23 Vision Storyboarding - **PY23 LAUNCH PLAN** (BREW-14)
 - [Task] Review banner ads - **READY FOR TEST** (BREW-40)
 - [Task] Onboarding tour refinements - **PY23 LAUNCH PLAN** (BREW-21)

Immagine 6: Esempio di *board* in Jira_G

Fonte: https://www.atlassian.com/it/software/jira_G/guides/boards/overview#what-is_G-a-jira_G-board

1.4.5.3 Strumenti di comunicazione

Google Meet e Google Chat

Sanmarco Informatica fa utilizzo della *suite* di strumenti offerta da Google per la comunicazione interna, in particolar modo Google Meet per le riunioni e Google Chat per la comunicazione testuale.

Google meet è uno strumento che permette di organizzare riunioni virtuali, con la possibilità di condividere schermo e documenti, e di registrare la riunione stessa.

Durante il mio tirocinio ho partecipato a diverse riunioni utilizzando questo strumento, in particolar modo ai *Daily Standup Meeting* (quando il *team* operava in remoto) e ai *meeting* di *Sprint_G Review_G* e *Sprint_G Retrospective* ([paragrafo 1.4.1](#)), dove mediante la condivisione dello schermo, il *team* presentava i risultati ottenuti.

Google Chat d'altro canto, è uno strumento di messaggistica istantanea che permette di comunicare in modo rapido e diretto con i colleghi. Ho utilizzato questo strumento per comunicare con i membri del *team* e per risolvere eventuali problematiche riscontrate durante lo sviluppo quando non era possibile un contatto diretto o si trattava di comunicazioni non urgenti.

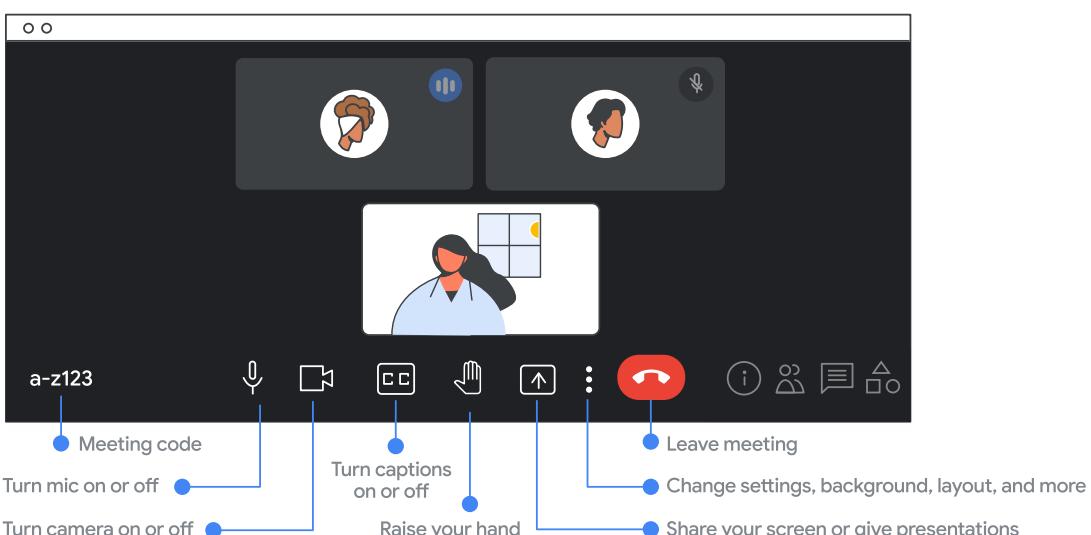


Immagine 7: Interfaccia di Google Meet

Fonte: <https://support.google.com/meet/answer/10550593?hl=it>

Scrumlr.io

Scrumlr.io è uno strumento che permette di creare diverse tipologie di *board* in supporto alla *Sprint_G Retrospective*, dove ogni membro del *team* può inserire i propri *feedback* e le proprie considerazioni relative allo *Sprint_G* concluso.

Nei *meeting* di retrospettiva che ho svolto, la *board* era divisa in **Kudos** (*feedback*

positivi ad uno o più membri del *team*), **Positive** (cosa è andato bene), **Negative** (cosa non è andato bene) e **Action** (azioni da intraprendere per migliorare i processi aziendali delineate dal *team leader*).

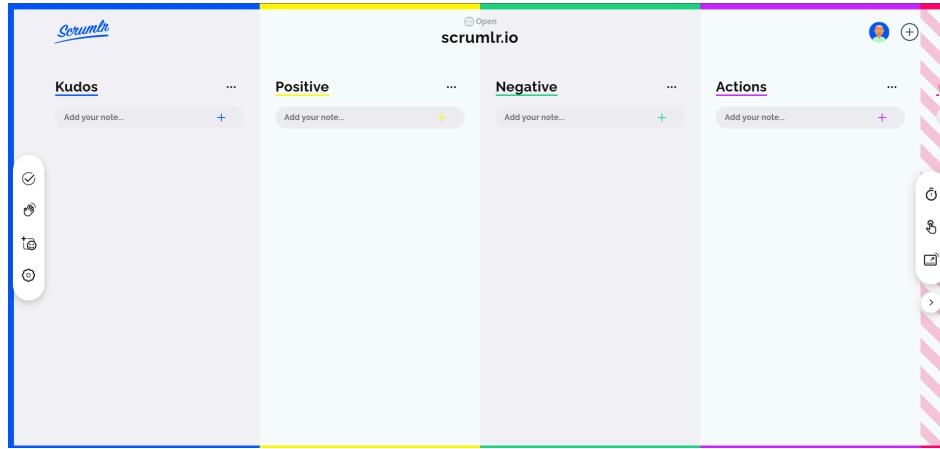


Immagine 8: Interfaccia di Scrumlr.io

Fonte: <https://www.scrumlr.io/>

1.4.5.4 Strumenti documentali

Google Sheets_G

Google Sheets_G è uno strumento di foglio elettronico che permette di creare e condividere documenti in modo collaborativo, specializzato nella rappresentazione di dati in forma tabellare.

Lo strumento è utilizzato dal *team* per la definizione delle tabelle relative al *database_G* del prodotto e per il tracciamento dei requisiti che intendono soddisfare.

Sale ID	Date	Point of Sale	Amount	Shirt 1			Shirt 2		
				Size	Design	Type	Size	Design	Type
1	12/1/2023	Online	4	L	Design 5 - Graffiti	V-neck	L	Design 2 - Sunset	Crewneck
2	12/2/2023	Michaela Ho	1	M	Design 4 - Skyline	V-neck	M	Design 2 - Sunset	Long-sleeved
3	12/2/2023	Online	5	M	Design 1 - Waves	V-neck	M	Design 2 - Sunset	Sweatshirt
4	12/3/2023	Online	2	M	Design 5 - Graffiti	V-neck	L	Design 3 - Boat	V-neck
5	12/4/2023	Online	3	M	Design 3 - Boat	Long-sleeved	M	Design 5 - Graffiti	Midtown
6	12/6/2023	Natasha Young	3	M	Design 5 - Graffiti	Tank Top	L	Design 6 - Midtown	Long-sleeved
7	12/7/2023	Danielo Rosas	1	L	Design 1 - Waves	V-neck	M	Design 5 - Graffiti	Sweatshirt
8	12/8/2023	Danielo Rosas	4	M	Design 2 - Sunset	V-neck	M	Design 5 - Graffiti	V-neck
9	12/9/2023	Michaela Ho	5	S	Design 2 - Sunset	V-neck	S	Design 1 - Waves	V-neck
10	12/10/2023	Chastity Smith	2	XXL	Design 2 - Sunset	V-neck	S	Design 1 - Waves	Sweatshirt
11	12/10/2023	Gabriel Medina	1	L	Design 1 - Waves	V-neck			
12	12/11/2023	Gabriel Medina	2	M	Design 2 - Sunset	Long-sleeved	L	Design 1 - Waves	Sweatshirt
13	12/12/2023	Stephanie Gilmore	1	M	Design 2 - Sunset	Crewneck	M	Design 4 - Skyline	Tank Top
14	12/12/2023	Online	3	S	Design 2 - Sunset	V-neck	M	Design 4 - Skyline	V-neck
15	12/13/2023	Online	1	XS	Design 1 - Waves	Crewneck	M	Design 4 - Skyline	Midtown
16	12/14/2023	Michaela Ho	2	XL	Design 5 - Graffiti	V-neck	XL	Design 4 - Skyline	Long-sleeved
17	12/14/2023	Gabriel Medina	1	XL	Design 1 - Waves	Tank Top	M	Design 4 - Skyline	Long-sleeved
18	12/16/2023	Michaela Ho	1	L	Design 3 - Boat	Crewneck	M	Design 4 - Skyline	Long-sleeved

Immagine 9: Interfaccia di Google Sheets_G

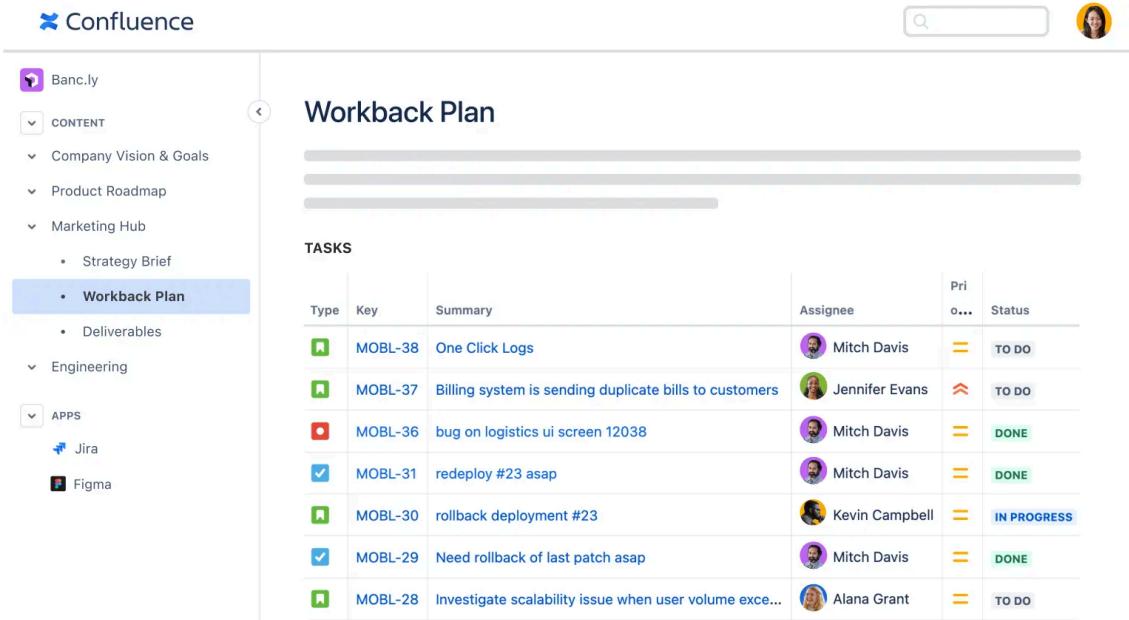
Fonte: <https://support.google.com/meet/answer/10550593?hl=it>

Confluence

Confluence è una piattaforma di documentazione che permette di creare, organizzare e condividere documenti in modo collaborativo. Possiede un registro delle modifiche aggiornato automaticamente, in modo da tracciare precisamente i cambiamenti apportati ai documenti.

Lo strumento è utilizzato dall'azienda per la documentazione dei processi e delle attività svolte, e per la condivisione di documenti e analisi.

Questa piattaforma è stata per me la principale fonte di informazioni in merito al prodotto fino a quel momento sviluppato, e mi ha permesso di avere una visione di insieme delle funzionalità implementate e delle esigenze del cliente.



The screenshot shows the Confluence interface with a sidebar on the left containing navigation links like 'Banc.ly', 'CONTENT' (with 'Company Vision & Goals', 'Product Roadmap', 'Marketing Hub' expanded), 'Engineering' (selected), and 'APPS' (with 'Jira' and 'Figma'). The main area is titled 'Workback Plan' and displays a 'TASKS' table. The table has columns for Type, Key, Summary, Assignee, Priority (Pri), and Status. The tasks listed are:

Type	Key	Summary	Assignee	Pri	Status
BUG	MOBL-38	One Click Logs	Mitch Davis	=	TO DO
BUG	MOBL-37	Billing system is sending duplicate bills to customers	Jennifer Evans	△	TO DO
BUG	MOBL-36	bug on logistics ui screen 12038	Mitch Davis	=	DONE
TASK	MOBL-31	redeploy #23 asap	Mitch Davis	=	DONE
BUG	MOBL-30	rollback deployment #23	Kevin Campbell	=	IN PROGRESS
TASK	MOBL-29	Need rollback of last patch asap	Mitch Davis	=	DONE
BUG	MOBL-28	Investigate scalability issue when user volume exce...	Alana Grant	=	TO DO

Immagine 10: Interfaccia di *Confluence*

Fonte: <https://www.atlassian.com/software/confluence>

1.4.5.5 Strumenti di sviluppo

Bitbucket_G

Bitbucket_G è uno strumento di controllo di versione utilizzato dall'azienda per la gestione del codice sorgente. Lo strumento permette di creare *repository_G* in cui caricare la *codebase_G*, e di gestire i diversi *branch_G* di sviluppo affinchè l'avanzamento dei lavori possa avvenire in modo parallelo, coordinato e collaborativo.

Grazie all'integrazione con Jira_G, Bitbucket_G permette di collegare i *task_G* presenti nella *board* con i *branch_G* di sviluppo, in modo da garantire che ogni *task_G* sia associato al *branch_G* corrispondente.

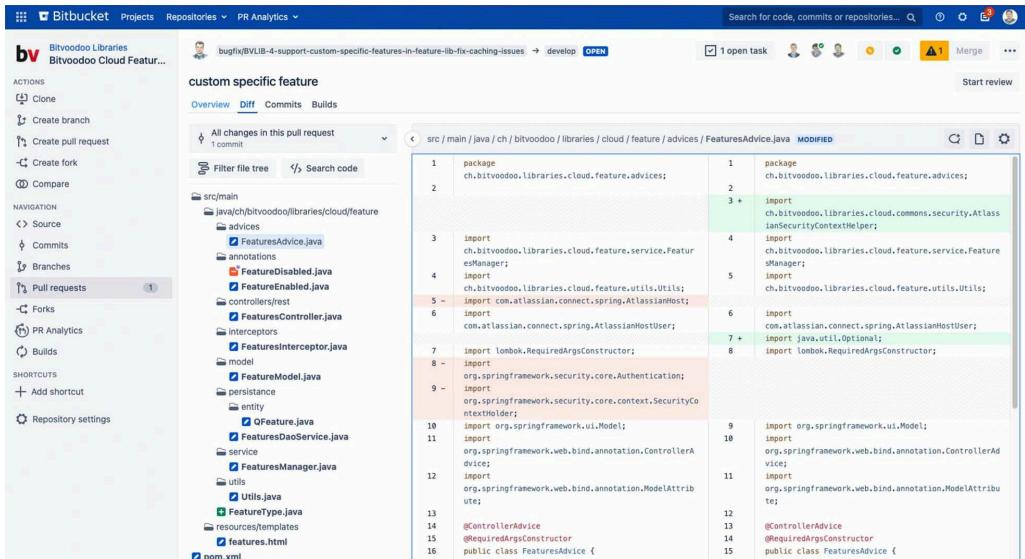


Immagine 11: Interfaccia di Bitbucket_G

Fonte: <https://www.atlassian.com/software/bitbucket>_G

Visual Studio Code

Visual Studio Code (o *VSCode*) è un ambiente di sviluppo integrato (IDE) utilizzato per la scrittura del codice sorgente. Lo strumento supporta diversi linguaggi di programmazione, e permette di eseguire *debugging* e *testing*_G del codice.

Le numerose estensioni disponibili, rendono questo strumento estremamente versatile e adattabile alle diverse esigenze di sviluppo.

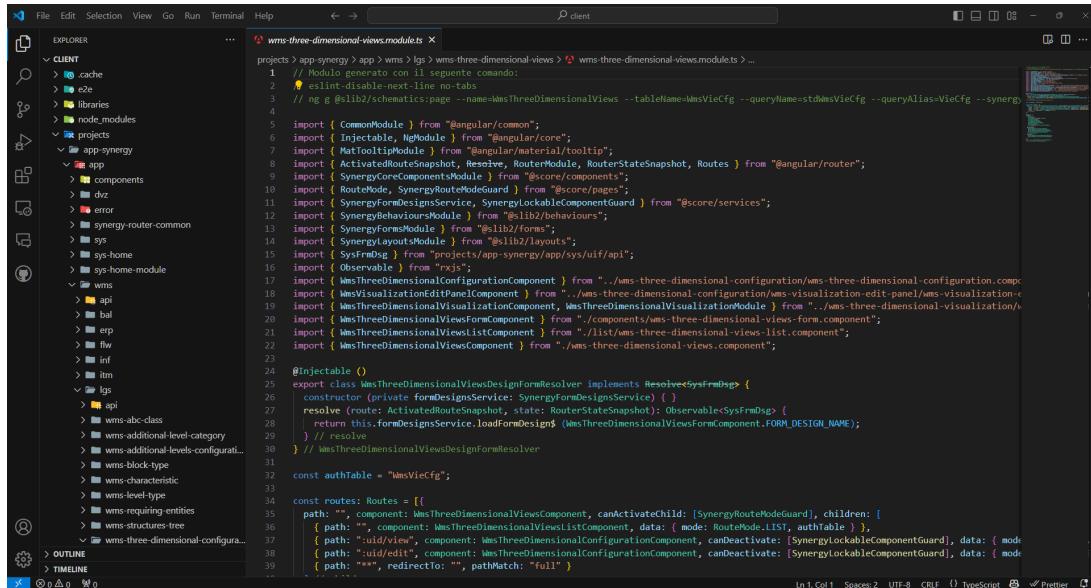
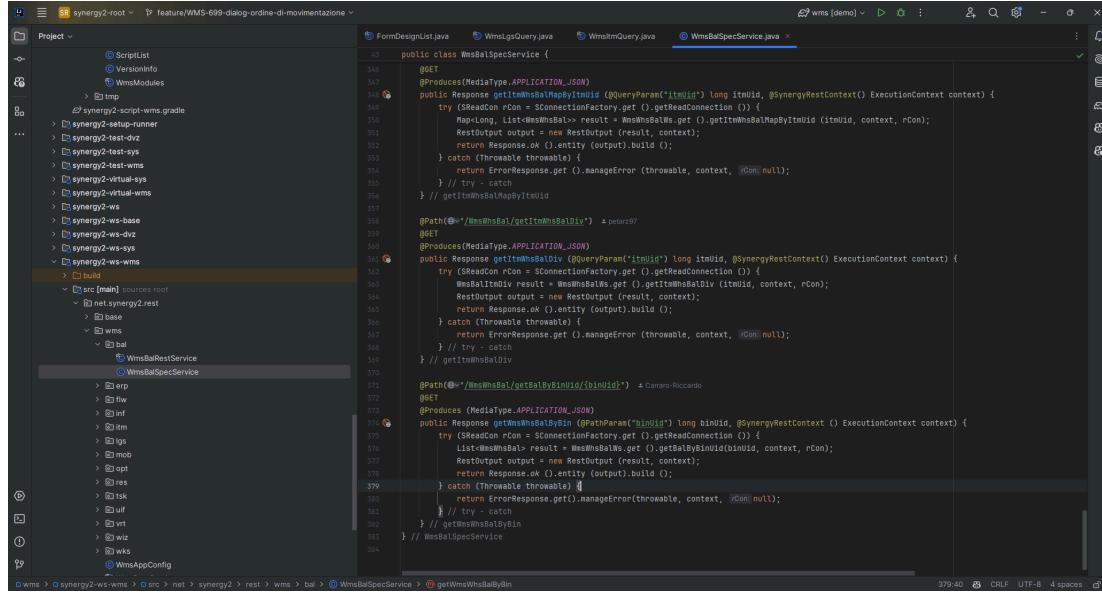


Immagine 12: Interfaccia di *VSCode* con il codice *frontend* del prodotto del tirocinio

IntelliJ

IntelliJ è un altro ambiente di sviluppo integrato (IDE) utilizzato dall'azienda per la scrittura del codice sorgente. Data la sua migliore integrazione con *gradle* e *tomcat*, il suo utilizzo semplifica lo sviluppo del codice *backend* realizzato in Java.



The screenshot shows the IntelliJ IDEA interface. The left sidebar displays the project structure for 'synergy2-root'. The code editor on the right contains the 'WmsBalSpecService.java' file, which includes several methods for handling requests related to 'WmsBal' and 'WmsBalByItmId'. The status bar at the bottom indicates the code length is 379.40 and the encoding is UTF-8.

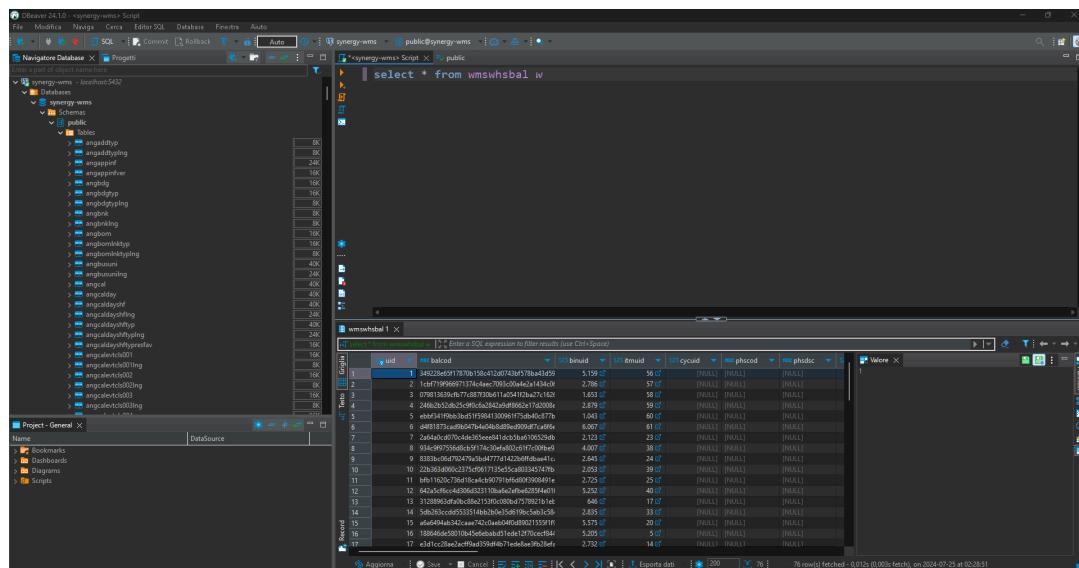
```
public class WmsBalSpecService {  
    @GET  
    @Produces(MediaType.APPLICATION_JSON)  
    public Response getWmsBalMapByItmid (@QueryParam("itmUid") long itmUid, @SynergyRestContext() ExecutionContext context) {  
        try (Sseadon rCon = SseadonFactory.get ().getReadConnection ()) {  
            Map<Long, List<WmsBal>> result = WmsBalRm.get ().getItmBalMapByItmid (itmUid, context, rCon);  
            RestOutput output = new RestOutput (result, context);  
            return Response.ok (ObjectUtils.entity (output).build ());  
        } catch (Throwable throwable) {  
            return ErrorResponse.get ().manageError (throwable, context, [COS] null);  
        } // try - catch  
    } // getWmsBalMapByItmid  
  
    @Path ("/{wmsBalBalByItmId}")  
    @GET  
    @Produces(MediaType.APPLICATION_JSON)  
    public Response getWmsBalByIdx (@QueryParam("itmUid") long itmUid, @SynergyRestContext() ExecutionContext context) {  
        try (Sseadon rCon = SseadonFactory.get ().getReadConnection ()) {  
            WmsBalIdx result = WmsBalRm.get ().getItmBalIdx (itmUid, context, rCon);  
            RestOutput output = new RestOutput (result, context);  
            return Response.ok (ObjectUtils.entity (output).build ());  
        } catch (Throwable throwable) {  
            return ErrorResponse.get ().manageError (throwable, context, [COS] null);  
        } // try - catch  
    } // getWmsBalIdx  
  
    @Path ("/{wmsBalBalByBinId}")  
    @GET  
    @Produces(MediaType.APPLICATION_JSON)  
    public Response getWmsBalByBinId (@PathParam("binUid") long binUid, @SynergyRestContext() ExecutionContext context) {  
        try (Sseadon rCon = SseadonFactory.get ().getReadConnection ()) {  
            List<WmsBal> result = WmsBalRm.get ().getBalByBinUid (binUid, context, rCon);  
            RestOutput output = new RestOutput (result, context);  
            return Response.ok (ObjectUtils.entity (output).build ());  
        } catch (Throwable throwable) {  
            return ErrorResponse.get ().manageError (throwable, context, [COS] null);  
        } // try - catch  
    } // getWmsBalByBinId  
}
```

Immagine 13: Interfaccia di *IntelliJ* con il codice *backend* del prodotto del tirocinio

DBeaver

DBeaver è uno strumento di amministrazione di *database_G* relazionali utilizzato dall'azienda per la gestione del *database_G* del prodotto.

La sua peculiarità è la semplicità di utilizzo, che permette, anche senza eseguire query, di visualizzare e modificare i dati presenti nel *database_G*, semplificando il processo di verifica.



The screenshot shows the DBeaver interface. The left sidebar displays the database structure for 'synergy-wms'. The central area shows a query results table titled 'wmsbal1' with columns: id, uid, bin_id, bin_uid, itmid, cycuid, phscod, phsdic. The table contains 17 rows of data. The bottom status bar indicates 78 rows fetched in 0.012s (0.003s fetch), dated 2024-07-25 at 02:28:51.

id	uid	bin_id	bin_uid	itmid	cycuid	phscod	phsdic
1	1	1	1	51.19	56	[NULL]	[NULL]
2	2	1	2	3.786	57	[NULL]	[NULL]
3	3	1	3	1.653	58	[NULL]	[NULL]
4	4	2	4	2.879	59	[NULL]	[NULL]
5	5	3	5	1.043	60	[NULL]	[NULL]
6	6	4	6	4.557	61	[NULL]	[NULL]
7	7	5	7	2.123	62	[NULL]	[NULL]
8	8	6	8	4.007	63	[NULL]	[NULL]
9	9	7	9	2.645	64	[NULL]	[NULL]
10	10	8	10	3.559	65	[NULL]	[NULL]
11	11	9	11	2.723	66	[NULL]	[NULL]
12	12	10	12	5.252	67	[NULL]	[NULL]
13	13	11	13	4.46	68	[NULL]	[NULL]
14	14	12	14	2.859	69	[NULL]	[NULL]
15	15	13	15	5.575	70	[NULL]	[NULL]
16	16	14	16	5.205	71	[NULL]	[NULL]
17	17	15	17	2.732	72	[NULL]	[NULL]

Immagine 14: Interfaccia di DBeaver con il *database_G* del prodotto del tirocinio

Postman

Postman è uno strumento di sviluppo di API_G utilizzato dall'azienda per testare e documentare le API_G del prodotto. Lo strumento permette di creare delle *request* al *server* dell'applicativo, e di visualizzare la risposta in modo chiaro e dettagliato.

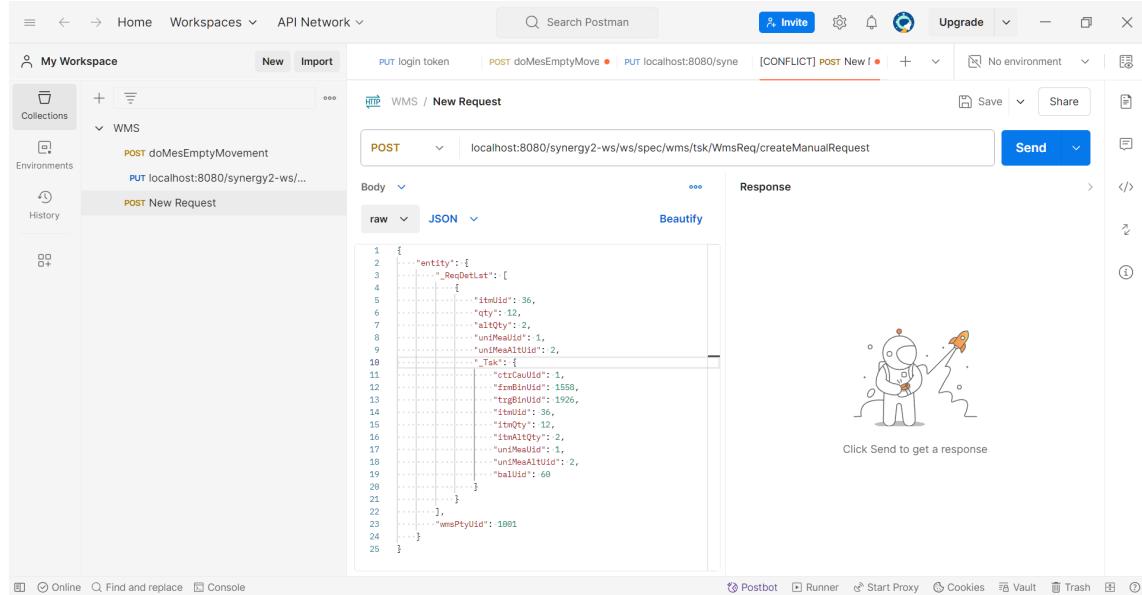


Immagine 15: Esempio di chiamata POST ad un servizio REST con Postman

1.4.5.6 Integrazione degli strumenti

Ecco una rappresentazione grafica di come gli strumenti sopra descritti siano integrati tra loro nello sviluppo del prodotto *software*:

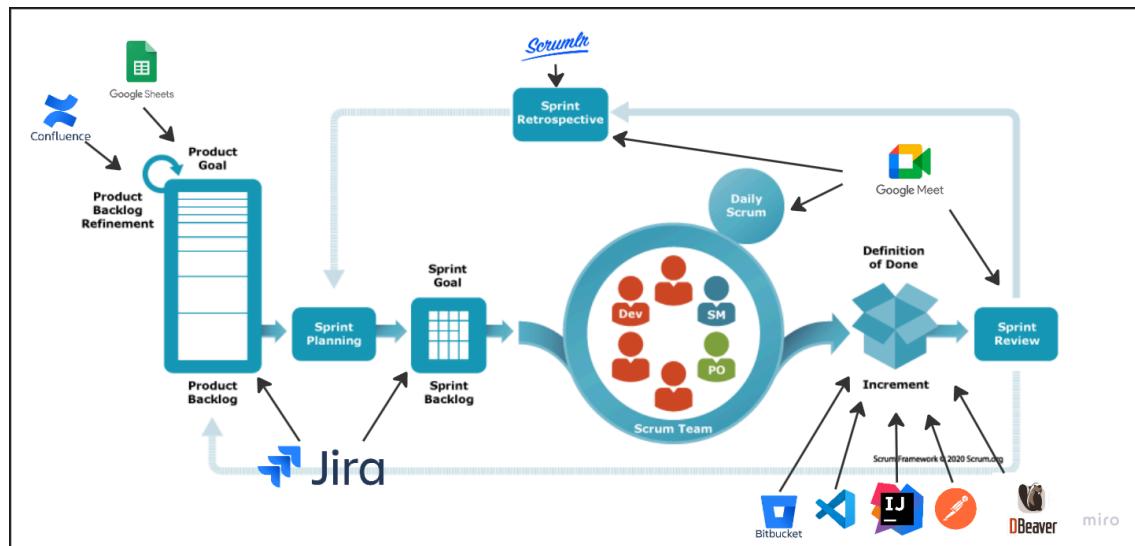


Immagine 16: Come gli strumenti si integrano nel modello di sviluppo aziendale

1.4.5.7 Gestione delle risorse umane

Il processo di gestione delle risorse umane si occupa di definire le competenze necessarie per lo sviluppo del prodotto, di assegnare i compiti ai membri del *team* e di garantire che le risorse siano utilizzate in modo efficace ed efficiente.

Nello svolgimento del mio percorso ho avuto la possibilità di comprendere come questo processo sia istanziato dall'azienda, e l'importanza che riveste la formazione e la crescita professionale dei membri del *team*.

Le prime due settimane del mio tirocinio sono state dedicate alla formazione, mediante lo svolgimento di lezioni frontali e di esercitazioni pratiche, permettendomi di apprendere le basi del *framework Synergy* mediante un approccio *learn by doing*. Inoltre ho avuto modo di constatare come la formazione sia un processo continuo che anche per i membri del *team* a cui sono stato affiancato, i quali svolgono regolarmente corsi offerti dall'azienda nella piattaforma Udemy.

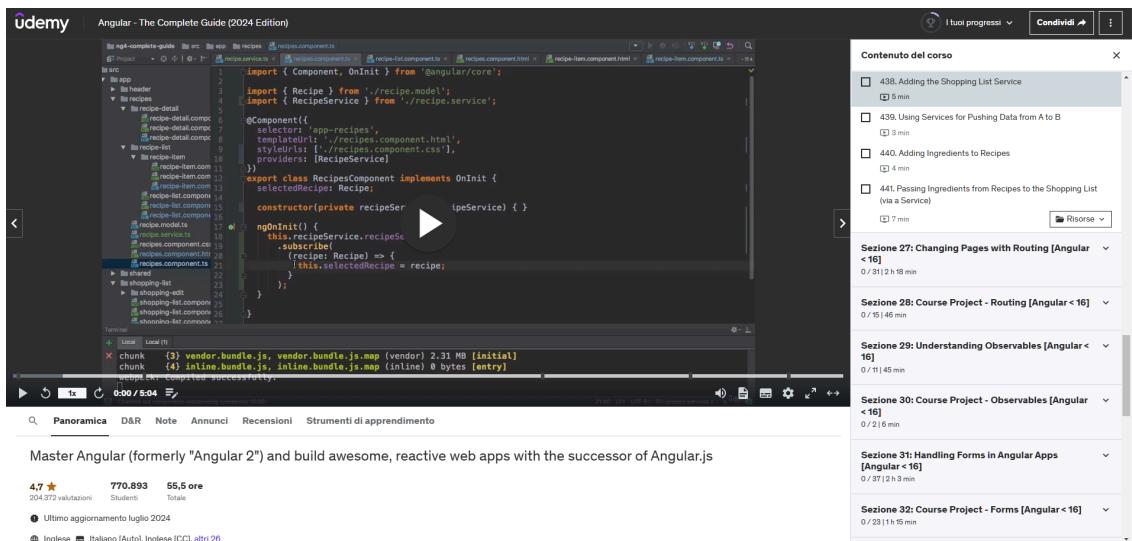


Immagine 17: Corso di formazione Angular su Udemy

Fonte: <https://www.udemy.com/course/the-complete-guide-to-angular-2/>

Come mostro nell'[immagine 17](#) Udemy, è una piattaforma di formazione *online* che permette di accedere a corsi di formazione in diversi argomenti, offrendo videolezioni e materiale didattico, permettendo di apprendere in modo autonomo e flessibile.

L'azienda stessa incentiva la formazione continua dei propri dipendenti, ritenuta fondamentale per perseguire gli obiettivi di innovazione e di crescita.

1.5 Il ruolo dell'innovazione

Un elemento distintivo della strategia aziendale di Sanmarco Informatica è l'importanza attribuita all'innovazione, come testimoniato dall'investimento annuale

di una quota significativa del fatturato, tra il 15% e il 20%, in Ricerca e Sviluppo. Questo impegno garantisce l'aggiornamento continuo dei prodotti e dei servizi, assicurando che rimangano allineati con le ultime tendenze tecnologiche.

La formazione continua dei dipendenti è un altro pilastro della filosofia aziendale. Come ho spiegato nel [paragrafo 1.4.5.7](#), Sanmarco Informatica offre costantemente corsi di formazione su nuove tecnologie e strumenti, avvalendosi di esperti interni e consulenti esterni, e utilizzando piattaforme di *e-learning* come Udemy. Questo investimento in competenze garantisce che il personale sia sempre aggiornato e in grado di affrontare le sfide tecnologiche future.

L'azienda inoltre promuove la partecipazione a conferenze e seminari come ad esempio l'evento "I nuovi paradigmi innovativi della Pianificazione su Commessa" tenutosi il 17 luglio 2024, o ancora il seminario "Intelligenza Artificiale al Servizio del *Business*" organizzato in collaborazione con IBM, *partner* storico dell'azienda.

Inoltre, data l'enorme risonanza che l'intelligenza artificiale sta avendo attualmente nel mondo dell'informatica, l'azienda ha in programma la definizione di un nuovo *team* dedicato, in modo da poter sfruttare appieno le potenzialità di questa nuova tecnologia su cui tante aspettative sono riposte.

2 Il tirocinio

2.1 Il ruolo dello stage per Sanmarco Informatica

Sanmarco Informatica attribuisce allo *stage* un ruolo fondamentale nel suo processo di crescita: come descritto nel [paragrafo 1.5](#), per perseguire gli obiettivi di innovazione e di crescita, l'azienda investe in formazione continua, e lo *stage*, è un'occasione per far crescere nuovi talenti e per portare nuove idee e competenze all'interno dell'azienda.

Durante il mio percorso, sono stato inserito in un *team* collaborativo e sempre presente, creando di fatto un ambiente accogliente e inclusivo, dove la figura dello stagista non era posta in secondo piano, ma anzi, era vista come un'opportunità per l'azienda stessa di crescere e innovare. Nelle due prime settimane, dedicate alla formazione, ho avuto modo di conoscere anche altri tirocinanti, alcuni provenienti come me dall'Università di Padova, altri da contesti lavorativi o universitari differenti, ulteriore sinonimo di come l'azienda investa nella formazione e nell'acquisizione di nuove risorse.

L'esperienza di *stage* infatti, rappresenta un'opportunità per gli studenti di mettere in pratica le conoscenze apprese durante il percorso di studi in un contesto aziendale, e allo stesso tempo, per le aziende, rappresenta un'occasione per conoscere nuovi talenti e per valutare la possibilità di inserirli nel *team* in modo permanente.

2.2 Il progetto proposto

2.2.1 Descrizione del progetto

Il progetto proposto consisteva nell'estensione delle funzionalità del prodotto WMS (*Warehouse Management System*) sviluppato dall'azienda, un applicativo volto alla gestione della logistica interna di un'azienda, monitorando l'utilizzo di *asset* e risorse, e ottimizzando operazioni di *handling* e movimentazione.

Nello specifico veniva richiesta l'implementazione di un ambiente tridimensionale in grado di rappresentare lo stato del magazzino, con la possibilità di interrogare i saldi presenti ed individuarne la collocazione.

A tale funzionalità si aggiungeva la possibilità di creare ordini di movimentazione della merce mediante un'operazione di *drag & drop* sull'interfaccia, semplificando il processo di creazione degli ordini e rendendolo più intuitivo e veloce.



Immagine 18: Come le funzionalità sviluppate nel tirocinio si integrano tra loro nel prodotto WMS

Nell'[immagine 18](#), mostro come le funzionalità sviluppate nel mio tirocinio si dovessero integrare tra loro, partendo dalla visualizzazione dello stato del magazzino, passando per la creazione degli ordini di movimentazione, fino alla gestione della loro presa in carico.

Il progetto di *stage* pertanto, non trattandosi di un'implementazione da zero, ma di un'estensione di un prodotto già esistente, mi ha permesso di lavorare con un prodotto *software* più complesso e strutturato, e, in questo senso, di mettere mano ad un prodotto *software* di carattere professionale, con tutte le sfide e le opportunità che questo comporta.

2.2.2 Obiettivi

2.2.2.1 Obiettivi aziendali

Gli obiettivi del tirocinio di interesse aziendale sono individuabili nello sviluppo e miglioramento delle funzionalità del prodotto WMS, in modo da renderlo più competitivo sul mercato e rispondere alle esigenze del cliente.

Farò riferimento agli obiettivi aziendali (OA) secondo la seguente notazione:

OA - TI

dove:

- **T** è il tipo di obiettivo, distinto in:
 - **Obbligatori (OB)** : obiettivi primari, che devono essere necessariamente raggiunti per il completamento del tirocinio;
 - **Desiderabili (D)** : obiettivi non vincolanti o strettamente necessari, ma dal riconoscibile valore aggiunto;
 - **Opzionali (OP)** : obiettivi secondari, che possono essere raggiunti in caso di tempo e risorse disponibili.
- **I** è un numero intero positivo, identificativo dell'obiettivo.

Obiettivi aziendali obbligatori (OB)	
OA-OB1	Implementazione dell'ambiente tridimensionale per la visualizzazione dello stato del magazzino
OA-OB2	Implementazione della funzionalità di <i>drag & drop</i> per la creazione degli ordini di movimentazione
Obiettivi aziendali desiderabili (D)	
OA-D1	Gestione assegnazione e presa in carico degli ordini di movimentazione
OA-D2	Gestione esecuzione degli ordini di movimentazione
Obiettivi aziendali opzionali (OP)	
OA-OP1	Documentazione funzionalità sviluppate

Tabella 2: Obiettivi aziendali

2.2.2.2 Obiettivi personali

Gli obiettivi del tirocinio di interesse personale riguardano l'insieme di quegli aspetti che mi aspettavo di curare durante il tirocinio, in modo da crescere professionalmente e personalmente.

In particolare, prima di iniziare il tirocinio, questi sono gli aspetti che ho tenuto in particolare considerazione:

- **Teamworking:** migliorare le mie capacità di collaborazione e di comunicazione all'interno di un *team* di sviluppo. Durante il percorso di studi infatti, vi sono state poche occasioni di lavorare in gruppo, e in quelle situazioni, spesso il gruppo era formato da persone che già conoscevo. Il tirocinio invece, mi dava la possibilità di lavorare con persone sconosciute, con competenze e *background* diversi, e quindi, di mettere alla prova le mie capacità di collaborazione e di comunicazione.
- **Problem solving:** migliorare il mio approccio all'affrontare problemi complessi e di trovare soluzioni efficaci, rispettando vincoli imposti dal contesto aziendale.
- **Conoscenze tecniche:** acquisire nuove conoscenze e competenze in merito alle tecnologie utilizzate e richieste nel mondo del lavoro, dandomi la possibilità di mettere in pratica i concetti appresi durante il corso di studi.
- **Conoscenze metodologiche:** acquisire nuove conoscenze e competenze in merito alle metodologie di sviluppo *software* e dei processi aziendali, in modo da poter mettere in pratica i concetti appresi durante i corsi di “Ingegneria del

software" e "Metodologie e Tecnologie per lo sviluppo *software*" in un contesto aziendale.

- **Lavoro di qualità:** garantire la qualità del prodotto *software* sviluppato, rispettando le convenzioni aziendali e i processi di verifica e validazione. Mi avrebbe fatto molto piacere poter consegnare all'azienda un prodotto *software* di qualità, pronto per essere utilizzato e integrato nel prodotto esistente, come di fatto è avvenuto.
- **Panoramica del mondo del lavoro:** acquisire una visione più chiara del mondo del lavoro, delle dinamiche e delle esigenze aziendali.

Si è trattato della mia prima vera esperienza lavorativa e ho voluto sfruttarla al meglio per crescere professionalmente e personalmente.

Farò riferimento agli obiettivi personali (OP) secondo la seguente notazione:

OP_I

- I è un numero intero positivo, identificativo dell'obiettivo.

Alla luce degli aspetti sopra descritti, gli obiettivi personali che mi sono posto sono i seguenti:

Obiettivi personali (OP)	
OP1	Sviluppare competenze con strumenti di comunicazione e collaborazione aziendali come Google Meet e GitHub _G
OP2	Approfondire l'utilizzo di ITS _G in un contesto aziendale, come ad esempio Jira _G
OP3	Partecipare attivamente ai processi di sviluppo <i>software</i> in un contesto aziendale
OP4	Sviluppare competenze con <i>framework</i> ampiamente utilizzati come Angular
OP5	Sviluppare competenze con nuovi linguaggi di programmazione come Java e TypeScript
OP6	Sviluppare codice di qualità tale da essere utilizzabile dall'azienda al termine del mio percorso
OP7	Comprendere i ritmi e le dinamiche di un lavoro in questo settore

Tabella 3: Obiettivi personali

2.2.3 Vincoli

2.2.3.1 Vincoli temporali

I vincoli temporali rappresentano le tempistiche entro cui il progetto doveva essere completato. Il periodo di tirocinio prevedeva una durata massima di 320 ore, organizzate in 8 settimane, con un impegno di 40 ore settimanali, tradotte in 8 ore giornaliere.

2.2.3.2 Vincoli tecnologici

Trattandosi di un progetto che prevedeva l'estensione di un prodotto già esistente, i vincoli tecnologici erano rappresentati dalle tecnologie già utilizzate e presenti nel prodotto, in modo da garantire la compatibilità e l'integrazione delle nuove funzionalità con quelle già esistenti.

In particolare lo *stack* tecnologico utilizzato è il seguente:

- **Frontend:**

- **Angular:** *framework* per lo sviluppo di applicazioni *web* in *TypeScript*;
- **TypeScript:** sovrastruttura di *JavaScript* che permette di scrivere codice più robusto e manutenibile basato sul paradigma della programmazione ad oggetti;
- **HTML e CSS:** linguaggi di *markup* e di stile per la definizione dell'interfaccia *web*.

- **Backend:**

- **Java:** linguaggio di programmazione utilizzato per lo sviluppo del *backend* dell'applicativo (mediante il *framework* proprietario Synergy);
- **Tomcat:** *server web* per l'esecuzione di applicazioni *web* in Java.

- **Database_G:**

- **PostgreSQL:** *database_G* relazionale utilizzato per la memorizzazione dei dati.

2.2.3.3 Vincoli organizzativi

L'organizzazione del periodo di tirocinio è stata fondamentale al fine di garantire un percorso valido e conforme alle aspettative, provvedendo ad un costante allineamento tra tutti gli attori coinvolti.

In questo senso, è stato fondamentale curare due aspetti chiave:

- **Comunicazione con il referente aziendale:** durante l'intero percorso il contatto con il referente aziendale doveva essere costante, in modo da monitorare l'avanzamento del progetto e garantire che gli obiettivi prefissati fossero rag-

giunti. Come discusso nel [paragrafo 1.4.1](#), lo svolgimento dei *Daily Standup Meeting* e delle *Sprint Review* e *Sprint Retrospective* erano fondamentali per garantire un allineamento costante tra le parti.

- **Comunicazione con il relatore:** il contatto con il relatore universitario doveva essere costante, in modo da essere allineato con l'andamento del tirocinio e sullo stato di avanzamento. A tal fine, ogni 5 giorni lavorativi, era necessario inviassi un resoconto delle attività svolte, degli obiettivi raggiunti e quanto pianificato per il periodo successivo.

2.3 Motivazione della scelta

Durante l'evento *StageIT* tenutosi in data 8 aprile 2024, ho avuto modo di conoscere diverse aziende e di valutare le opportunità di *stage* offerte. Nel valutare l'azienda presso cui svolgere il tirocinio ho tenuto in considerazione diversi aspetti, dalla presentazione dell'azienda, allo *stack* tecnologico utilizzato, e le conseguenti possibilità di crescita professionale e personale.

Le ragioni per cui ho scelto di svolgere il tirocinio presso Sanmarco Informatica sono molteplici e coprono i diversi aspetti che ho ritenuto fondamentali anche in relazione agli obiettivi personali che mi ero posto:

- **L'azienda:** durante il progetto del corso di *Ingegneria del software* mi ero già relazionato con l'azienda, la quale si era dimostrata e disponibile durante tutto il percorso. Questo aspetto per me era fondamentale: in un'esperienza formativa come lo *stage*, necessaria al completamento del percorso di studi, era importante per me poter contare su un rapporto costante e collaborativo con l'azienda.
- **Il contesto aziendale:** nella mia precedente esperienza con l'azienda nella realizzazione del progetto di *Ingegneria del software*, avevo avuto modo di realizzare un prodotto dalle finalità simili, ma in un ambiente molto diverso, di carattere accademico. Svolgere il tirocinio presso Sanmarco Informatica, mi dava la possibilità di vedere come invece un progetto del genere venisse sviluppato in un contesto aziendale, con tutte le sfide e le opportunità che questo comporta. Ho ritenuto la possibilità di confrontare le due esperienze e i due approcci allo sviluppo, molto affascinante e formativa, dandomi la possibilità di vedere nel concreto le differenze tra le due tipologie di approccio.
- **Stack tecnologico:** lo *stack* tecnologico utilizzato dall'azienda aveva suscitato il mio interesse. Ho lavorato con *framework* come Angular, estremamente diffuso e richiesto nel mondo del lavoro, e con tecnologie come Java e PostgreSQL, che mi avrebbero permesso di acquisire nuove competenze e di mettere in pratica i concetti appresi durante il corso di studi. Angular e Java infatti sono due

tecniche che già conoscevo, ma che non avevo mai utilizzato e approfondito, specialmente in un contesto professionale.

- **Tecnologie 3D:** un aspetto che aveva particolarmente colto la mia attenzione e che ho avuto modo di apprezzare, è stato lavorare con tecnologie 3D, in particolare modo Three.js. Si tratta di un campo diverso e peculiare, dove si vengono a creare anche ulteriori sfide come la gestione delle prestazioni e la rappresentazione di oggetti complessi. Questo aspetto mi ha dato la possibilità di mettermi alla prova e di apprendere nuove competenze in un campo stimolante e diverso dal classico sviluppo *web*.

Nella scelta non ho tenuto in particolare considerazione la posizione geografica delle aziende presso cui ho svolto i colloqui, in quanto il principale obiettivo era l'aspetto formativo e l'esperienza che avrei potuto acquisire. Nel caso specifico di Sanmarco Informatica, la sede dista tra i 30 e i 40 minuti in auto da dove risiedo.

Prima di iniziare il percorso di tirocinio, ho svolto due colloqui conoscitivi in sede con l'azienda, in presenza del *team* delle risorse umane e del referente aziendale, e solo a seguito del processo di selezione tenutosi nei giorni successivi, ho avuto modo di procedere con l'inizio del tirocinio.

3 Svolgimento del tirocinio

3.1 Pianificazione

L'organizzazione del tirocinio, secondo i vincoli temporali discussi nel [paragrafo 2.2.3.1](#), prevedeva una pianificazione delle attività atta a garantire il raggiungimento degli obiettivi prefissati.

Il mio percorso è suddiviso in quattro periodi di due settimane, ciascuno dedicato ad un aspetto specifico del piano di *stage*, in modo da garantire un'organizzazione efficace e una suddivisione chiara delle attività.

I periodi del tirocinio sono stati organizzati secondo la seguente tabella:

Periodo	Descrizione	Data inizio	Data fine
1	Formazione	20/05/2024	02/06/2024
2	Ambiente 3D	03/06/2024	16/06/2024
3	Funzionalità <i>drag & drop</i>	17/06/2024	30/06/2024
4	Validazione e documentazione	01/07/2024	14/07/2024

Tabella 4: Macrosuddivisione del tirocinio

Ciascuno dei periodi prevedeva lo svolgimento di attività specifiche, il cui tracciamento e monitoraggio avveniva mediante l'utilizzo di Jira_G.

Nel dettaglio, i quattro periodi del tirocinio sono stati organizzati come segue:

1) Formazione:

- **Formazione frontale *framework Synergy***: formazione sul *framework Synergy*, mediante lezioni frontali e esercitazioni pratiche. Questo periodo mi ha permesso di apprendere le basi del *framework* dell'azienda;
- **Visione video di formazione *frontend***: videolezioni registrate dell'azienda per approfondire le mie conoscenze su Angular, il *framework frontend* utilizzato dall'azienda;
- **Creazione e configurazione dell'ambiente di sviluppo**: configurazione dell'ambiente di sviluppo per poter iniziare a lavorare sul prodotto WMS;
- **Formazione frontale del prodotto WMS**: formazione frontale sul prodotto WMS, per comprendere meglio le funzionalità del prodotto e il contesto in cui mi sarei inserito.

Tali attività sono state organizzate come mostro nell'[immagine 19](#):

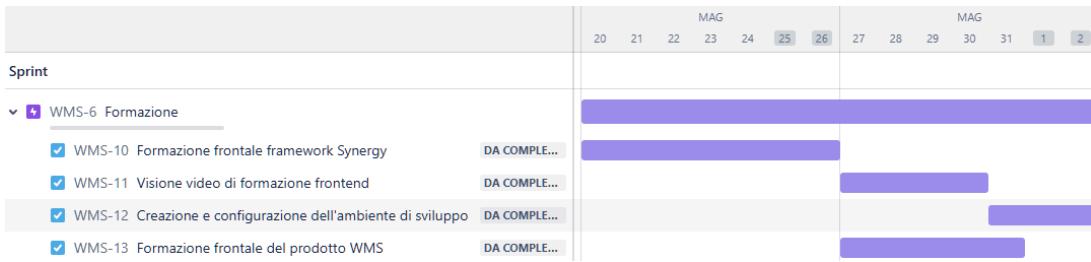


Immagine 19: Diagramma di Gantt delle attività del primo periodo

2) Ambiente 3D:

- Risoluzione di bug_G per approcciare il prodotto:** risoluzione di alcuni bug_G presenti nel prodotto al fine di approcciare gradualmente il *software* e comprendere meglio il contesto in cui mi sarei inserito;
- Analisi e studio di fattibilità per la ristrutturazione del codice:** analisi del codice esistente, anche con l'aiuto di colleghi, per capire come poter ristrutturare l'ambiente 3D;
- Implementazione delle classi di modello dell'ambiente 3D:** definizione e implementazione delle classi di modello necessarie per la visualizzazione dell'ambiente 3D, rivedendo la logica presente in una logica maggiormente strutturata e modulare;
- Integrazione dell'ambiente 3D nell'applicativo:** integrazione dell'ambiente 3D nell'applicativo esistente, assicurandomi che le funzionalità esistenti non venissero compromesse;
- Verifica corretta integrazione dell'ambiente 3D con le funzionalità esistenti:** verifica che l'ambiente 3D si integrasse correttamente con le funzionalità esistenti e con gli altri componenti del prodotto.

Tali attività sono state organizzate come mostro nell'[immagine 20](#):

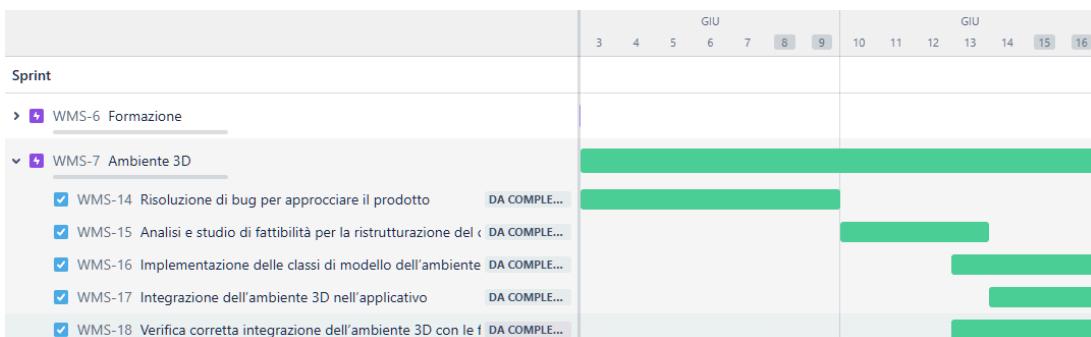


Immagine 20: Diagramma di Gantt delle attività del secondo periodo

3) Funzionalità *drag & drop*:

- **Implementazione della funzionalità di *drag & drop*:** implementazione della funzionalità di *drag & drop* per la creazione degli ordini di movimentazione, mediante il trascianamento di un *bin* (unità di contenimento) sull’interfaccia;
- **Implementazione del *dialog* di creazione dell’ordine di movimentazione:** implementazione del *dialog* per la creazione dell’ordine di movimentazione per la definizione dei saldi da movimentare, aperto al termine dell’operazione di *drag & drop*;
- **Analisi tabelle necessarie per la creazione dell’ordine di movimentazione:** analisi collettiva con i membri del *team* delle tabelle necessarie per la creazione dell’ordine di movimentazione;
- **Implementazione dei servizi REST per la creazione dell’ordine di movimentazione:** implementazione dei servizi REST per la creazione dell’ordine di movimentazione;
- **Verifica corretto funzionamento della funzionalità *drag & drop*:** verifica del corretto funzionamento della funzionalità di *drag & drop* e implementazione dei *test* necessari.

Tali attività sono state organizzate come mostro nell’[immagine 21](#):

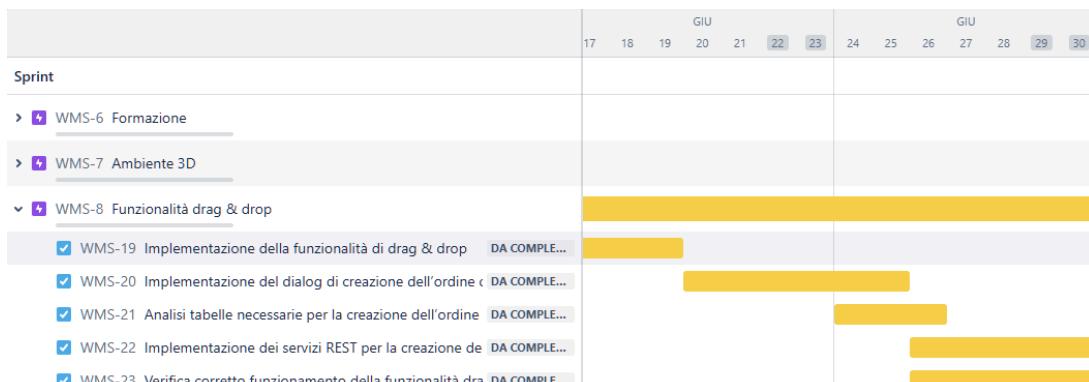


Immagine 21: Diagramma di Gantt delle attività del terzo periodo

4) Validazione e documentazione:

- **Presentazione finale del lavoro svolto:** presentazione finale del lavoro svolto durante il tirocinio al *team* e al referente aziendale;
- **Documentazione delle funzionalità sviluppate:** produzione della documentazione delle funzionalità sviluppate durante il tirocinio.

Tali attività sono state organizzate come mostro nell’[immagine 22](#):

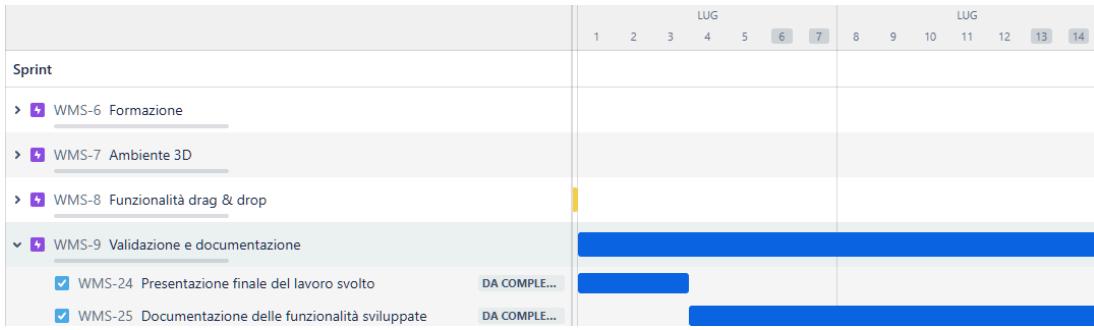


Immagine 22: Diagramma di Gantt delle attività del quarto periodo

Nel complesso, la pianificazione del tirocinio è la seguente:

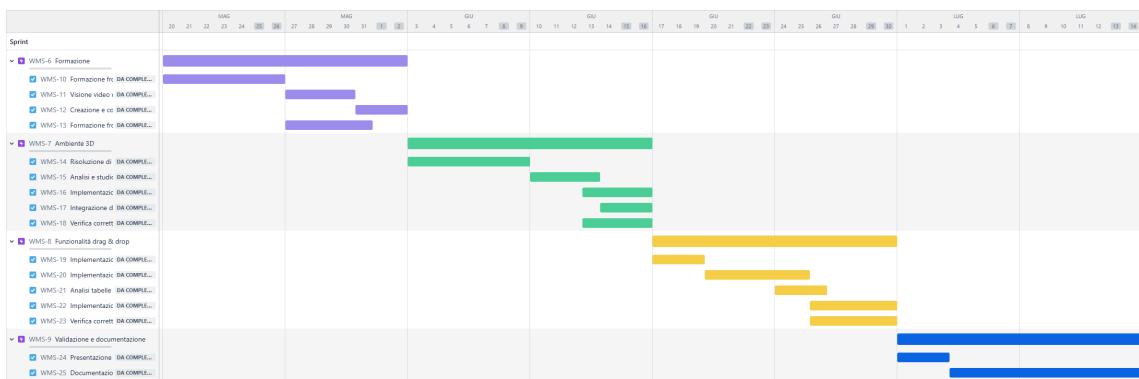


Immagine 23: Diagramma di Gantt complessivo delle attività svolte durante il tirocinio

3.2 Metodo di lavoro

3.2.1 Way of Working_G

Durante il corso di “Ingegneria del software” ho avuto modo di comprendere l’importanza di seguire una metodologia di sviluppo *software* strutturata e organizzata, e il tirocinio mi ha permesso di metterli in pratica, e vedere come questi venissero applicati in un contesto aziendale.

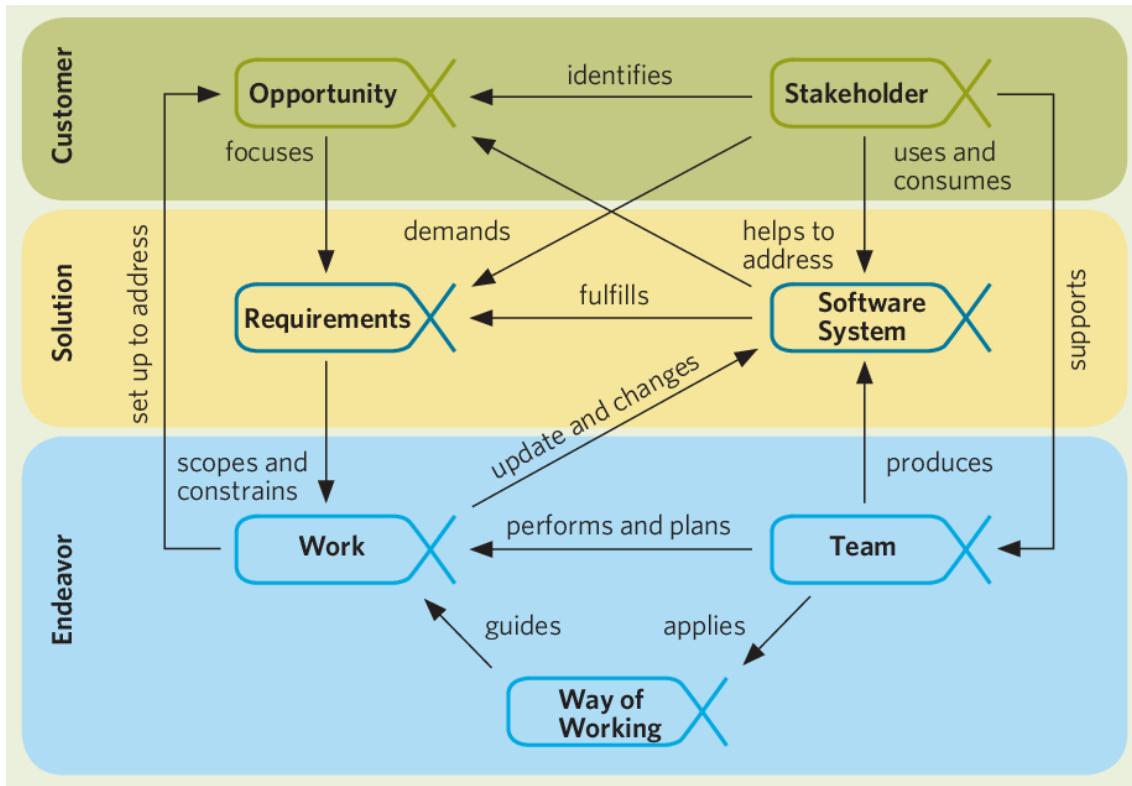


Immagine 24: L'importanza del *Way of Working*_G nel SEMAT

Fonte: <https://www.semanticscholar.org/paper/The-Essence-of-Software-Engineering%3A-The-SEMAT-Jacobson-Ng/ba4a3c5706ced64a2a71a230b30ba6ff5370ab6d>

Come descritto dal SEMAT (immagine 24), il *way of working*_G è fondamentale per garantire il successo di un progetto *software*, situato di fatto alla base di tutti gli aspetti del progetto.

Durante il mio tirocinio, mi sono impegnato a seguire un approccio strutturato e organizzato, che possedesse le seguenti caratteristiche:

- **Sistematico:** ho cercato di seguire un approccio sistematico e strutturato, organizzando le attività in modo da garantire un avanzamento costante e un monitoraggio efficace;
- **Disciplinato:** ho cercato di seguire le procedure e le convenzioni aziendali, rispettando le scadenze e gli impegni presi, e garantendo la qualità del prodotto *software* sviluppato;
- **Quantificabile:** ho cercato di quantificare le attività svolte, in modo da poter monitorare l'avanzamento del progetto e garantire il rispetto delle scadenze.

Come descritto nel paragrafo 1.4.1, l'azienda segue un modello di sviluppo *software Agile*_G, implementando nello specifico la metodologia *Scrum*_G. Questo approccio mi ha permesso di lavorare in modo organizzato e strutturato, garantendo un avanzamento costante e un monitoraggio efficace delle attività svolte.

In particolare, ho partecipato attivamente ai *Daily Standup Meeting* e alle *Sprint Review_G* e *Sprint_G Retrospective*, in modo da garantire un allineamento costante tra le parti e un monitoraggio efficace dell'avanzamento del progetto.

Trattandosi di un'esperienza dal carattere fortemente formativo, ho ritenuto inoltre opportuno prendere costantemente appunti, configurando una bacheca personale su Notion (piattaforma per la presa di note in modo strutturato) dove ho annotato le attività svolte, le problematiche riscontrate e le soluzioni adottate, in modo da garantire una traccia costante del mio percorso e un'analisi critica delle attività svolte, come mostrato nell'[immagine 25](#):

Titolo	Data	Partecipanti
Introduzione Synergy pt1	May 20, 2024	Samuele Baldan
Corso moodle sammarco synergy	May 22, 2024	
Model, Dao, Square	May 23, 2024	
Logica e servizi	May 24, 2024	
Riassunto Synergy	May 27, 2024	
Analisi WMS	May 28, 2024	
Glossario WMS	May 30, 2024	
Analisi WMS pt. 2	May 30, 2024	

Introduzione
Synergy è un fw sviluppato da SMI per la realizzazione di applicazioni web, con un'infrastruttura client-server, dove il client può comunicare attraverso la network con un unico server centrale.
Il framework supporta diverse tipologie di database, e la loro gestione è completamente trasparente al programmatore.
Utilizza tomcat 9 per esporre servizi rest|

Service layer
service -> API interfacciamento tra client e backend per operazioni CRUD;
Basato su tomcat 9.
spesso autogenerato direttamente dal framework (chiavi, etc) con API già pronte per le operazioni CRUD.
Nella parte service ci sono
• API rest esposte
• controllo di autenticazione

Immagine 25: B bacheca personale su Notion

3.2.2 Obiettivi di qualità

Ispirandomi ai principi dell'Ingegneria del *software*, ho cercato di garantire efficienza ed efficienza nel conseguimento dei miei obiettivi, seguendo un approccio di qualità e di conformità alle convenzioni aziendali.

In particolare, ho posto attenzione a due processi fondamentali, quali:

- **Verifica:** ho svolto attività di verifica costanti, mediante l'utilizzo di strumenti di analisi statica e dinamica del codice, e di test automatici e manuali, in modo da garantire la qualità del prodotto *software* sviluppato;
- **Validazione:** ho svolto attività di validazione costanti, mediante l'esecuzione di test di sistema_G e di accettazione (svolti dal *tester* del *team*), in modo da garantire che il prodotto realizzato fosse conforme alle aspettative e alle esigenze del cliente.

Attraverso l'applicazione rigorosa di questi processi, ho mirato a creare un prodotto che consideri questi aspetti cruciali:

- **Funzionalità:** il prodotto deve essere esaustivo nelle sue caratteristiche, preciso nel suo funzionamento e adattato al suo contesto d'uso;

-
- **Aderenza agli standard:** è essenziale che il prodotto rispetti le norme e le convenzioni aziendali, garantendo una coerenza e una uniformità nel codice e nelle funzionalità;
 - **Facilità d'uso:** l'interfaccia e le funzionalità devono essere intuitive e accessibili per gli utenti, minimizzando il rischio di errori;
 - **Flessibilità:** il *design* deve essere modulare, permettendo adattamenti e riutilizzi in base alle esigenze mutevoli dell'azienda;
 - **Durevolezza:** il prodotto deve dimostrarsi resistente nel tempo, con una struttura che faciliti eventuali interventi di manutenzione o riparazione;

L'obiettivo, come discusso nel [section 2.2.2](#), era di garantire un prodotto *software* di qualità, pronto per essere utilizzato e integrato nel prodotto esistente.

Il raggiungimento di questi obiettivi è stato perseguito mediante l'utilizzo di strumenti di verifica e validazione descritti nel [paragrafo 3.2.6](#).

3.2.3 Obiettivi di qualità di processo

Durante il mio tirocinio, ho cercato di garantire efficacia ed efficienza nel conseguimento dei miei obiettivi, seguendo un approccio di qualità e di conformità alle convenzioni aziendali, dando particolare rilevanza a due elementi chiave: l'efficacia e l'efficienza.

- **Efficacia:** rappresenta il primo cardine di questa metodologia. Essa si traduce nella capacità del prodotto di soddisfare pienamente le esigenze e le aspettative dell'azienda. Ogni componente sviluppato viene sottoposto a un rigoroso processo di convalida, garantendo così la sua conformità agli obiettivi prestabiliti e il suo effettivo contributo al valore complessivo del progetto.
- **Efficienza:** il secondo pilastro è costituito dall'efficienza del processo di sviluppo. Qui l'attenzione si concentra sull'ottimizzazione delle risorse disponibili, con l'obiettivo di contenere i costi mantenendo inalterati gli *standard* qualitativi del prodotto finale. Questo aspetto assume particolare rilevanza considerando i limiti temporali imposti al progetto. Si punta quindi a creare un flusso di lavoro agile_G e ben coordinato, capace di massimizzare i risultati entro le scadenze prefissate.

Il raggiungimento di questi obiettivi è stato possibile grazie alle diverse attività caratterizzanti il modello di sviluppo *Agile_G* e *Scrum_G*, come descritto nel [paragrafo 1.4.1](#).

In particolare, grazie alle *Sprint_G Review_G* e *Sprint_G Retrospective*, io e l'intero *team* di sviluppo, abbiamo avuto modo di valutare costantemente l'andamento del

progetto, individuando eventuali criticità e aree di miglioramento, e di adattare di conseguenza il nostro approccio di lavoro.

3.2.4 Interazione con il referente aziendale

Il rapporto con il referente aziendale è stato fondamentale per garantire il successo del mio tirocinio. Durante il percorso (svolto per quasi la totalità in presenza in sede), ho mantenuto un contatto costante, garantendo un allineamento tra le parti e un monitoraggio efficace dell'avanzamento del progetto.

Giornalmente ho partecipato ai *Daily Standup Meeting*, in cui ho condiviso con il *team* le attività svolte, le problematiche riscontrate e le soluzioni adottate: in questo modo, il referente aziendale ha potuto monitorare costantemente il mio percorso e fornirmi un *feedback* costante sulle attività svolte.

Quando il referente aziendale lavorava in modalità *smart working*, ho mantenuto il contatto con lui tramite gli strumenti di comunicazione aziendali come Google Meet e Google Chat descritti nel [paragrafo 1.4.5.3](#).

3.2.5 Revisioni di progresso

Le revisioni di progresso sono state fondamentali per garantire un monitoraggio costante dell'avanzamento e per ottenere *feedback* valido e tempestivo sulle attività svolte.

Come menzionato nel [paragrafo 1.4.1](#), ho avuto modo di partecipare attivamente a diverse attività di revisione_G, dalle giornaliere durante i *Daily Standup Meeting*, alle revisioni di fine Sprint_G (*Sprint_G Review_G* e *Sprint_G Retrospective*), fino alla revisione_G finale del lavoro svolto durante la presentazione conclusiva al *team* e al referente aziendale. Man mano che prendevo confidenza con le pratiche e con le tecnologie aziendali, ho potuto partecipare in modo sempre più attivo a queste attività, riuscendo a fare domande sempre più mirate e proporre soluzioni sempre più precise.

Queste attività mi hanno dato la possibilità di avere un rapporto attivo e partecipativo con il *team* con cui ho lavorato, permettendomi di insermi sia nel contesto lavorativo sia all'interno del *team*.

3.2.6 Strumenti di verifica

Gli strumenti utilizzati per la garantire *standard* elevati di qualità e di conformità alle convenzioni aziendali, in modo tale che il prodotto che ho realizzato fosse direttamente utilizzabile dall'azienda. In questo paragrafo mi riferisco a strumenti come:

- Test automatici: unità, integrazione e di sistema (attraverso il framework synergy e Mockito)

- Analisi statica del codice
- Analisi dinamica del codice (in merito soprattutto alle prestazioni visto il refactoring dell’ambiente tridimensionale)
- Controllo di versione
- Test manuali di validazione (eseguiti dal tester nel team).

Al fine di perseguire gli obiettivi di qualità indicati nel [paragrafo 3.2.2](#), ho utilizzato strumenti e tecnologie che rendessero i processi di verifica e validazione efficaci e conformi alle esigenze aziendali.

Come menzionato nel [paragrafo 1.4.3.2](#), l’azienda opera con un processo di *continuous integration*_G e *continuous deployment*, garantendo un monitoraggio costante del codice e delle funzionalità sviluppate, al fine di accettare all’interno del *repository*_G un prodotto sempre funzionante e conforme alle aspettative.

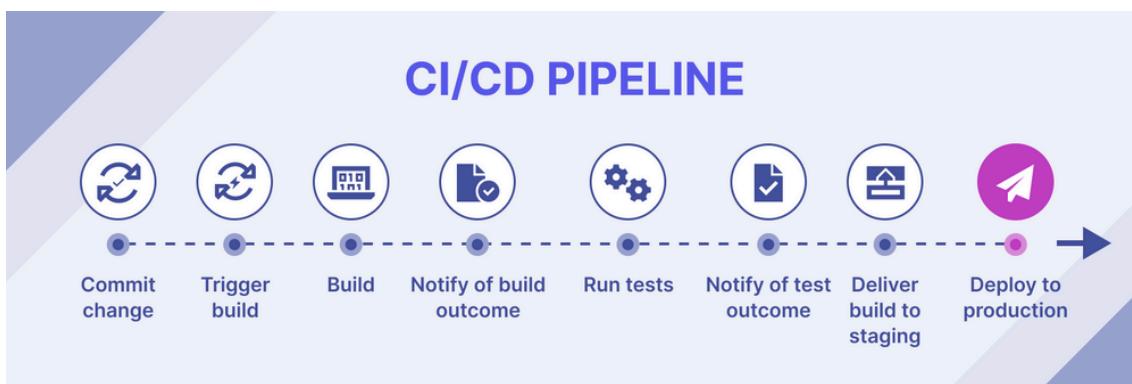


Immagine 26: Pipeline di *Continuous Integration*_G e *Continuous Deployment*

Fonte: <https://katalon.com/resources-center/blog/ci-cd-pipeline>

Come mostro nell’[immagine 26](#), i *test* rappresentano un aspetto fondamentale di questo processo: alla creazione di una *pull request*_G, viene eseguita la *build* del progetto e successivamente vengono eseguiti i *test* automatici. Solo qualora tutta la *pipeline* venga superata con successo, allora la *pull request*_G viene accettata e il codice viene integrato nel *repository*_G. In caso contrario, la *pull request*_G viene respinta e il lavoro deve essere rivisto e corretto.

Gli strumenti e le tecnologie che ho utilizzato per garantire la qualità del prodotto sono le seguenti:

- **Test automatici:** questa tipologia di *test* viene eseguita in modo automatico. Durante il mio percorso ho implementato 3 principali tipologie di *test*:
 - **Test di unità:** *test* che verificano il corretto funzionamento di singole unità di codice, garantendo che ciascuna unità funzioni correttamente;
 - **Test di integrazione:** *test* che verificano il corretto funzionamento dell’integrazione tra le diverse unità di codice, garantendo che le unità funzionino correttamente anche quando integrate tra loro;

- **Test di sistema:** *test* che verificano il corretto funzionamento del sistema nel suo complesso, garantendo che tutte le funzionalità siano conformi alle aspettative.

Il *framework* Synergy predisponiva un ambiente di test completo, che mi ha permesso di implementare i *test* agevolmente e in modo conforme alle esigenze aziendali.

- **Analisi statica del codice:** ho utilizzato strumenti di analisi statica per verificare la qualità del codice prodotto in grado di evidenziare errori e *code smell*, permettendomi di produrre codice che rispettasse le convenzioni aziendali e fosse conforme alle aspettative. In particolare, ho utilizzato i seguenti *linter*:
 - **SonarLint:** *linter* per Javascript e TypeScript;
 - **IntelliJ IDEA:** *linter* integrato nell’IDE utilizzato per lo sviluppo.
- **Analisi dinamica del codice:** ho utilizzato strumenti di analisi dinamica per verificare le prestazioni del codice prodotto, garantendo che il prodotto fosse conforme alle aspettative e rispondesse ai requisiti di *performance* richiesti. Infatti, il mio tirocinio comprendeva la ristrutturazione del codice dell’ambiente tridimensionale, e quindi era fondamentale garantire che le prestazioni del prodotto fossero adeguate. In particolare, ho utilizzato i *DevTools* di Google Chrome, che mi hanno permesso di verificare il livello di carico del prodotto e di identificare eventuali criticità nel *rendering* dell’ambiente tridimensionale.
- **Controllo di versione:** come descritto nel [paragrafo 1.4.5.5](#), ho utilizzato Bit-Bucket come sistema di controllo di versione, garantendo un monitoraggio costante del codice e delle funzionalità sviluppate.

Come mostro nell’[immagine 27](#), la *pipeline* per l’accettazione di una *Pull Request*_G prevedeva una serie di passaggi, tra cui la *build* del progetto e l’esecuzione dei *test* automatici, garantendo che il codice prodotto fosse conforme alle aspettative e pronto per essere integrato nel *repository*_G.

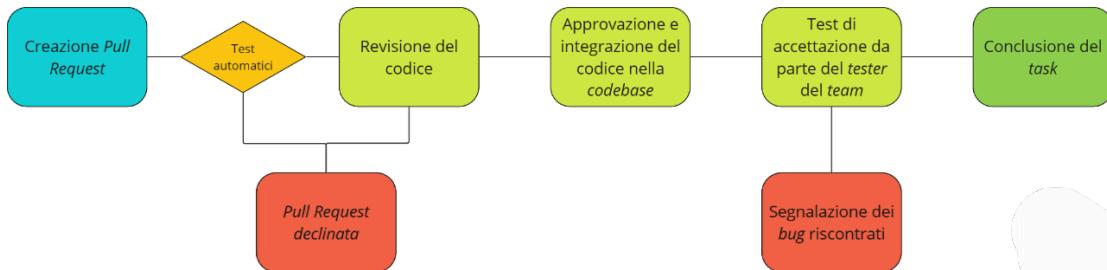


Immagine 27: *Pipeline* per l’accettazione di una *Pull Request*_G

Il tracciamento pertanto delle modifiche apportate e il versionamento_G del codice prodotto hanno rappresentato un aspetto fondamentale del mio percorso, avendo ulteriore conferma dei principi appresi durante il corso di “Ingegneria del software”.

Ogni attività veniva tracciata mediante un riferimento alla *issue_G* di Jira_G corrispondente, identificata da un codice univoco così strutturato:

WMS - XX

dove:

- **WMS**: identifica il progetto WMS;
- **XX**: identifica il numero progressivo della *issue_G*.

Ogni *task_G* ha il riferimento all’assegnatario, al tempo stimato per il completamento e al *branch_G* di riferimento. Ad ogni *pull request_G* veniva associato invece il membro del *team* che avrebbe dovuto effettuare la revisione_G del codice.

In questo modo, l’intero sviluppo del progetto è stato tracciato e monitorato costantemente, avendo sempre la possibilità di comprendere in ogni momento “chi fa cosa” e “quando”, con un chiaro riferimento alle modifiche apportate al cambiamento di versione.

3.2.7 Resoconti

Il mio approccio a tenere documentata l’intera esperienza, mediante resoconti giornalieri relativi al lavoro svolto durante la giornata e i resoconti settimanali a lei inviati.

3.3 Analisi dei requisiti

3.3.1 Casi d’uso

Per rappresentare il comportamento delle funzionalità implementate ho utilizzato ne ho derivato i casi d’uso. Descrizione di cosa sono e come li ho utilizzati.

3.3.2 Tracciamento dei requisiti

Come sono stati identificati i requisiti e il mio approccio al loro soddisfacimento.

3.4 Progettazione

3.4.1 Tecnologie utilizzate

Panoramica dello stack tecnologico che ho utilizzato.

3.4.2 Progettazione dell'ambiente tridimensionale

Descrizione ad alto livello del refactor dell'ambiente tridimensionale che ho svolto.

3.4.3 Progettazione della funzionalità *drag & drop*

Descrizione ad alto livello della funzionalità di *drag & drop* per la creazione dell'ordine di movimentazione.

3.4.4 Architettura del sistema

Descrizione dell'architettura del sistema su cui ho lavorato, sia lato backend sia lato frontend, in modo da avere una comprensione maggiore dell'applicativo e del contesto in cui mi sono inserito.

3.4.5 Design pattern

I design pattern che ho utilizzato per garantire un prodotto software di qualità, spiegando le motivazioni delle scelte e i vantaggi nel loro utilizzo.

3.5 Codifica

3.5.1 Visualizzazione tridimensionale

3.5.1.1 Classi implementate

Descrizione delle classi implementate per la visualizzazione dell'ambiente 3D.

3.5.1.2 Cambiamenti apportati

Come ho svolto il *refactoring* dell'ambiente 3d, quali i componenti, i servizi, i cambiamenti apportati per adattarsi alla nuova logica implementata, inserendo anche immagini dell'interfaccia.

3.5.2 Drag & Drop e creazione ordini di movimentazione

3.5.2.1 Componenti

Descrizione dei componenti che ho implementato, con immagini dell'interfaccia.

3.5.2.2 Servizi

Descrizione dei servizi che ho implementato.

3.5.2.3 Servizi REST

Descrizione dei servizi REST che ho implementato.

3.6 Verifica e validazione

3.6.1 Test di unità_G

I test di unità_G che ho implementato per il lato backend, con il supporto del framework Synergy.

3.6.2 Test di integrazione_G

I test di integrazione_G che ho implementato per il lato backend, con il supporto del framework Synergy.

3.6.3 Test di *performance*

I test che ho svolto per verificare che le *performance* del prodotto realizzato rispecchiassero le aspettative. (particolare riferimento all'ambiente tridimensionale).

3.6.4 Test di sistema_G

I test di sistema_G che ho svolto per accertarmi del corretto funzionamento delle funzionalità implementate.

3.6.5 Test di accettazione_G

I test di accettazione_G svolti dal tester del_team_ che permettono di definire concluso il processo di sviluppo della funzionalità.

3.7 Risultati raggiunti

3.7.1 Il prodotto realizzato

Descrizione di quanto ho prodotto dal punto di vista dell'utente finale.

3.7.2 Copertura dei requisiti

Il livello di copertura dei requisiti individuati durante l'analisi.

3.7.3 Copertura di testing_G

Il livello di copertura di codice in relazione ai test implementati.

3.7.4 Materiali prodotti

Il livello complessivo dei materiali prodotti durante il tirocinio: oltre infatti al codice, durante il percorso di tirocinio ho tenuto traccia giornalmente della attività svolte mediante una bacheca su Notion. Inoltre mi sono preoccupato di redigere puntualmente la documentazione relativa a tutte le funzionalità prodotte e alla loro analisi, condividerla anche il referente aziendale. Questa sezione ri-

porterà una panoramica quantitativa di questi materiali prodotti, le linee di codice scritte, i *meeting* svolti e le issue_G su Jira_G svolte (*bugfix* e funzionalità).

4 Valutazione retrospettiva

4.1 Soddisfacimento degli obiettivi

4.1.1 Obiettivi aziendali

Descrizione del livello di soddisfacimento degli obiettivi aziendali indicati nel paragrafo 2.2.3.

4.1.2 Obiettivi personali

Descrizione del livello di soddisfacimento degli obiettivi personali indicati nel paragrafo 2.2.4.

4.2 Competenze acquisite

Analisi delle competenze tecniche e personali che ho sviluppato durante il percorso di tirocinio. Entrerò maggiormente nel dettaglio rispetto al paragrafo 4.1.2.

4.3 Valutazione personale

Valutazione personale dell'esperienza del tirocinio.

4.4 Università e mondo del lavoro

Valutazione del percorso universitario in relazione al mondo del lavoro.

Acronimi e abbreviazioni

A

API: Application Programming Interface.

B

BU: Business Unit.

O

OA: Obiettivi aziendali.

OA-OB: Obiettivi aziendali obbligatori.

OA-D: Obiettivi aziendali desiderabili.

OA-OP: Obiettivi aziendali opzionali.

OP: Obiettivi personali.

I

IDE: Integrated Development Environment.

ITS: Issue Tracking System.

U

UML: Unified Modeling Language.

Glossario

A

Agile

Metodologia di sviluppo software che prevede la realizzazione di progetti in modo iterativo e incrementale, con particolare attenzione alla collaborazione tra i membri del team e alla risposta rapida ai cambiamenti.

Application Programming Interface (API)

Insieme di regole e protocolli che consente la comunicazione standardizzata tra software distinti. Definisce le modalità di scambio e le strutture dati scambiate durante la comunicazione.

B

Backlog

Gruppo di attività da completare per conseguire un certo obiettivo.

Bitbucket

Software di versionamento distribuito utilizzabile tramite linea di comando.

Branch

Nel contesto di Git rappresenta un ramo di sviluppo, implementato come puntatore ad un commit. La suddivisione in branch permette di lavorare parallelamente a parti diverse dello stesso prodotto.

Bug

Anomalia che porta al malfunzionamento di un software, producendo un risultato inatteso o errato.

C

Codebase

Insieme di codice sorgente di un software.

Continuous Delivery (CI/CD)

Pratica di sviluppo software che prevede il deployment continuo in produzione delle modifiche al codice sorgente.

Continuous Integration (CI/CD)

Pratica di sviluppo software che prevede l'integrazione continua delle modifiche al codice sorgente da parte dei vari membri del team. L'obiettivo è quello di rilevare e risolvere i problemi di integrazione il prima possibile (feedback rapido).

D

Database (DB)

Insieme di informazioni (o dati) strutturate, in genere archiviate elettronicamente in un sistema informatico. Solitamente i database sono gestiti da un DBMS (Database Management System), il quale deve garantire efficacia, efficienza, privacy ed affidabilità.

Diagrammi di burndown

Un burndown chart è una rappresentazione grafica del lavoro da fare su un progetto nel tempo. Di solito il lavoro rimanente (o backlog) è indicato sull'asse verticale e il tempo sull'asse orizzontale. Il diagramma rappresenta una serie storica del lavoro da fare.

F

Feature

Specifico funzionalità o caratteristica del prodotto sviluppato. Può essere richiesta esplicitamente dal Proponente o individuata dal Fornitore sulla base del processo di Analisi dei Requisiti.

GitHub (GH)

Servizio di hosting di progetti software basato su Git utilizzato da singoli ed organizzazioni. Fornisce un Issue Tracking System integrato e permette di implementare sistemi di Continuous Integration e Continuous Delivery tramite automazioni chiamate Action.

Google Drive

Servizio di web cloud offerto da Google basato sull'archiviazione e condivisione di file e documenti all'interno di uno spazio condiviso e modificabile via web.

Google Sheets

Applicazione web-based che permette la creazione, aggiornamento e modifica di fogli di calcolo.

I

Ingegneria del Software (IS, SWE)

Disciplina che si occupa della progettazione, sviluppo, test e manutenzione del software. Adotta un approccio sistematico, disciplinato e quantificabile, che mira a creare software di alta qualità, affidabile e manutenibile.

Issue

Specifico problema riscontrato durante una delle fasi del ciclo di vita del prodotto sviluppato.

Issue Tracking System (ITS)

Sistema informatico che gestisce e mantiene elenchi di problemi riscontrati durante il ciclo di vita di un'applicazione.

J

Jira

Suite di software proprietari per l'ITS sviluppata da Atlassian, che consente il bug tracking e la gestione dei progetti sviluppati con metodologie agile.

P

Pull Request (PR)

Richiesta, rivolta all'autore originale di un software da parte di suoi collaboratori, di includere modifiche al suo progetto. Questa richiesta ha come oggetto due branch, sui quali, in caso di accettazione della PR, viene effettuato un merge (unione).

R

Repository

Archivio in cui vengono conservati tutti i file di un progetto, utilizzato dagli sviluppatori per apportare e gestire le modifiche al codice sorgente/documentazione del prodotto. Questa cartella può essere salvata localmente o ospitata su una piattaforma online (ad esempio GitHub).

Requisito

È una condizione o funzionalità che deve essere verificata o posseduta dal sistema o da un componente del sistema per soddisfare un contratto, uno standard, una specifica o qualsiasi altro documento formalmente specificato.

Review

Revisione di documenti o codice. È un processo di peer review utilizzato per individuare eventuali problemi e migliorare la qualità del prodotto. È un'attività fondamentale per garantire uno standard qualitativo al prodotto.

S

Scrum

Framework agile di gestione dello sviluppo di progetti complessi con l'obiettivo di consegnare il maggior valore business nel più breve tempo possibile.

Sprint

Uno sprint è un breve periodo di tempo (4 settimane per Sanmarco Informatica), in cui un team Scrum collabora per completare una determinata quantità di lavoro. Gli sprint rappresentano una parte essenziale delle metodologie Scrum e Agile.

T

Test di accettazione

Test formale eseguito per verificare se l'applicazione soddisfa i requisiti richiesti.

Test di integrazione

Attività di testing che verifica il corretto funzionamento dell'interazione o dell'interfaccia tra due o più moduli integrati.

Test di sistema (E2E)

Attività di testing che verifica il corretto funzionamento del sistema integrato nel suo complesso. Questa tipologia di test è anche detta "end-to-end".

Test di unità

Attività di testing che verifica il corretto funzionamento di una singola unità di un software. A seconda del paradigma di programmazione adottato, l'unità può essere un singolo metodo, una classe, un modulo o un componente.

Testing

Attività di verifica e validazione del software tramite il collaudo di partizioni di esso.

Ticket

Rappresenta il lavoro da completare a sostegno degli obiettivi più ampi, ogni ticket è individuale e atomico.

U

Unified Modeling Language (UML)

Linguaggio di modellazione visivo, destinato a fornire un modo standard per visualizzare la progettazione di un sistema.

V

Versionamento

Processo di assegnazione di identificatori univoci a diversi momenti di sviluppo di un software o progetto. Comprende il recupero di versioni diverse da quella attuale e l'aggiornamento della versione stessa.

W

Way of Working (WoW)

Rappresenta il modo di lavorare del gruppo. È la descrizione di tutte le tecnologie, attività, metodologie adottate dal gruppo e di come queste sono utilizzate.

Bibliografia e sitografia

- *Gestione di progetto*, slide del corso di Ingegneria del Software
<https://www.math.unipd.it/~tullio/IS-1/2023/Dispense/T4.pdf> (ultimo accesso 05/08/2024)

I

- *I processi di ciclo di vita del software*, slide del corso di Ingegneria del Software
<https://www.math.unipd.it/~tullio/IS-1/2023/Dispense/T2.pdf> (ultimo accesso 25/07/2024)

M

- *Modelli di sviluppo software*, slide del corso di Ingegneria del Software
<https://www.math.unipd.it/~tullio/IS-1/2023/Dispense/T3.pdf> (ultimo accesso 25/07/2024)

Q

- *Qualità del Software*, slide del corso di Ingegneria del Software
<https://www.math.unipd.it/~tullio/IS-1/2023/Dispense/T7.pdf> (ultimo accesso 06/08/2024)

S

- Sito web dell'azienda Sanmarco Informatica
<https://www.sanmarcoinformatica.it/> (ultimo accesso 25/07/2024)
- Sito web ufficiale di Scrum
<https://www.scrum.org/resources/what-scrum-module> (ultimo accesso 25/07/2024)
- Sito web ufficiale di SEMAT
<https://semat.org/> (ultimo accesso 25/07/2024)