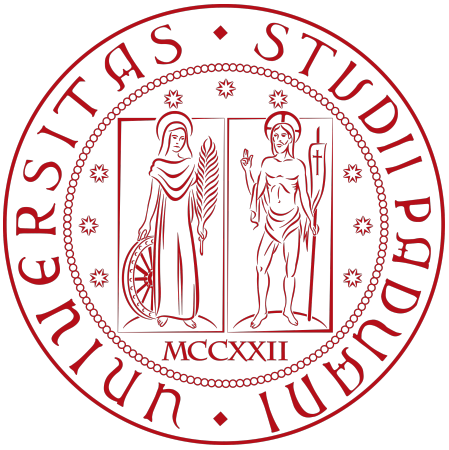

UNIVERSITÀ DEGLI STUDI DI PADOVA

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'ambeitente 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**: 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 correlata da una didascalia e la fonte da cui è stata tratta;
- i termini riportati nel glossario sono corredati da 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

Indice delle immagini

Indice delle tabelle

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 organizzazta 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*$_G$ 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.

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 familiartià 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.

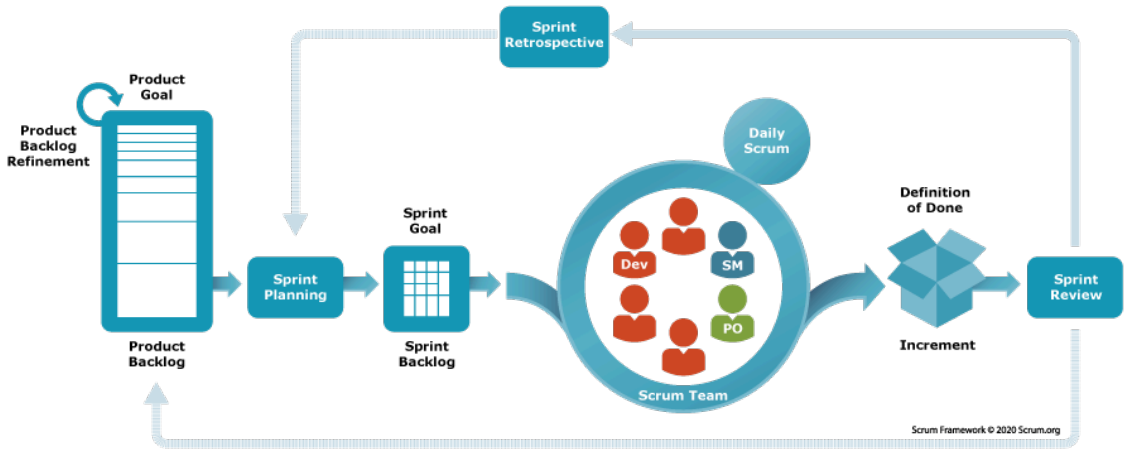

Immagine 1: Modello di sviluppo *Agile*$_G$

Fonte: https://www.scrum$_G$.org/resources/what-is$_G$-scrum$_G$

Quanto mostro nell'immagine 1 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 allinemanto costante tra ogni membro del *team* in merito allo stato di avanzamneto, 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 *Product Backlog*$_G$ e di priorizzare le *User Stories* in base alle esigenze del cliente.
Team leader	Responsabile del coordinamento del *team* di sviluppo, si occupa di assegnare i compiti e di garantire che il *team* sia allineato con gli obiettivi dello *Sprint*$_G$.
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 *bug*$_G$ al *team* 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.

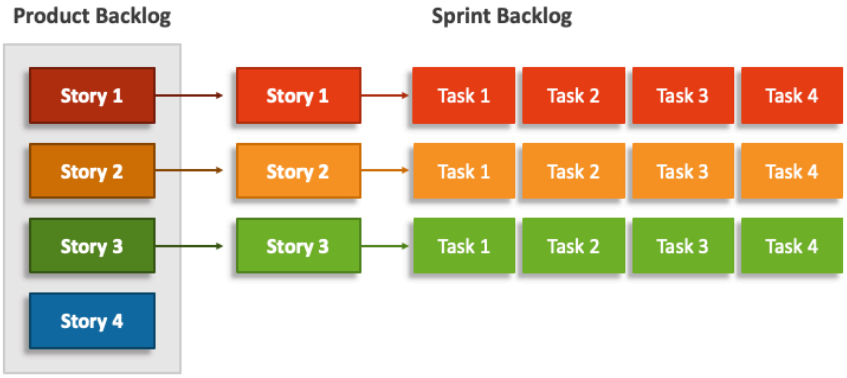

Immagine 2: Relazione *User Stories*, *Product Backlog*$_G$ e *Sprint*$_G$ *Backlog*$_G$
Fonte: https://www.collidu.com/presentation-product-backlog$_G$

Con l'immagine 2 mostro come le *User Stories* siano l'*input* fondamentale per la definizione del *Product Backlog*$_G$ e dello *Sprint*$_G$ *Backlog*$_G$, responsabili del delinea-

mento 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 implementazione 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.

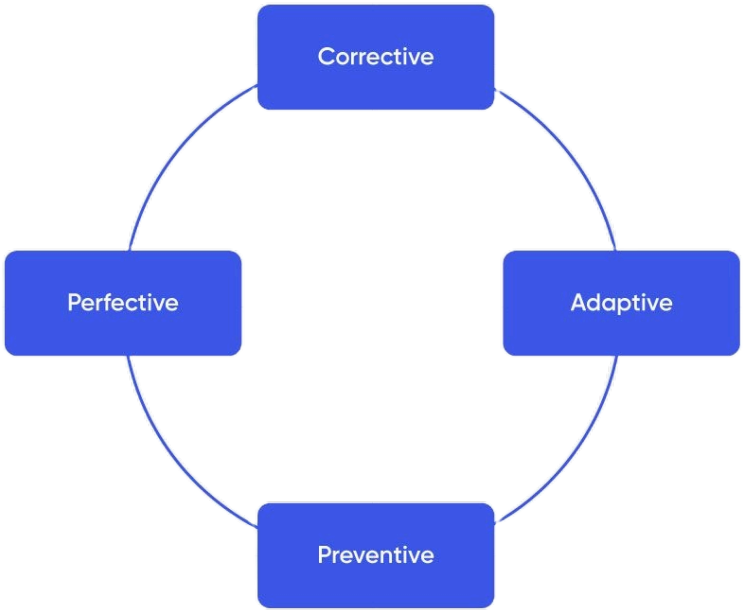

Immagine 3: Manutenzione *software*

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

Come mostro nell'immagine 3, 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*$_G$ 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 affinchè 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, residieva 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.

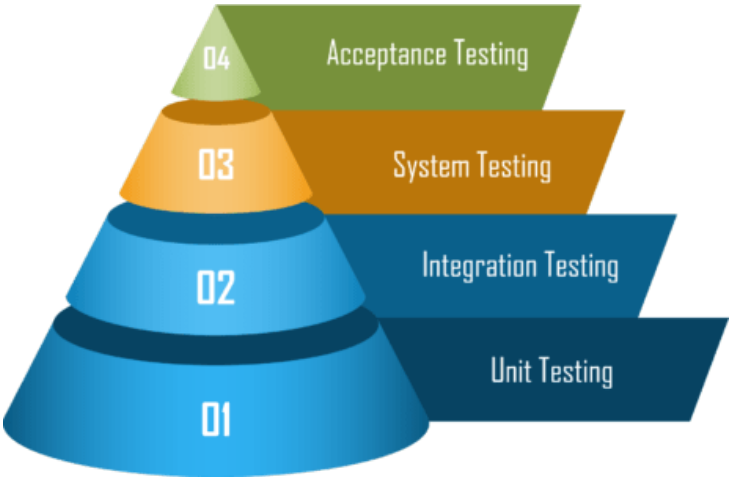

Immagine 4: Le tipologie di *Software testing*$_G$

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

$_G$

Come mostro nell'immagine 4, 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 durante il periodo di tirocinio, 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 lo *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 mostro nell'immagine 5, 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 stata creata;
- **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 assgnato 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.

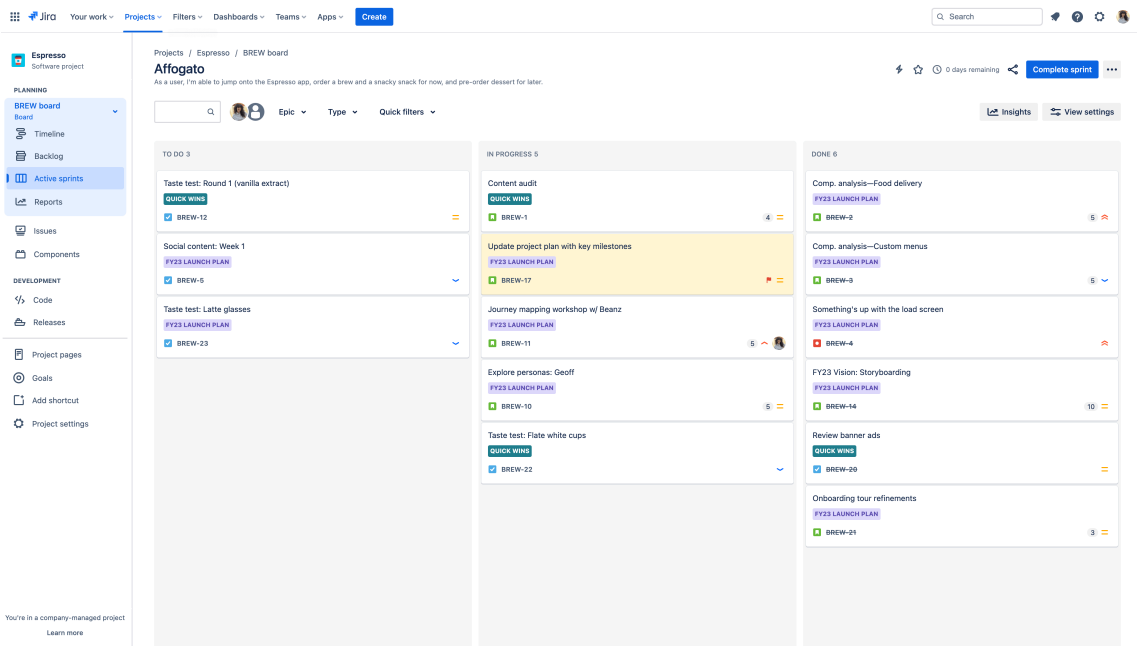

Immagine 5: Esempio di *board* in Jira$_G$

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.

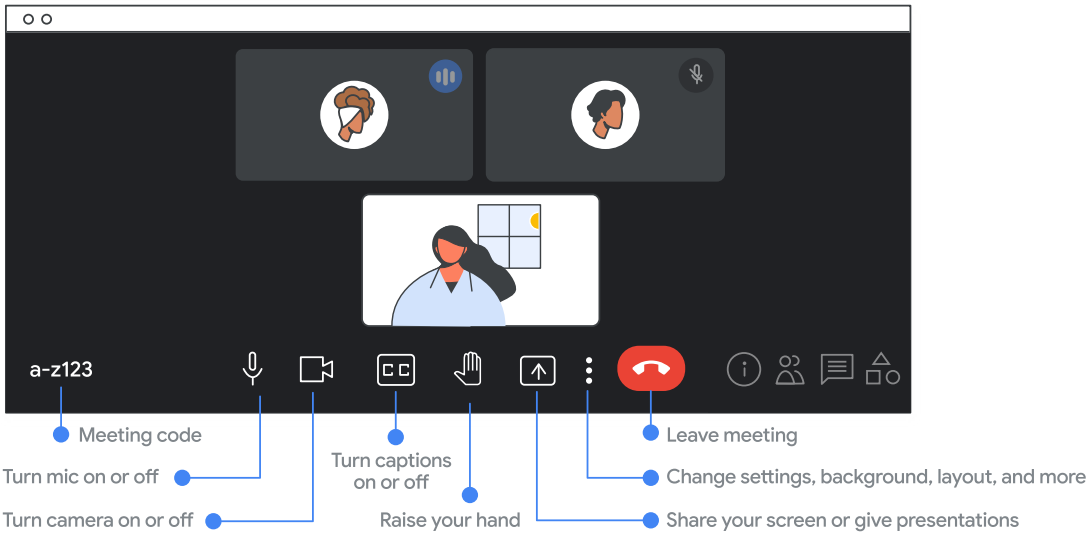

Immagine 6: Interfaccia di *Google Meet*

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*).

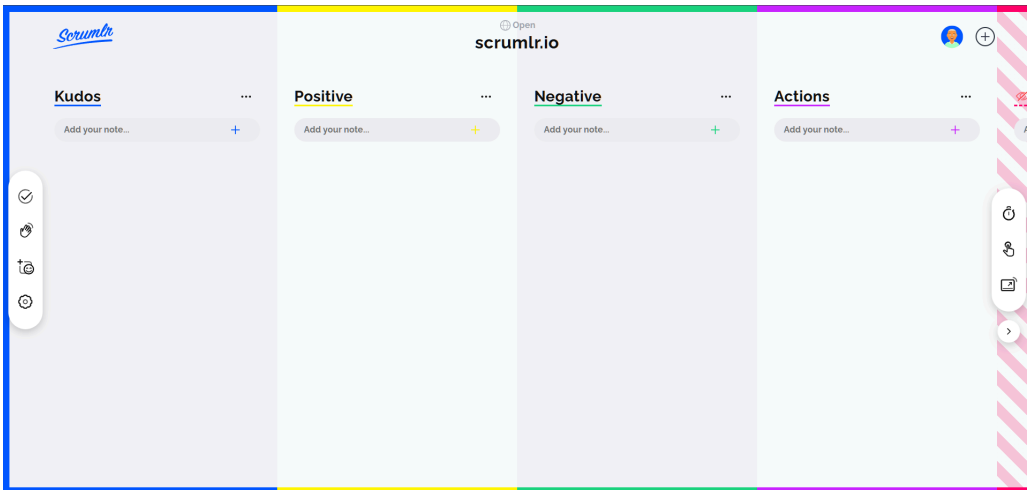

Immagine 7: Interfaccia di *Scrumlr.io*

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.

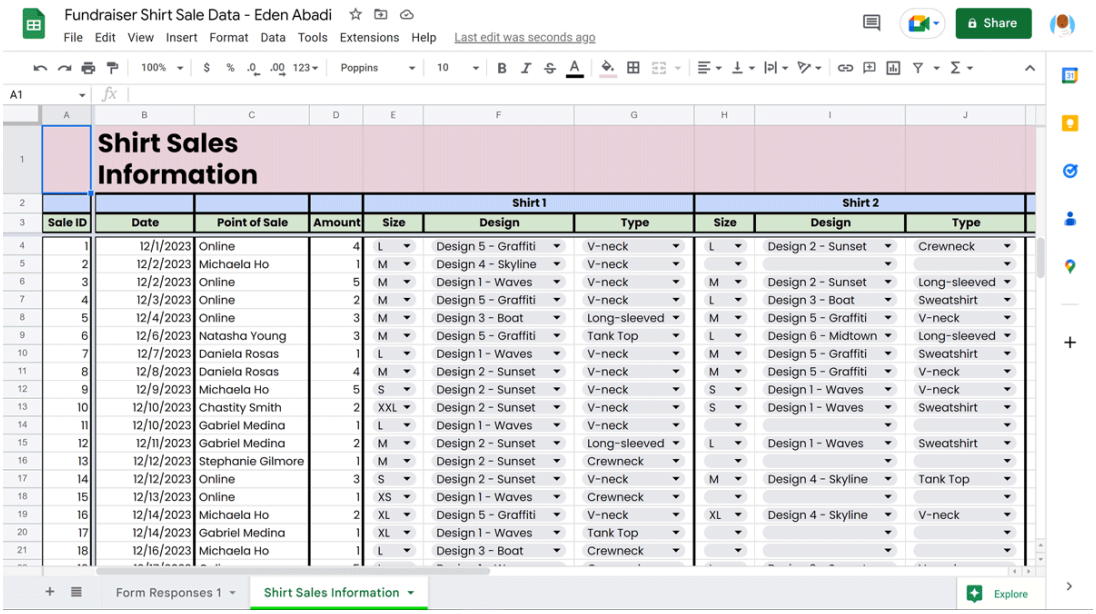

Immagine 8: Interfaccia di *Google Sheets*$_G$

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 in modo preciso 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.

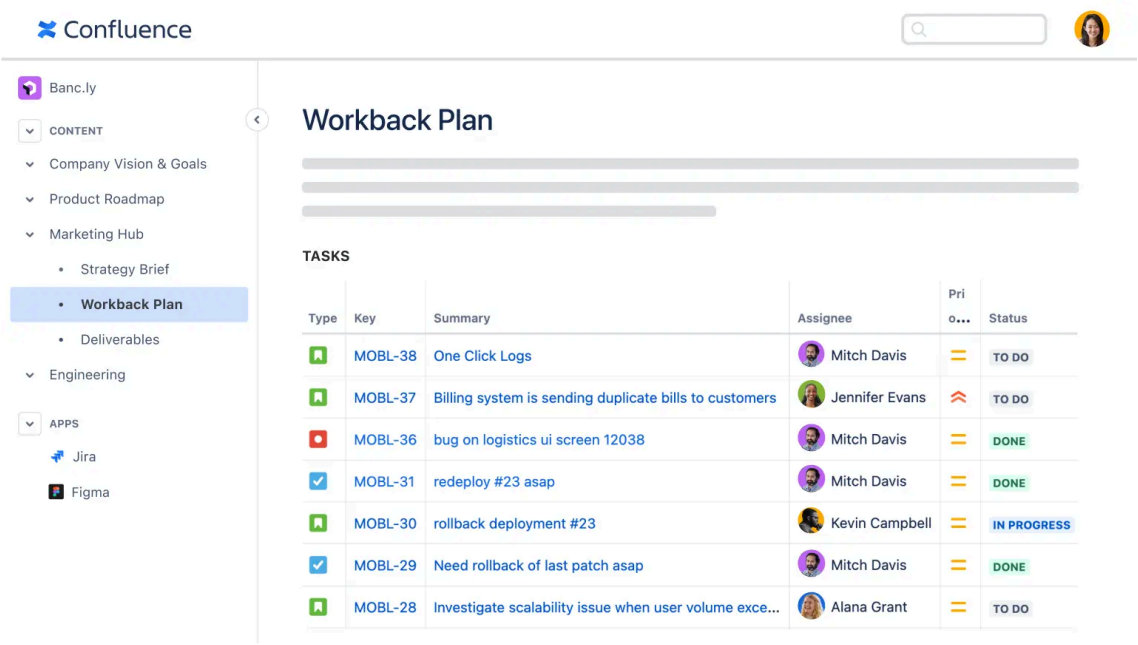

Immagine 9: Interfaccia di *Confluence*

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 il codice sorgente, 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.

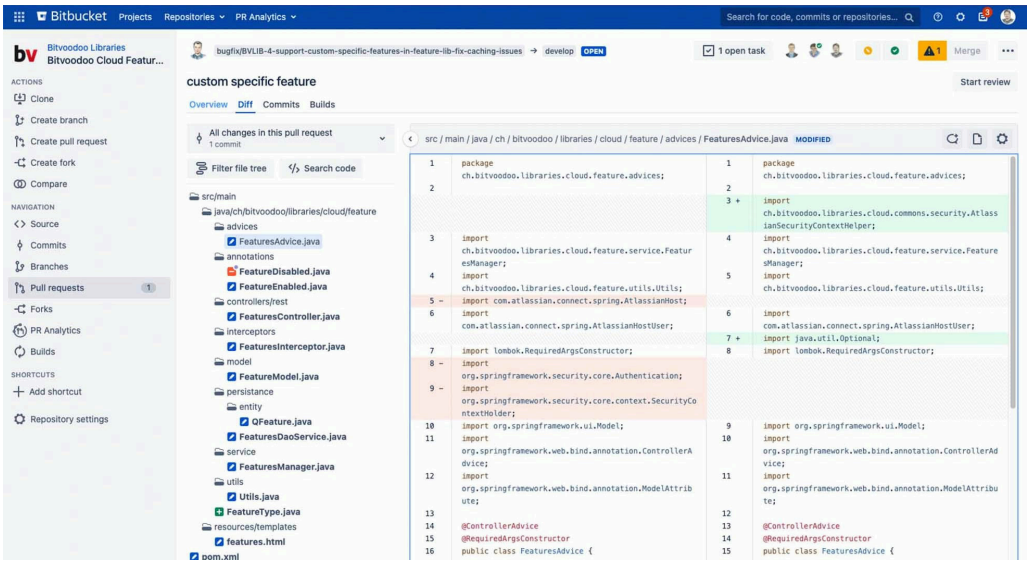

Immagine 10: Interfaccia di Bitbucket$_G$

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.

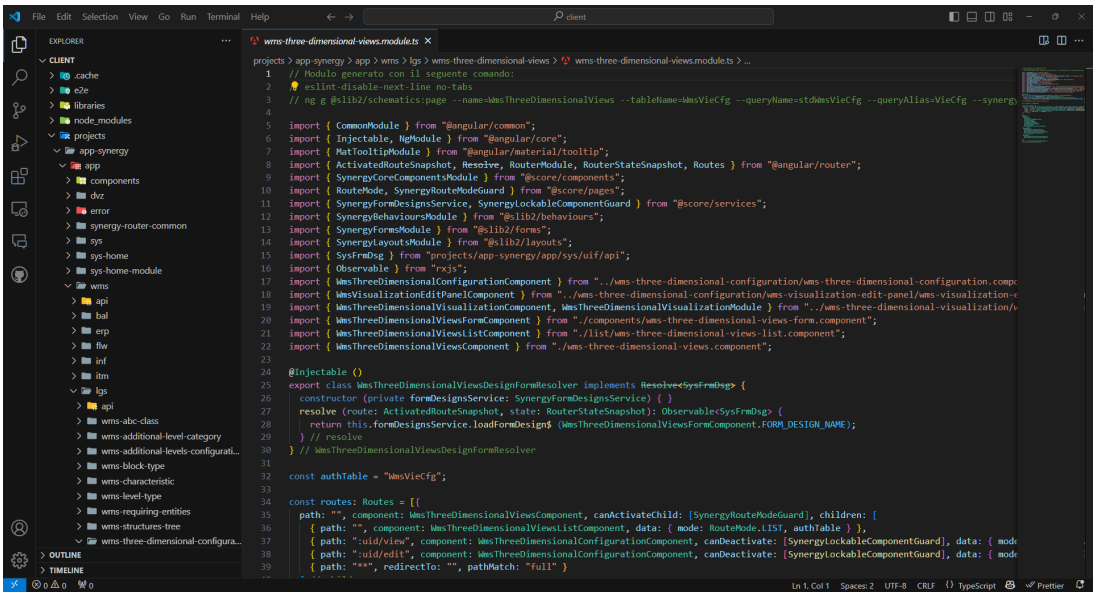

Immagine 11: Interfaccia di *VSCode* con il codice *frontend* del prodotto del tirocinio

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*.

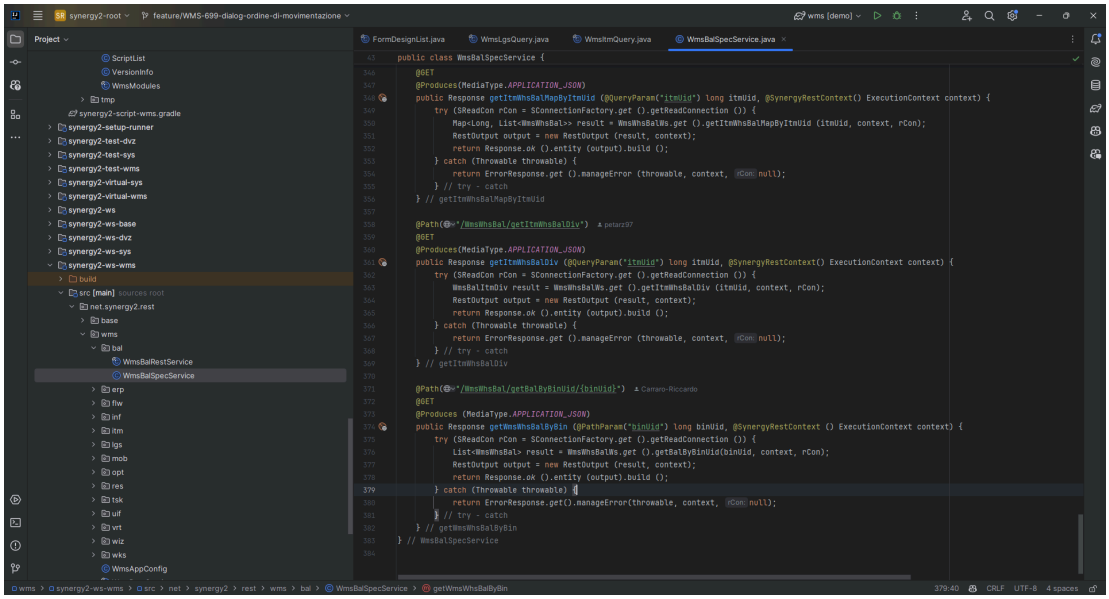

Immagine 12: Interfaccia di *IntelliJ* con il codice *backend* del prodotto del tirocinio

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.

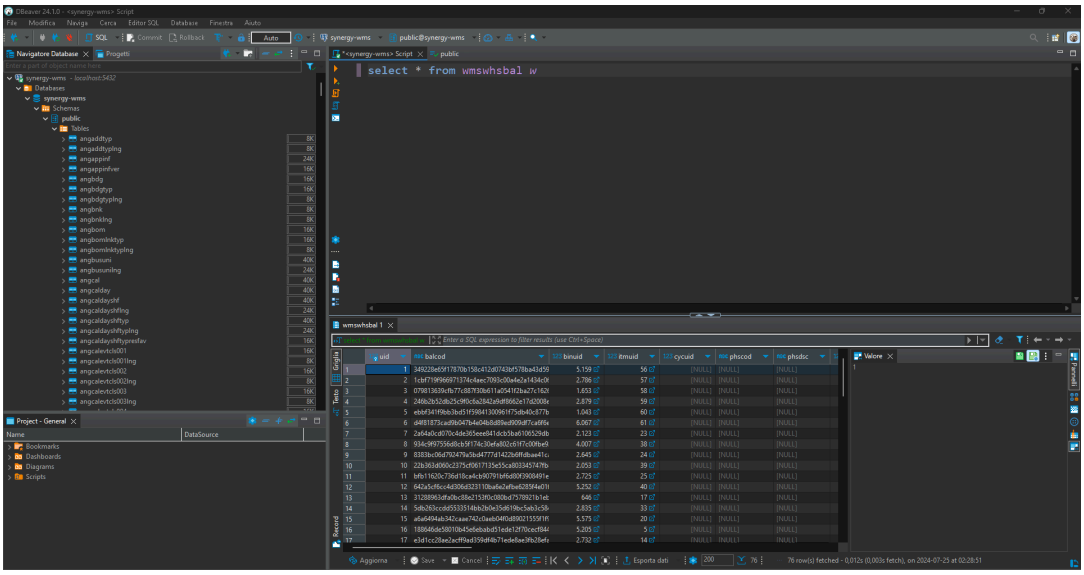

Immagine 13: Interfaccia di DBeaver con il *database*$_G$ del prodotto del tirocinio

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.

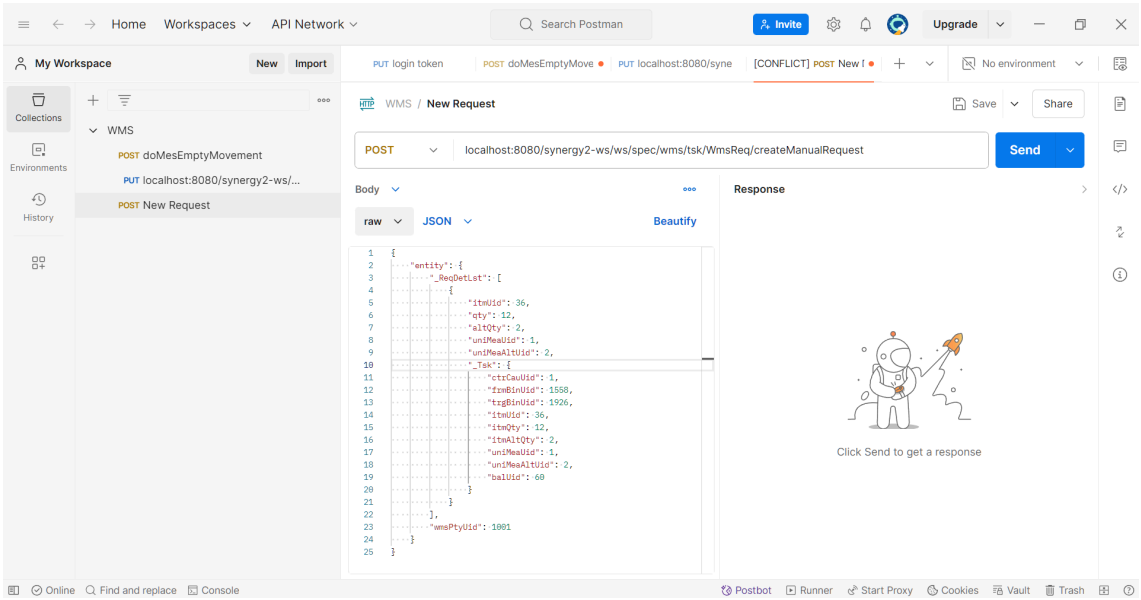

Immagine 14: Esempio di chiamata POST ad un servizio REST con Postman

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*:

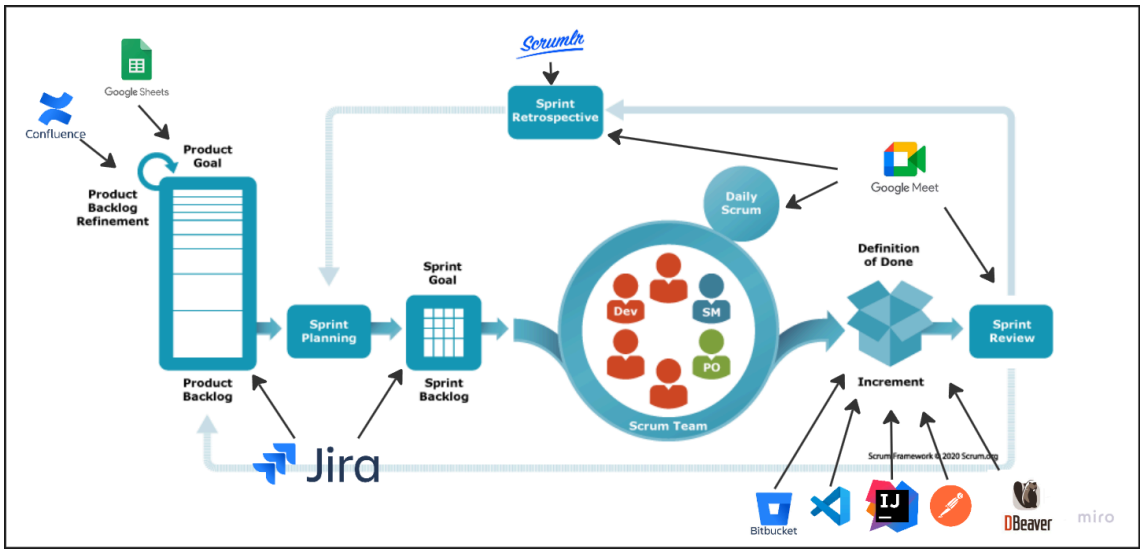

Immagine 15: Come gli strumenti si integrano nel modello di sviluppo aziendale

1.4.5.7 Gestione delle risorse umane

Il processo di gestione delle risorse umane è quello che 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 comequesto 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, permettondomi di apprendere le basi del *framework Synergy* mediante un approccio *learn by doing*. Inoltre la formazione è un processo continuo che anche i membri del *team* a cui sono stato affiancato, svolgono costantemente, grazie ai corsi offerti dall'azienda nella piattaforma Udemy.

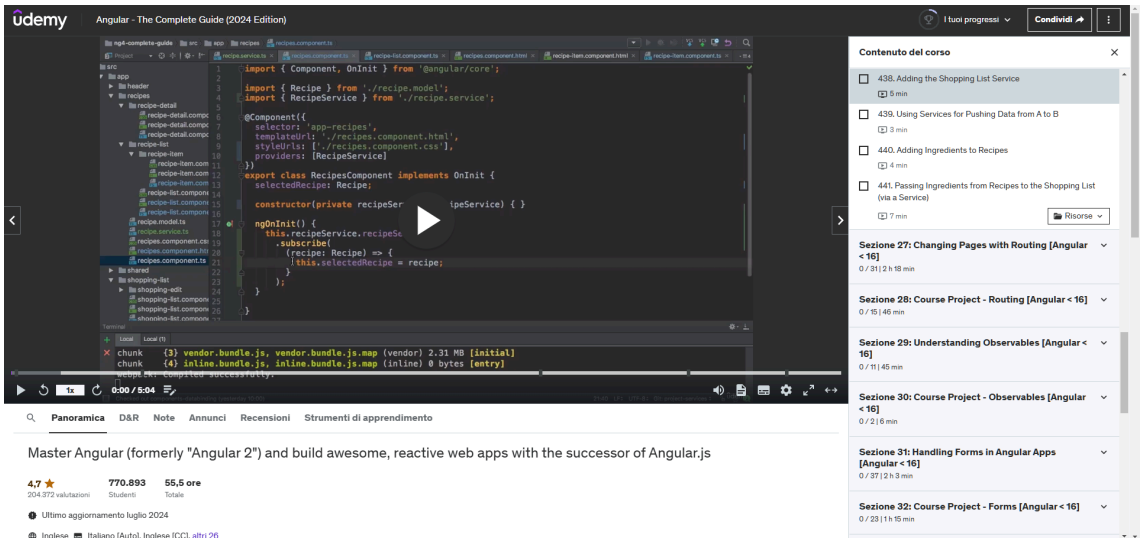

Immagine 16: Corso di formazione Angular su Udemy

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

Come mostro nell'immagine 16 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 costantemtne corsi di formazione su nuove tecnologie e strumenti, avvalendosi di esperti interni e consulenti esterni, e utilizzando piattaforme di e-learning come Udemy Business. 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

Descrizione dell'approccio dell'azienda con lo stage e la sua importanza, in relazione al ruolo che ricopre l'innovazione per Sanmarco Informatica.

2.2 Il progetto proposto

2.2.1 Descrizione del progetto

Descrizione del progetto proposto, partendo da una descrizione ad alto livello di cos'è il prodotto WMS, e di come il lavoro da me svolto lo sia andato ad integrare, con quali funzionalità e a quali necessità doveva rispondere.

2.2.2 Obiettivi

2.2.2.1 Obiettivi aziendali

Gli obiettivi del tirocinio di interesse aziendale, individuati all'interno del piano di stage.

2.2.2.2 Obiettivi personali

Gli obiettivi del tirocinio di interesse personali, individuabili nello sviluppo e miglioramento delle capacità tecniche e organizzative, sfruttando la possibilità di essere inserito in un contesto aziendale.

2.2.3 Vincoli

2.2.3.1 Vincoli temporali

I vincoli temporali a cui sottostare durante il periodo di tirocinio.

2.2.3.2 Vincoli tecnologici

I vincoli tecnologici.

2.2.3.3 Vincoli organizzativi

I vincoli organizzativi.

2.3 Motivazione della scelta

Le motivazioni che mi hanno spinto a svolgere questo percorso tirocinio.

3 Svolgimento del tirocinio

3.1 Pianificazione

3.2 Metodo di lavoro

3.2.1 *Way of Working*$_{G}$

Come mi sono approcciato al tirocinio, le attività che lo hanno caratterizzato, l'organizzazione per sprint$_{G}$ e i compiti assegnati su Jira$_{G}$. In questa sezione discuterei dei principi appresi durante il corso di ingegneria del software$_{G}$ (e consolidati durante il periodo di tirocinio) che hanno caratterizzato il mio *way of working*$_{G}$.

3.2.2 Obiettivi di qualità

Panoramica del mio approccio alla realizzazione di prodotto software e documentale di qualità, descrivendo gli obiettivi dei processi di verifica e validazione.

3.2.3 Obiettivi di qualità di processo

Panoramica del mio approccio a garantire efficacia ed efficienza nel conseguimento dei miei obiettivi.

3.2.4 Interazione con il referente aziendale

Il rapporto che ho tenuto durante lo svolgimento del tirocinio con il mio referente aziendale.

3.2.5 Revisioni di progresso

Come durante il tirocinio ho svolto le revisioni di progresso insieme al_team_e al tutor aziendale.

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).

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 tencologico 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, condividendola anche il referente aziendale. Questa sezione riporterà 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

Anali 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

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

Specifica 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.

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 (una settimana nel nostro specifico caso), 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

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)*

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)*