

# 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

<b>1 L’azienda Sanmarco Informatica .....</b>	<b>1</b>
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
<b>2 Il tirocinio .....</b>	<b>20</b>
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
2.4 Premesse allo svolgimento del tirocinio .....	26
2.4.1 Approccio al lavoro .....	26

---

2.4.2 Obiettivi di qualità .....	27
2.4.3 Obiettivi di qualità di processo .....	28
<b>3 Svolgimento del tirocinio .....</b>	<b>29</b>
3.1 Pianificazione .....	29
3.2 Metodo di lavoro .....	32
3.2.1 <i>Way of Working</i> .....	32
3.2.2 Interazione con il referente aziendale .....	33
3.2.3 Revisioni di progresso .....	33
3.2.4 Strumenti di verifica .....	34
3.2.5 Resoconti .....	36
3.3 Analisi dei requisiti .....	37
3.3.1 Casi d'uso .....	37
3.3.2 Tracciamento dei requisiti .....	42
3.4 Progettazione .....	45
3.4.1 Tecnologie utilizzate .....	45
3.4.2 <i>Workspace</i> e <i>widget</i> .....	48
3.4.3 Progettazione dell'ambiente tridimensionale .....	49
3.4.4 Progettazione della funzionalità di creazione degli ordini di movimentazione .....	51
3.4.5 Architettura del sistema .....	53
3.5 Codifica .....	55
3.5.1 Visualizzazione tridimensionale .....	55
3.5.2 Drag & Drop e creazione ordini di movimentazione .....	57
3.6 Verifica e validazione .....	60
3.6.1 <i>Test</i> di unità .....	60
3.6.2 <i>Test</i> di integrazione .....	60
3.6.3 <i>Test</i> prestazionali .....	61
3.6.4 <i>Test</i> di sistema .....	62
3.6.5 <i>Test</i> di accettazione .....	62
3.7 Risultati raggiunti .....	63
3.7.1 Il prodotto realizzato .....	63
3.7.2 Copertura dei requisiti .....	65
3.7.3 Copertura di testing .....	66
3.7.4 Materiali prodotti .....	66
<b>4 Valutazione retrospettiva .....</b>	<b>68</b>
4.1 Soddisfacimento degli obiettivi .....	68
4.1.1 Obiettivi aziendali .....	68
4.1.2 Obiettivi personali .....	68
4.2 Competenze acquisite .....	68

---

4.3 Valutazione personale .....	68
4.4 Università e mondo del lavoro .....	68
<b>Acronimi e abbreviazioni .....</b>	<b>69</b>
<b>Glossario .....</b>	<b>70</b>
<b>Bibliografia e sitografia .....</b>	<b>75</b>

---

# 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: L'importanza del <i>Way of Working</i> nel SEMAT .....	26
Immagine 20: Diagramma di Gantt delle attività del primo periodo .....	30
Immagine 21: Diagramma di Gantt delle attività del secondo periodo .....	30
Immagine 22: Diagramma di Gantt delle attività del terzo periodo .....	31
Immagine 23: Diagramma di Gantt delle attività del quarto periodo .....	32

---

Immagine 24: Diagramma di Gantt complessivo delle attività svolte durante il tirocinio .....	32
Immagine 25: Bacheca personale su Notion .....	33
Immagine 26: Pipeline di <i>Continuous Integration</i> e <i>Continuous Deployment</i> .....	34
Immagine 27: <i>Pipeline</i> per l'accettazione di una <i>Pull Request</i> .....	36
Immagine 28: Casi d'uso per l'ambiente tridimensionale .....	38
Immagine 29: Casi d'uso per la creazione dell'ordine di movimentazione .....	40
Immagine 30: Struttura Angular .....	46
Immagine 31: Ambiente 3D realizzato durante il tirocinio con Three.js .....	47
Immagine 32: Integrazione delle tecnologie utilizzate .....	48
Immagine 33: Esempio dei <i>widget</i> presenti nel <i>workspace</i> .....	49
Immagine 34: Cambiamento apportato durante il <i>refactor</i> dell'ambiente 3D ....	49
Immagine 35: Comportamento del <i>dependency injector</i> in Angular .....	50
Immagine 36: Comportamento creazione dell'ordine di movimentazione .....	52
Immagine 37: Tabelle coinvolte nella creazione degli ordini di movimentazione .	
53	
Immagine 38: Pattern MVVM del <i>frontend</i> con Angular .....	53
Immagine 39: Architettura a <i>layer</i> del backend con Synergy .....	54
Immagine 40: Diagramma UML delle classi Bin3D e Struct3D .....	55
Immagine 41: Bin3D e Struct3D all'interno dell'ambiente 3D .....	56
Immagine 42: Interazione con i bin .....	57
Immagine 43: <i>Dialog</i> per la creazione degli ordini di movimentazione .....	58
Immagine 44: <i>Drag &amp; drop</i> nell'ambiente 3D .....	59
Immagine 45: Servizio REST per la creazione dell'ordine esposto dal <i>backend</i> .	59
Immagine 46: <i>Layer WS</i> per la creazione dell'ordine .....	60
Immagine 47: <i>Test</i> prestazionali dell'ambiente 3D .....	61
Immagine 48: Uso della GPU prima e dopo il caricamento dell'ambiente 3D ...	62
Immagine 49: Interfaccia finale della <i>workspace</i> .....	63
Immagine 50: Visualizzazione dei saldi contenuti all'interno del <i>bin</i> selezionato .	
64	

---

Immagine 51: Visualizzazione dei <i>bin</i> con almeno un saldo .....	64
Immagine 52: Visualizzazione posizione del saldo selezionato dal <i>widget</i> dei saldi .....	64
Immagine 53: <i>Dialog</i> di creazione dell'ordine di movimentazione .....	65

---

# Indice delle tabelle

Tabella 1: Ruoli aziendali .....	5
Tabella 2: Obiettivi aziendali .....	22
Tabella 3: Obiettivi personali .....	23
Tabella 4: Macrosuddivisione del tirocinio .....	29
Tabella 5: Requisiti funzionali .....	42
Tabella 6: Requisiti di qualità .....	44
Tabella 7: Requisiti prestazionali .....	44
Tabella 8: Requisiti di vincolo .....	44
Tabella 9: Riepilogo requisiti .....	45
Tabella 10: Copertura dei requisiti .....	65
Tabella 11: Copertura dei <i>test</i> .....	66
Tabella 12: Materiali complessivamente prodotti durante il tirocinio .....	66

---

# 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.

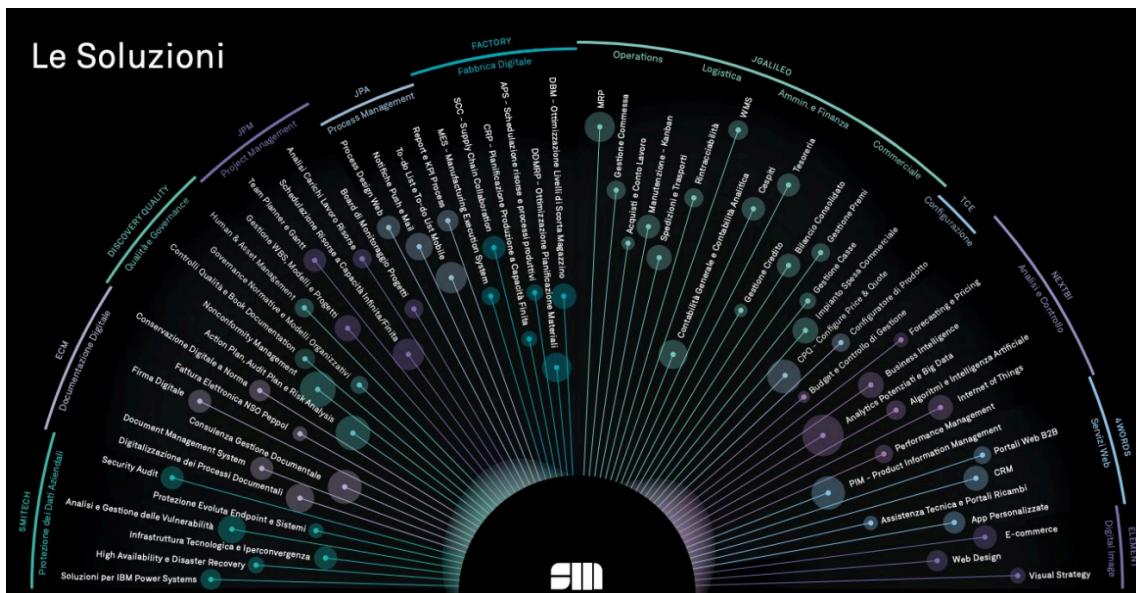
## 1.2 Organizzazione aziendale e i prodotti

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.sanmarcoinformatica.it/intranet.pag>

## 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<sub>G</sub>*, implementando nello specifico il *framework Scrum<sub>G</sub>*. 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.

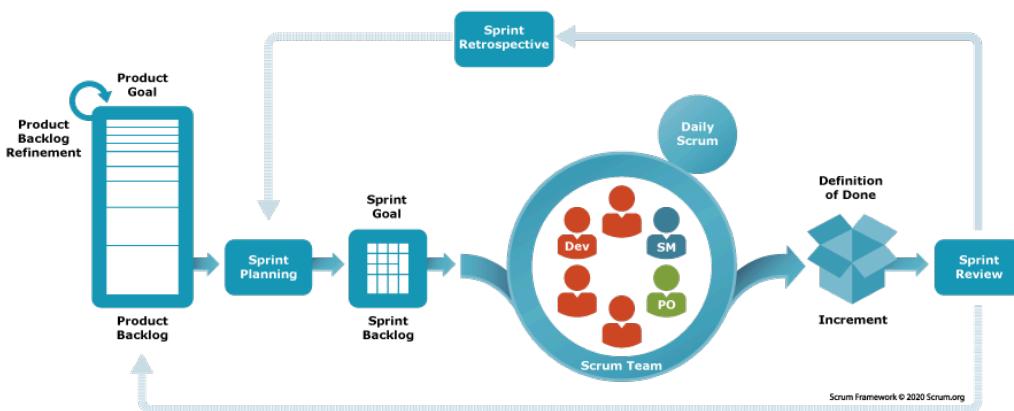


Immagine 2: Modello di sviluppo *Agile<sub>G</sub>*

Fonte: [https://www.scrum.org/resources/what-is-G-scrum\\_G](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<sub>G</sub>* sono le *User Stories* definite in collaborazione con il cliente, sulla base delle quali si andrà a definire il *Product Backlog<sub>G</sub>*, ovvero l'insieme di tutte i *task<sub>G</sub>* che il *team* di sviluppo dovrà svolgere al fine di implementare le funzionalità desiderate.

Il modello *Agile<sub>G</sub>* suddivide il periodo di realizzazione in *Sprint<sub>G</sub>*, 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<sub>G</sub> Backlog<sub>G</sub>*.

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<sub>G</sub>*, denominata *Sprint<sub>G</sub> Review<sub>G</sub>*, durante la quale il *team* presenta il progresso ottenuto, susseguita successivamente dalla *Sprint<sub>G</sub> 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<sub>G</sub>* successivo e alla definizione del nuovo *Sprint<sub>G</sub> Backlog<sub>G</sub>*.

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<sub>G</sub>* 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
<b>Product Owner</b>	Responsabile della definizione delle funzionalità del prodotto, in collaborazione con il cliente. Si occupa di definire il <i>Product Backlog</i> <sub>G</sub> e di priorizzare le <i>User Stories</i> in base alle esigenze del cliente.
<b>Team leader</b>	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> <sub>G</sub> .
<b>Sviluppatore</b>	Responsabile della realizzazione effettiva delle funzionalità del prodotto.
<b>Tester</b>	Responsabile della verifica del prodotto, si occupa di testare le funzionalità implementate e di segnalare eventuali <i>bug</i> <sub>G</sub> al <i>team</i> di sviluppo.
<b>Consulente</b>	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*<sub>G</sub> infatti, vi è la capacità di adattamento dello sviluppo ai cambiamenti, ottenibile mediante una stretta collaborazione tra il *Product Owner* e il cliente.

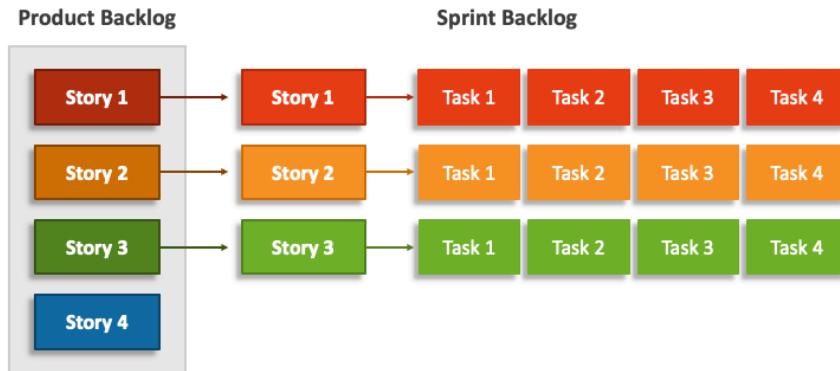


Immagine 3: Relazione *User Stories*, *Product Backlog<sub>G</sub>* e *Sprint<sub>G</sub> Backlog<sub>G</sub>*

Fonte: [https://www.collidu.com/presentation-product-backlog<sub>G</sub>](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<sub>G</sub>* e dello *Sprint<sub>G</sub> Backlog<sub>G</sub>*, responsabili del delineamento delle funzionalità del prodotto e delle attività da svolgere durante lo *Sprint<sub>G</sub>*.

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<sub>G</sub>* 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<sub>G</sub> e *mockup* dell’interfaccia.

- **Software coding and testing<sub>G</sub>**: attività di codifica e *test* del prodotto. Il suo obiettivo è l’implementazione delle funzionalità e verificare che siano conformi alle aspettative. Il *testing<sub>G</sub>* 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<sub>G</sub>*.
- **Software qualification testing<sub>G</sub>**: 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<sub>G</sub>*, dove grazie allo strumento di controllo di versione Bitbucket<sub>G</sub> ([paragrafo 1.4.5.5](#)), ad ogni modifica apportata alla *codebase<sub>G</sub>* 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.

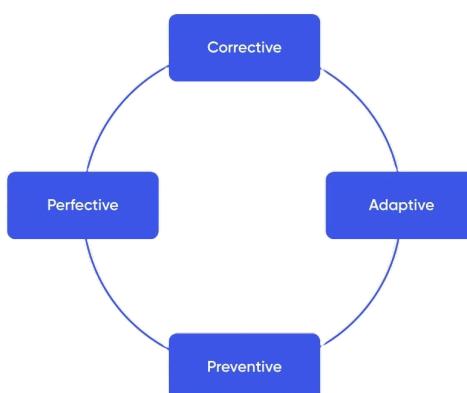


Immagine 4: Manutenzione *software*

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.

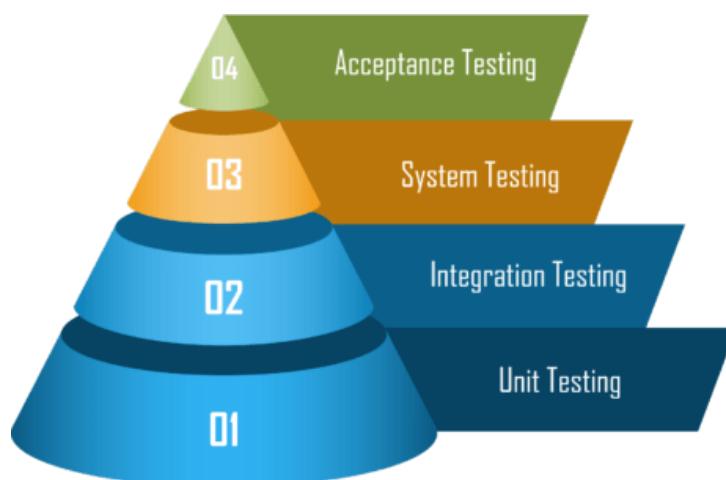


Immagine 5: Le tipologie di *Software testing*<sub>G</sub>

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<sub>G</sub>**: 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<sub>G</sub>**: 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<sub>G</sub>**: 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<sub>G</sub>**: 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 e di integrazione, con l’obiettivo di garantire che il prodotto fosse pronto per il *Software qualification testing<sub>G</sub>* ([paragrafo 1.4.3.2](#)).

Nel mio caso infatti, prima di procedere all’integrazione della *codebase<sub>G</sub>* 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<sub>G</sub>

Jira<sub>G</sub> è uno strumento di *issue tracking system<sub>G</sub>* (ITS<sub>G</sub>) utilizzato dall’azienda per la gestione delle attività di sviluppo. Lo strumento permette al *team leader* ad ogni *Sprint<sub>G</sub> planning*, di strutturare la *board* con i diversi *task<sub>G</sub>* (o *issue<sub>G</sub>*) da svolgere

entro la fine dello *Sprint<sub>G</sub>*, 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<sub>G</sub> permette di strutturare la *board* in modo da avere una visione di insieme delle attività da svolgere, con la possibilità di organizzare i *task<sub>G</sub>* in colonne in base allo stato di avanzamento.

Durante il mio tirocinio ho utilizzato lo strumento secondo le convenzioni aziendali, lavorando su *task<sub>G</sub>* di due tipologie principali:

- **Bug<sub>G</sub>**: attività di correzione di *bug<sub>G</sub>* e problematiche riscontrate nel prodotto;
- **User story<sub>G</sub>**: attività di implementazione di nuove funzionalità.

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

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

Le integrazioni con strumenti come Bitbucket<sub>G</sub> ([paragrafo 1.4.5.5](#)) rendono Jira<sub>G</sub> uno strumento estremamente versatile e in grado di adattarsi alle diverse esigenze dell'azienda.

The screenshot shows a Jira board titled "Affogato". The board has three main columns: "TO DO 3", "IN PROGRESS 5", and "DONE 4".

- TO DO 3:**
  - Taste test: Board 1 (vanilla extract) - **READY FOR TEST** (BREW-12)
  - Social content: Week 1 - **PY23 LAUNCH PLAN** (BREW-6)
  - Taste test: Latte glasses - **PY23 LAUNCH PLAN** (BREW-22)
- IN PROGRESS 5:**
  - Content with **BLOCK WIPE** (BREW-1)
  - Update project plan with key milestones - **PY23 LAUNCH PLAN** (BREW-10)
  - Journey mapping workshop v1 (Brew) - **PY23 LAUNCH PLAN** (BREW-11)
  - Explore personas: Goff - **PY23 LAUNCH PLAN** (BREW-15)
  - Taste test: Fibre white cupcake - **READY FOR TEST** (BREW-23)
- DONE 4:**
  - Comp. analysis - Food delivery - **PY23 LAUNCH PLAN** (BREW-2)
  - Comp. analysis - Custom menus - **PY23 LAUNCH PLAN** (BREW-3)
  - Something's up with the load screen - **PY23 LAUNCH PLAN** (BREW-4)
  - PY23 Vision Storyboarding - **PY23 LAUNCH PLAN** (BREW-14)
  - Review banner ads - **READY FOR TEST** (BREW-40)
  - Onboarding tour refinements - **PY23 LAUNCH PLAN** (BREW-21)

Immagine 6: Esempio di *board* in Jira<sub>G</sub>

Fonte: [https://www.atlassian.com/it/software/jira<sub>G</sub>/guides/boards/overview#what-is<sub>G</sub>-a-jira<sub>G</sub>-board](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<sub>G</sub> Review<sub>G</sub>* e *Sprint<sub>G</sub> 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.

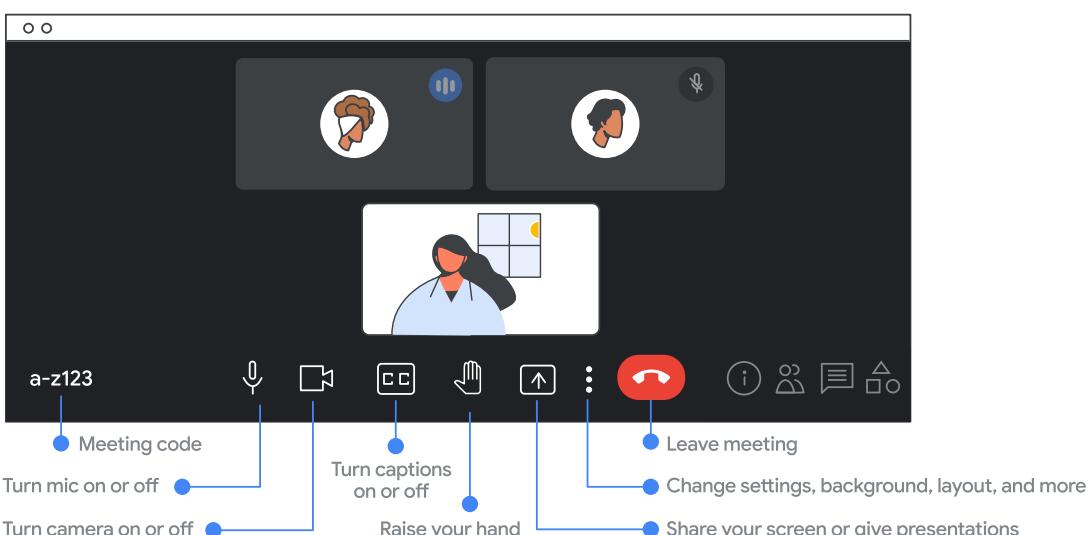


Immagine 7: Interfaccia di Google Meet

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<sub>G</sub> Retrospective*, dove ogni membro del *team* può inserire i propri *feedback* e le proprie considerazioni relative allo *Sprint<sub>G</sub>* 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

Google Sheets 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* del prodotto e per il tracciamento dei requisiti che intendono soddisfare.

The screenshot shows a Google Sheets document titled "Fundraiser Shirt Sale Data - Eden Abadi". The main table is titled "Shirt Sales Information" and is divided into two sections: "Shirt 1" and "Shirt 2". The columns for both sections include Sale ID, Date, Point of Sale, Amount, Size, Design, and Type. The table contains 21 rows of data, each representing a sale. The data includes various dates from December 1, 2023, to December 16, 2023, and different sales points like Online and Michaela Ho. The "Shirt 1" section has 13 rows, and the "Shirt 2" section has 8 rows. The "Amount" column shows values like 4, 1, 5, etc. The "Size" column includes options like L, M, XXL, S, XS, XL, and XXL. The "Design" column lists various patterns like Design 5 - Graffiti, Design 1 - Waves, etc. The "Type" column includes V-neck, Crewneck, Long-sleeved, Boat, and Midtown. The "Point of Sale" column shows names like Michaela Ho, Daniela Rosas, and Gabriel Medina. The "Date" column shows specific dates like 12/1/2023, 12/2/2023, etc. The "Amount" column shows the quantity sold. The "Size" column shows the size of the shirt. The "Design" column shows the specific design of the shirt. The "Type" column shows the type of shirt (e.g., V-neck, Crewneck). The "Point of Sale" column shows the person who made the sale. The "Date" column shows the date of the sale.

Immagine 9: Interfaccia di Google Sheets

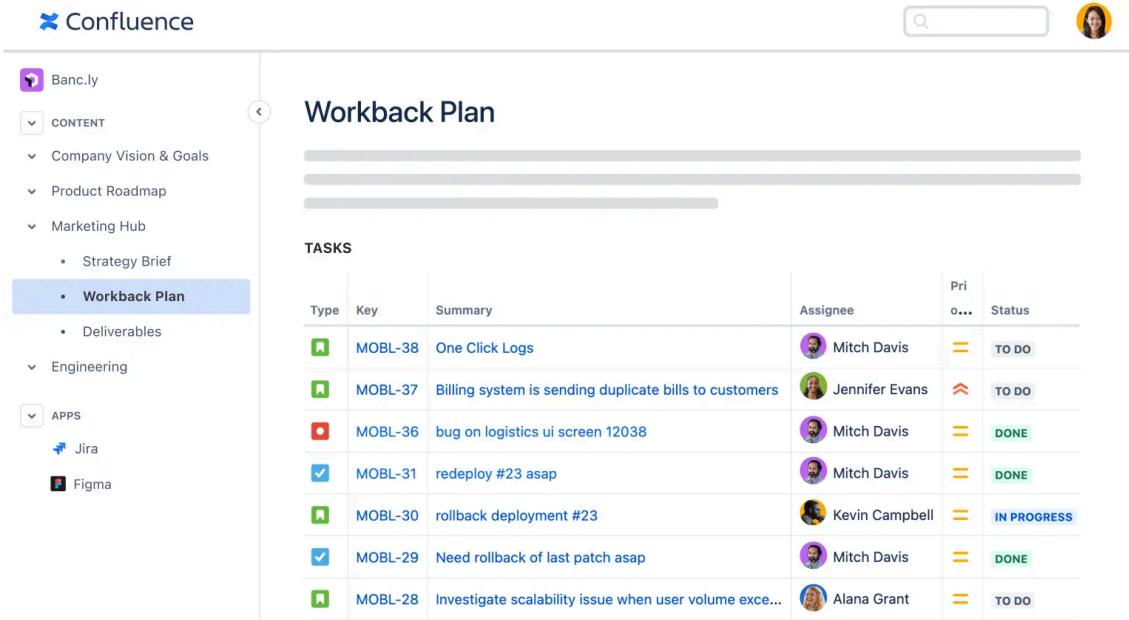
Fonte: <https://support.google.com/meet/answer/10550593?hl=it>

## 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	<span style="color: orange;">=</span>	TO DO
BUG	MOBL-37	Billing system is sending duplicate bills to customers	Jennifer Evans	<span style="color: red;">△</span>	TO DO
BUG	MOBL-36	bug on logistics ui screen 12038	Mitch Davis	<span style="color: orange;">=</span>	DONE
BUG	MOBL-31	redeploy #23 asap	Mitch Davis	<span style="color: orange;">=</span>	DONE
BUG	MOBL-30	rollback deployment #23	Kevin Campbell	<span style="color: orange;">=</span>	IN PROGRESS
BUG	MOBL-29	Need rollback of last patch asap	Mitch Davis	<span style="color: orange;">=</span>	DONE
BUG	MOBL-28	Investigate scalability issue when user volume exce...	Alana Grant	<span style="color: orange;">=</span>	TO DO

Immagine 10: Interfaccia di *Confluence*

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

### 1.4.5.5 Strumenti di sviluppo

#### Bitbucket<sub>G</sub>

Bitbucket<sub>G</sub> è uno strumento di controllo di versione utilizzato dall'azienda per la gestione del codice sorgente. Lo strumento permette di creare *repository<sub>G</sub>* in cui caricare la *codebase<sub>G</sub>*, e di gestire i diversi *branch<sub>G</sub>* di sviluppo affinchè l'avanzamento dei lavori possa avvenire in modo parallelo, coordinato e collaborativo.

Grazie all'integrazione con Jira<sub>G</sub>, Bitbucket<sub>G</sub> permette di collegare i *task<sub>G</sub>* presenti nella *board* con i *branch<sub>G</sub>* di sviluppo, in modo da garantire che ogni *task<sub>G</sub>* sia associato al *branch<sub>G</sub>* corrispondente.

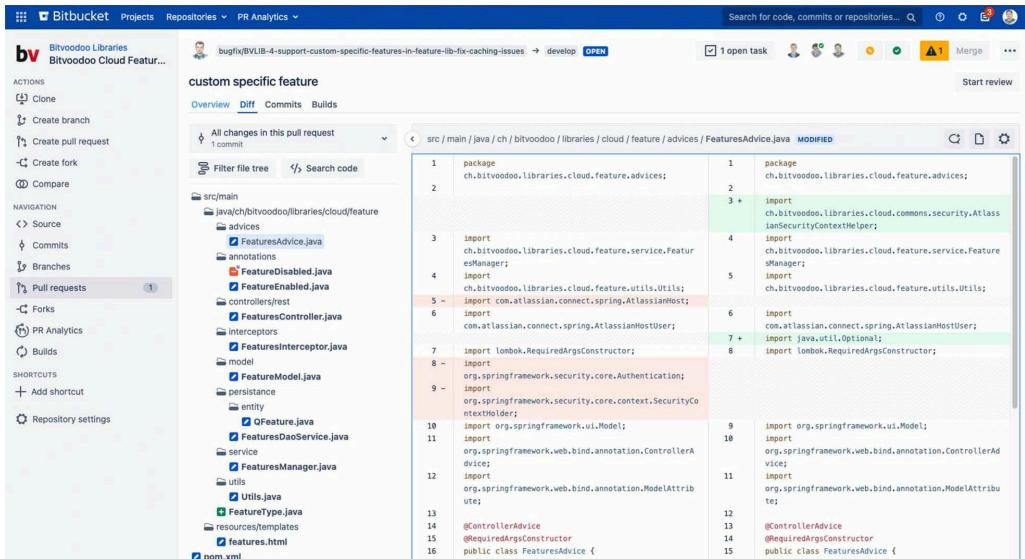


Immagine 11: Interfaccia di Bitbucket<sub>G</sub>

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

## *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* del codice.

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

```
File Edit Selection Go Run Terminal Help < > client

EXPLORER
CLIENT
  > .cache
  > e2e
  > libraries
  > node_modules
  > app
    > app-synergy
      > app
        > components
        > dzh
        > error
        > sys
        > sys-home
        > sys-home-module
      > wms
        > api
        > bal
        > erp
        > fw
        > inf
        > item
      > lgs
        > api
        > wms-abc-class
        > wms-additional-level-category
        > wms-additional-level-configur...
        > wms-block-type
        > wms-characteristic
        > wms-level-type
        > wms-requiring-entities
        > wms-structures-tree
      > wms-three-dimensional-configur...
> OUTLINE
> TIMELINE
> WIDGETS

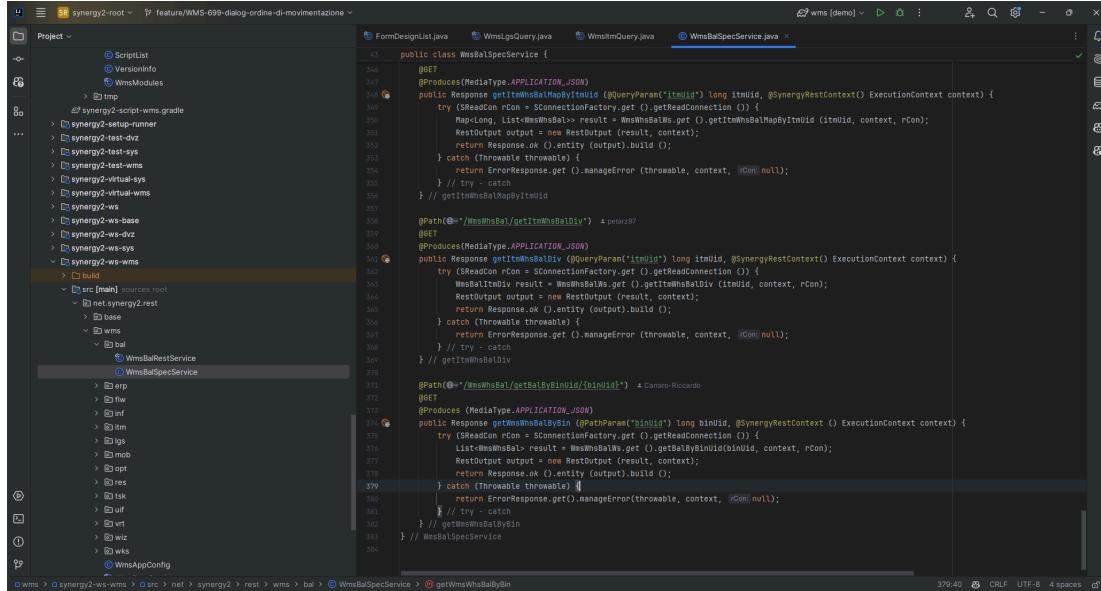
wms-three-dimensional-views.module.ts x
projects // app-synergy app > wms > ts > wms-three-dimensional-views > wms-three-dimensional-views.module.ts ...
1  // Modulo generado con el siguiente comando:
2  1 eslintrc-disable-next-line no-tabs
3  // ng g @lib#schemas:page --name=WmsThreeDimensionalViews --tableName=WmsVieCfg --queryName=stdWmsVieCfg --queryAlias=VieCfg --synonym
4
5  import { CommonModule } from '@angular/common';
6  import { Injectable, IgModule } from '@angular/core';
7  import { MatTooltipModule } from '@angular/material/tooltip';
8  import { ActivatedRouteSnapshot, Resolve, RouterModule, RouterStateSnapshot, Routes } from '@angular/router';
9  import { SynergyCoreComponentsModule } from '@core/components';
10 import { RouteMode, SynergyRouteModeGuard } from '@core/pages';
11 import { SynergyFormDesignsService, SynergyLockableComponentGuard } from '@core/services';
12 import { SynergyBehavioursModule } from '@lib#behaviours';
13 import { SynergyFormModule } from '@lib#forms';
14 import { SynergyLayoutModule } from '@lib#layouts';
15 import { SysFrmDsg } from '@wms/app-synergy/app-synergy/sys/frm/api';
16 import { SysFrmList } from '@wms/app-synergy/app-synergy/sys/frm/api';
17 import { WmsThreeDimensionalViewConfigurationComponent } from '../../../../wms-three-dimensional-configuration/wms-three-dimensional-configuration.compo...
18 import { WmsVisualizationEditPanelComponent } from '../../../../wms-three-dimensional-configuration/wms-visualization-edit-panel/wms-visualization-e...
19 import { WmsThreeDimensionalVisualizationComponent, WmsThreeDimensionalVisualizationModule } from '../../../../wms-three-dimensional-visualizatio...
20 import { WmsThreeDimensionalViewFormComponent } from '../../../../components/wms-three-dimensional-views-form/component';
21 import { WmsThreeDimensionalViewListComponent } from '../../../../list/wms-three-dimensional-views-list.component';
22 import { WmsThreeDimensionalViewsComponent } from '../../../../wms-three-dimensional-views.component';
23
24 @Injectable()
25 export class WmsThreeDimensionalViewsDesignFormResolver implements Resolve<SysFrmDsg> {
26   constructor(private formDesignService: SynergyFormDesignsService) {}
27   resolve(route: ActivatedRouteSnapshot, state: RouterStateSnapshot): Observable<SysFrmDsg> {
28     return this.formDesignService.loadFormDesign(WmsThreeDimensionalViewFormComponent.FORM_NAME);
29   }
30   // resolve
31 }
32 // WmsThreeDimensionalViewsDesignFormResolver
33
34 const authTable = "WmsVieCfg";
35
36 const routes: Routes = [
37   { path: "", component: WmsThreeDimensionalViewListComponent, canActivateChild: [SynergyRouteModeGuard], children: [
38     { path: "", component: WmsThreeDimensionalViewListComponent, data: { mode: RouteMode.LIST, authTable } },
39     { path: "uid/:view", component: WmsThreeDimensionalViewConfigurationComponent, canActivate: [SynergyLockableComponentGuard], data: { mode: RouteMode.EDIT, authTable } },
40     { path: "uid/:edit", component: WmsThreeDimensionalViewConfigurationComponent, canActivate: [SynergyLockableComponentGuard], data: { mode: RouteMode.EDIT, authTable } },
41     { path: "", redirectTo: "", pathMatch: "full" }
42   ] }
43 ];

```

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.



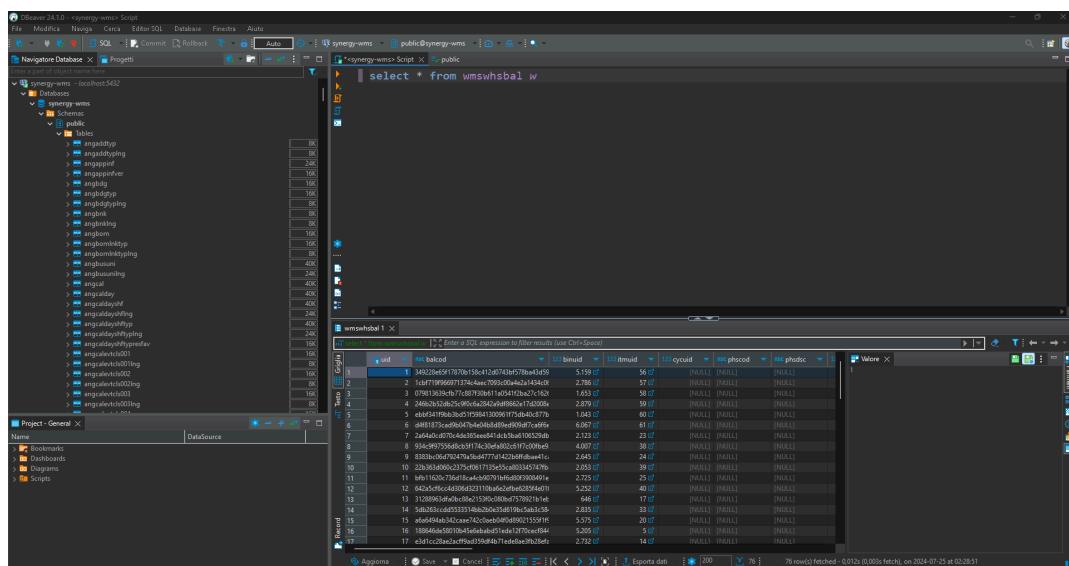
```
public class WmsBalSpecService {  
    @GET  
    @Produces(MediaType.APPLICATION_JSON)  
    public Response getWmsBalMapByItmId (@QueryParam("itmUid") long itmUid, @SynergyRestContext() ExecutionContext context) {  
        try (SedReader rCon = SConnectionFactory.get ().getReadConnection ()) {  
            Map<Long, List<WmsBalMap>> result = WmsBalMapByItmId (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 ("wmsBal/getItmWmsBalDiv") + peterz97  
    @GET  
    @Produces(MediaType.APPLICATION_JSON)  
    public Response getWmsBalDiv (@QueryParam("itmUid") long itmUid, @SynergyRestContext() ExecutionContext context) {  
        try (SedReader rCon = SConnectionFactory.get ().getReadConnection ()) {  
            WmsBalDiv result = WmsBalMapByItmId (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  
    } // getWmsBalDiv  
  
    @Path ("wmsBal/getBalByBinId/{binUid}") + Camaro-Riccardo  
    @GET  
    @Produces(MediaType.APPLICATION_JSON)  
    public Response getWmsBalByBinId (@PathParam("binUid") long binUid, @SynergyRestContext() ExecutionContext context) {  
        try (SedReader rCon = SConnectionFactory.get ().getReadConnection ()) {  
            List<WmsBal> result = WmsBalByBinId (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* relazionali utilizzato dall'azienda per la gestione del *database* 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*, semplificando il processo di verifica.



	id	batch	binuid	bmid	binuid	cyroid	phicod	phicod
1	1	149231ebe51707b156c12d761b378be3d39	5.159	56	[NULL]	[NULL]	[NULL]	[NULL]
2	2	1af7109566971374e7093c04a2a4434d0	2.786	57	[NULL]	[NULL]	[NULL]	[NULL]
3	3	19781305fb7f71d8703611a05412a7c71c62	1.653	58	[NULL]	[NULL]	[NULL]	[NULL]
4	4	149231ebe51707b156c12d761b378be3d39	2.853	59	[NULL]	[NULL]	[NULL]	[NULL]
5	5	149231ebe51707b156c12d761b378be3d39	1.943	60	[NULL]	[NULL]	[NULL]	[NULL]
6	6	149231ebe51707b156c12d761b378be3d39	6.667	61	[NULL]	[NULL]	[NULL]	[NULL]
7	7	2.4a42d0c70c4a62d4e4d6ed9e90ef7e7f9	2.733	23	[NULL]	[NULL]	[NULL]	[NULL]
8	8	149231ebe51707b156c12d761b378be3d39	4.007	37	[NULL]	[NULL]	[NULL]	[NULL]
9	9	149231ebe51707b156c12d761b378be3d39	2.654	38	[NULL]	[NULL]	[NULL]	[NULL]
10	10	2.2b3349e0e274e00115e5c003574e	2.053	39	[NULL]	[NULL]	[NULL]	[NULL]
11	11	1bf11620c756d7e4ca6567916fbfd9303641e	2.725	40	[NULL]	[NULL]	[NULL]	[NULL]
12	12	149231ebe51707b156c12d761b378be3d39	5.320	40	[NULL]	[NULL]	[NULL]	[NULL]
13	13	149231ebe51707b156c12d761b378be3d39	4.557	41	[NULL]	[NULL]	[NULL]	[NULL]
14	14	149231ebe51707b156c12d761b378be3d39	2.835	33	[NULL]	[NULL]	[NULL]	[NULL]
15	15	149231ebe51707b156c12d761b378be3d39	5.575	20	[NULL]	[NULL]	[NULL]	[NULL]
16	16	149231ebe51707b156c12d761b378be3d39	5.205	5	[NULL]	[NULL]	[NULL]	[NULL]
17	17	149231ebe51707b156c12d761b378be3d39	2.732	14	[NULL]	[NULL]	[NULL]	[NULL]

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

## Postman

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

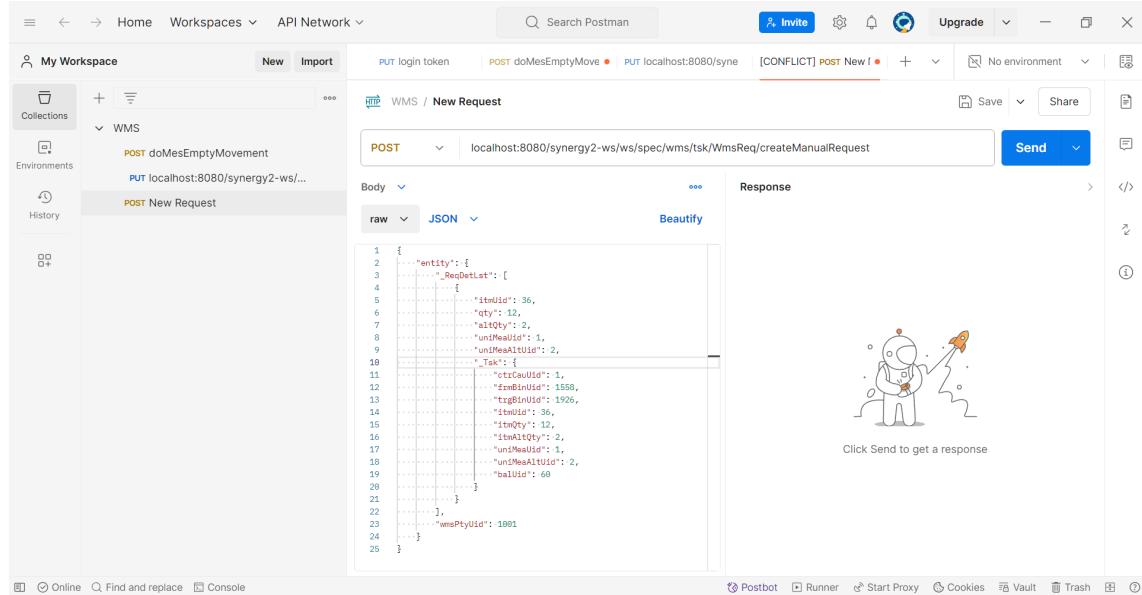


Immagine 15: Esempio di chiamata POST ad un servizio REST con Postman

### 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*:

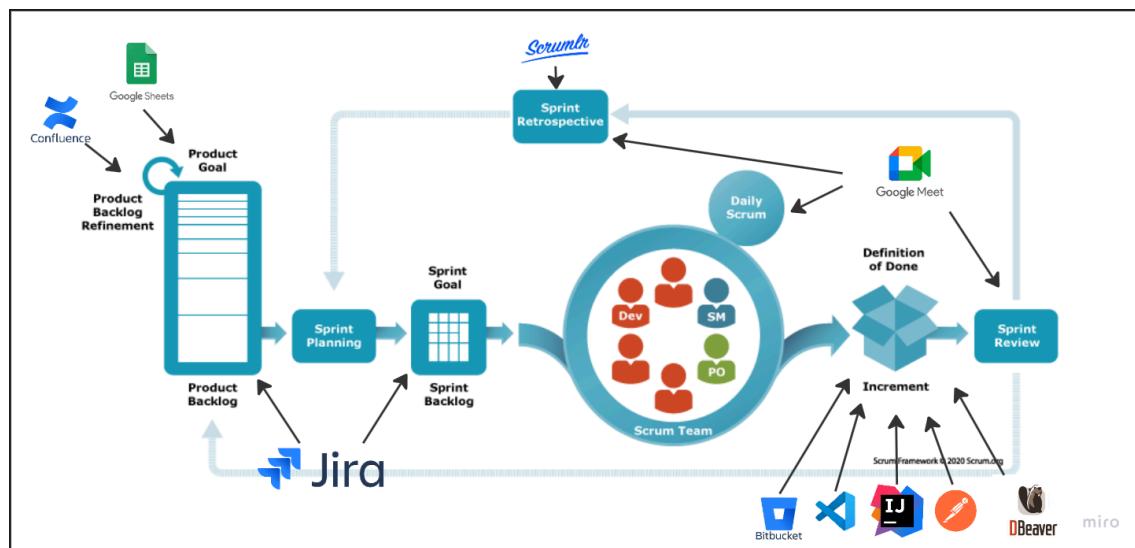


Immagine 16: Come gli strumenti si integrano nel modello di sviluppo aziendale

#### 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.

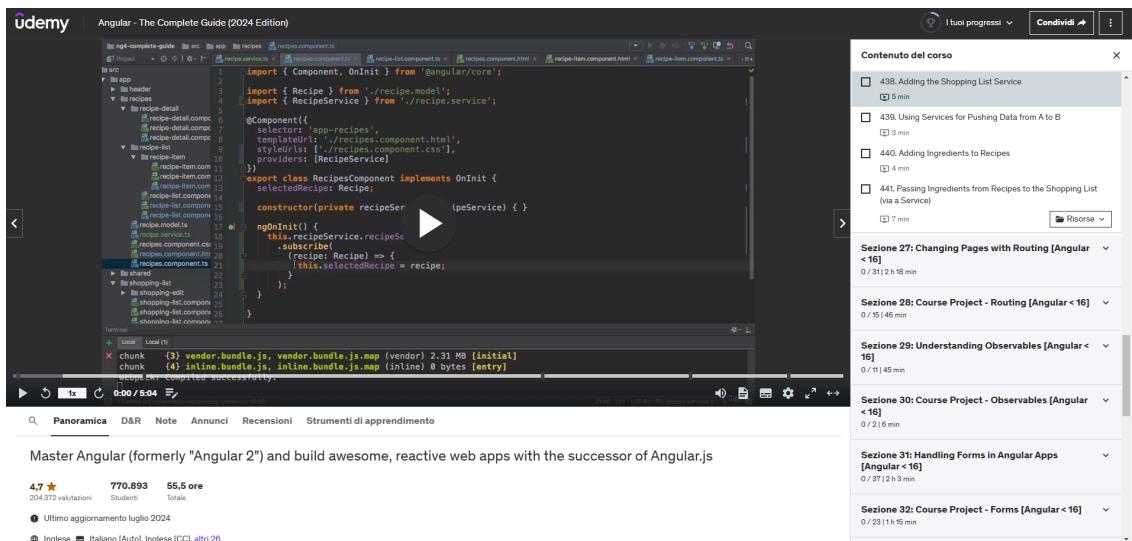


Immagine 17: Corso di formazione Angular su Udemy

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

- 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)	
<b>OP1</b>	Sviluppare competenze con strumenti di comunicazione e collaborazione aziendali come Google Meet e GitHub
<b>OP2</b>	Approfondire l'utilizzo di ITS <sub>G</sub> in un contesto aziendale, come ad esempio Jira <sub>G</sub>
<b>OP3</b>	Partecipare attivamente ai processi di sviluppo <i>software</i> in un contesto aziendale
<b>OP4</b>	Sviluppare competenze con <i>framework</i> ampiamente utilizzati come Angular
<b>OP5</b>	Sviluppare competenze con nuovi linguaggi di programmazione come Java e TypeScript
<b>OP6</b>	Sviluppare codice di qualità tale da essere utilizzabile dall'azienda al termine del mio percorso
<b>OP7</b>	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:**

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

Una descrizione più dettagliata delle tecnologie la presenterò nel [paragrafo 3.4.1](#).

### 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.

## 2.4 Premesse allo svolgimento del tirocinio

### 2.4.1 Approccio al lavoro

Durante il corso di “Ingegneria del *software*” ho avuto modo di comprendere l’importanza di seguire i principi di una metodologia di sviluppo *software* strutturata e organizzata, e, grazie al tirocinio, ho avuto modo di mettere in pratica questi concetti in un contesto aziendale.

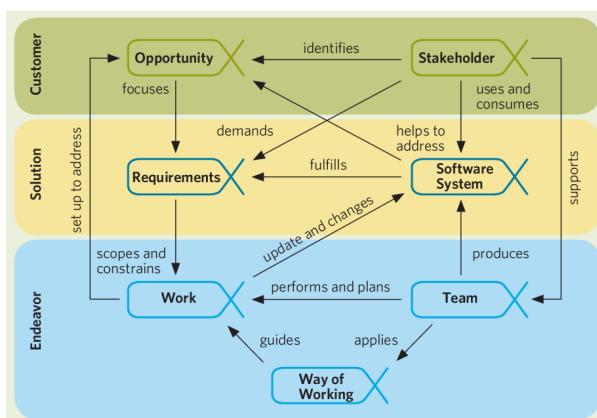


Immagine 19: L’importanza del *Way of Working*<sub>G</sub> nel SEMAT

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

---

Come descritto dal SEMAT ([immagine 19](#)), il *way of working*<sub>G</sub> 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.

#### 2.4.2 Obiettivi di qualità

Ispirandomi ai principi dell'Ingegneria del *software*, ho cercato di garantire efficienza ed efficienza nel conseguimento dei miei obiettivi, perseguiendo qualità e 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 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 [paragrafo 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 3.2.4](#).

#### 2.4.3 Obiettivi di qualità di processo

Durante il mio tirocinio, ho cercato di garantire efficacia ed efficienza nel conseguimento dei miei obiettivi, perseguitando qualità e 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. 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 dal tirocinio.

Il raggiungimento di questi obiettivi è stato possibile grazie alle diverse attività caratterizzanti il modello di sviluppo *Agile<sub>G</sub>* e *Scrum<sub>G</sub>*, come descritto nel [paragrafo 1.4.1](#).

In particolare, grazie alle *Sprint<sub>G</sub> Review<sub>G</sub>* e *Sprint<sub>G</sub> 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 al lavoro.

---

# 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 &amp; 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<sub>G</sub>.

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 20](#):

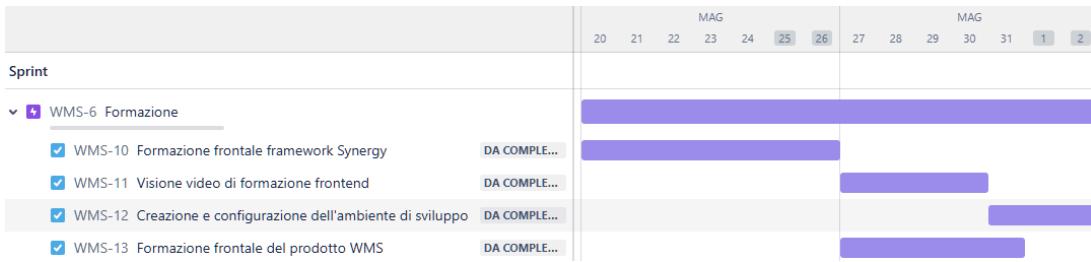


Immagine 20: Diagramma di Gantt delle attività del primo periodo

## 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 21](#):

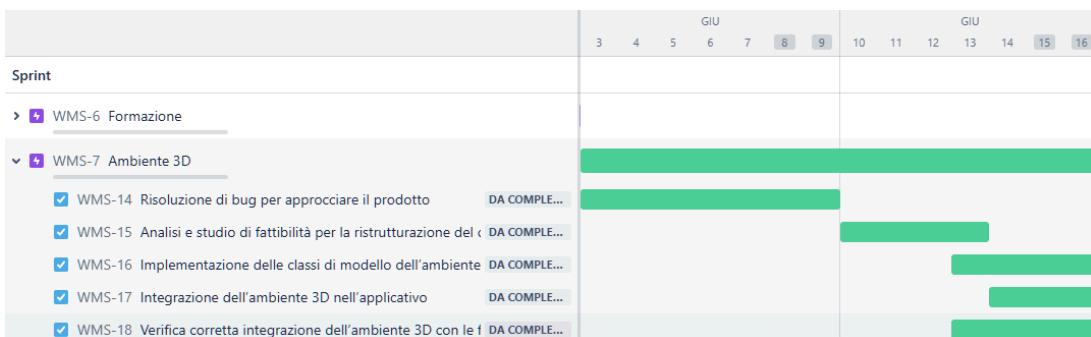


Immagine 21: Diagramma di Gantt delle attività del secondo periodo

---

### 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<sub>G</sub>* (unità di contenimento) sull’interfaccia;
- **Implementazione del *dialog* di creazione dell’ordine:** 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:** analisi collettiva con i membri del *team* delle tabelle necessarie per la creazione dell’ordine di movimentazione;
- **Implementazione dei servizi REST:** 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 22](#):

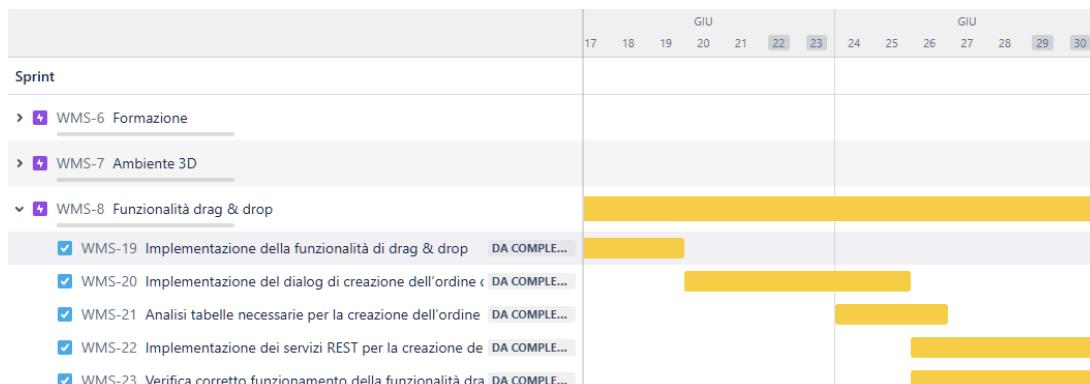


Immagine 22: Diagramma di Gantt delle attività del terzo periodo

---

### 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 23](#):

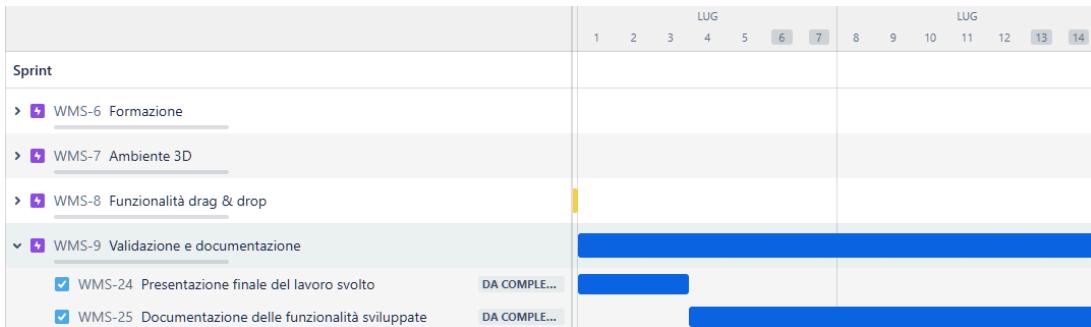


Immagine 23: Diagramma di Gantt delle attività del quarto periodo

Nel complesso, la pianificazione del tirocinio è la seguente:

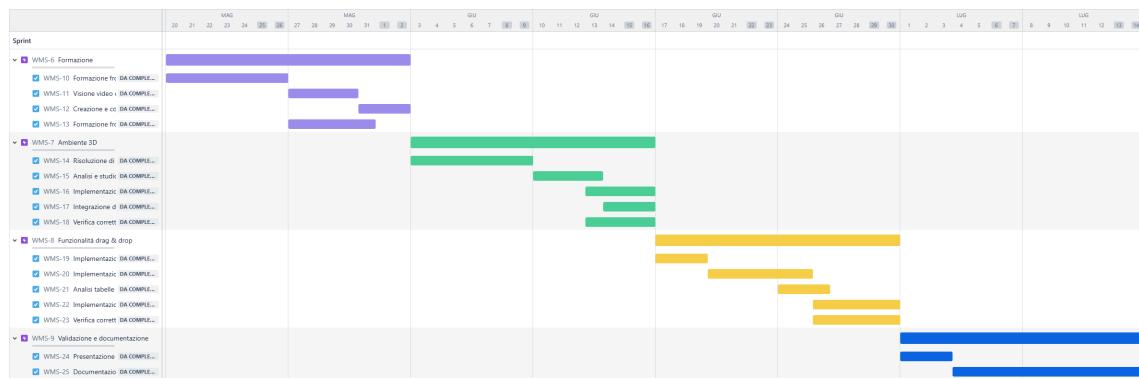


Immagine 24: Diagramma di Gantt complessivo delle attività svolte durante il tirocinio

## 3.2 Metodo di lavoro

### 3.2.1 Way of Working<sub>G</sub>

Come descritto nel [paragrafo 1.4.1](#), l'azienda segue un modello di sviluppo *software Agile<sub>G</sub>*, implementando nello specifico la metodologia *Scrum<sub>G</sub>*. Questo approccio mi ha permesso di lavorare in modo organizzato e strutturato, garantendo un avanzamento costante e un monitoraggio efficace delle attività svolte, prendendo di riferimento i principi precedentemente descritti nel [paragrafo 2.4.1](#).

In particolare, ho partecipato attivamente ai *Daily Standup Meeting* e alle *Sprint<sub>G</sub> Review<sub>G</sub>* e *Sprint<sub>G</sub> 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 organizzato) dove ho

annotato quotidianamente le attività svolte, al fine di rendicontare il mio percorso e svolgere un’analisi critica del lavoro svolto, come mostrato nell’[immagine 25](#):

The screenshot shows a Notion workspace with a sidebar on the left containing sections for Stage, Appunti, and Rapportini. The main area displays a table titled 'Stage' with columns for Name, Date, and Participants. One row is selected, showing 'Introduzione Synergy pt1' with the date 'May 20, 2024' and participant 'Samuele Baldan'. To the right of the table is a detailed view of the selected note, titled 'Introduzione Synergy pt1'. It includes fields for Date (May 20, 2024), Participant (Samuele Baldan), and a 'Comments' section. Below the note card is a 'Service layer' section with a brief description of Synergy's architecture.

Immagine 25: Bacheca personale su Notion

### 3.2.2 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* utile e tempestivo 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](#).

### 3.2.3 Revisioni di progresso

Le revisioni di progresso sono state fondamentali per monitorare lo stado di avanzamento e per ottenere *feedback* valido sulle attività svolte.

Come menzionato nel [paragrafo 1.4.1](#), ho avuto modo di partecipare attivamente a diverse attività di revisione<sub>G</sub>, dalle giornaliere durante i *daily standup meeting*, alle revisioni di fine Sprint<sub>G</sub> (*Sprint<sub>G</sub> review<sub>G</sub>* e *Sprint<sub>G</sub> retrospective*), fino alla revisione<sub>G</sub> 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 nel contesto lavorativo sia dal punto di vista tecnico che umano.

### 3.2.4 Strumenti di verifica

Al fine di perseguire gli obiettivi di qualità indicati nel [paragrafo 2.4.2](#), 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*<sub>G</sub> e *continuous deployment*, garantendo un monitoraggio costante del codice e delle funzionalità sviluppate, al fine di accettare all'interno del *repository*<sub>G</sub> un prodotto sempre funzionante e conforme alle aspettative.

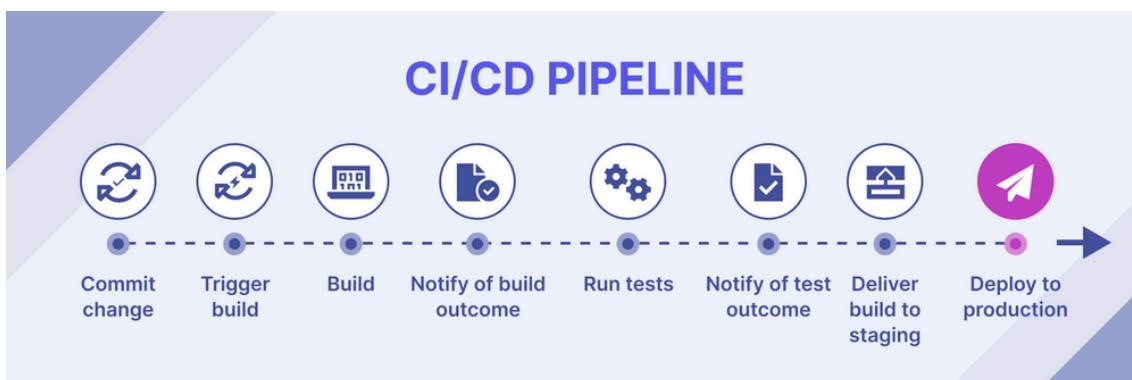


Immagine 26: Pipeline di *Continuous Integration*<sub>G</sub> e *Continuous Deployment*

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*<sub>G</sub>, si procede con la *build* del progetto e successivamente vengono eseguiti i *test* automatici. Solo qualora tutta la *pipeline* venga eseguita con successo, allora la *pull request*<sub>G</sub> viene accettata e il codice viene integrato nel *repository*<sub>G</sub>. In caso contrario, la *pull request*<sub>G</sub> 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* automatici:
  - **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;

---

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

- **Test manuali:** *test* svolti manualmente. Durante il mio percorso ho implementato 2 principali tipologie di *test* manuali:
  - **Test di sistema:** *test* che verificano il corretto funzionamento del sistema nel suo complesso, garantendo che tutte le funzionalità siano conformi alle aspettative;
  - **Test prestazionali:** *test* che verificano le prestazioni del sistema, garantendo che il prodotto risponda ai requisiti prestazionali richiesti.
- **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 prestazionali 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 BitBucket 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*<sub>G</sub> 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*<sub>G</sub>.

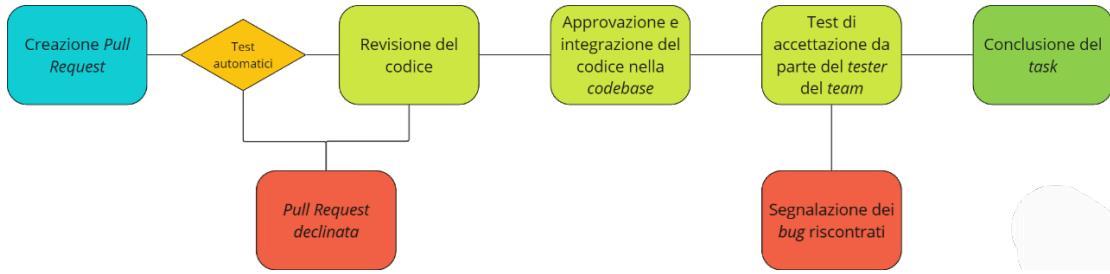


Immagine 27: Pipeline per l'accettazione di una *Pull Request<sub>G</sub>*

Il tracciamento pertanto delle modifiche apportate e il versionamento 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<sub>G</sub>* di Jira<sub>G</sub> corrispondente, identificata da un codice univoco così strutturato:

WMS - XX

dove:

- **WMS**: identifica il progetto WMS;
- **XX**: identifica il numero progressivo della *issue<sub>G</sub>*.

Ogni *task<sub>G</sub>* ha il riferimento all’assegnatario, al tempo stimato per il completamento e al *branch<sub>G</sub>* di riferimento. Ad ogni *pull request<sub>G</sub>* veniva associato invece il membro del *team* che avrebbe dovuto effettuare la revisione<sub>G</sub> 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.

### 3.2.5 Resoconti

Durante l’intero percorso mi sono impegnato a tenere documentata l’esperienza, sia mediante dei resoconti giornalieri su una bacheca personale su Notion ([immagine 25](#)), sia mediante dei resoconti settimanali inviati al relatore. Questo approccio mi ha permesso di avere una traccia costante del mio percorso e di analizzare criticamente le attività svolte, garantendo un monitoraggio costante dell’avanzamento del progetto e un *feedback* costante.

Infatti, oltre ad agevolare la scrittura del documento di tesi, i resoconti mi hanno permesso di avere uno strumento di autovalutazione costante del progresso ottenuto, rendendo più chiaro il percorso svolto e le attività completate.

I resoconti settimanali, d’altro canto, mi hanno permesso di allineare regolarmente il mio relatore sullo stato di avanzamento dello *stage*, indicando gli obi-

---

ettivi prefissati, gli obiettivi raggiunti e descrivendo le attività svolte durante la settimana.

## 3.3 Analisi dei requisiti

### 3.3.1 Casi d'uso

I casi d'uso (*use case*) sono uno strumento utile nel rappresentare le interazioni tra l'utente e l'applicativo. Un diagramma dei casi d'uso illustra graficamente queste interazioni, evidenziando gli attori coinvolti e le funzionalità del sistema.

Nel contesto di questo tirocinio, ho adottato la seguente convenzione per la descrizione dei casi d'uso:

UC-X.Y.Z: nome del caso d'uso

dove:

- **X**: intero positivo che rappresenta il numero del caso d'uso;
- **Y e Z**: interi positivi che rappresentano eventuali sottocasi d'uso.

Ogni caso d'uso evidenzia inoltre le seguenti proprietà:

- **Descrizione**: una breve descrizione del caso d'uso;
- **Attore primario**: l'attore principale coinvolto nel caso d'uso;
- **Precondizioni**: le condizioni necessarie affinché il caso d'uso possa essere eseguito;
- **Postcondizioni**: le condizioni che devono essere soddisfatte al termine dell'esecuzione del caso d'uso;
- **Estensioni o inclusioni (se presenti)**: scenari alternativi che possono verificarsi durante l'esecuzione del caso d'uso;
- **Generalizzazioni (se presenti)**: casi d'uso che rappresentano una generalizzazione del caso d'uso principale.

Nel contesto specifico di questo sistema, l'attore principale è l'operatore di *back-office*. Questa figura rappresenta un utente dotato di completa autonomia e competenza nell'utilizzo del prodotto WMS. L'operatore di *backoffice* è in grado di utilizzare tutte le funzionalità offerte dal sistema, essendo una figura amministrativa che gestisce i diversi aspetti del magazzino, inclusa la creazione degli ordini di movimentazione.

È importante notare che gli *use case* presentati in questa sezione si concentrano sui casi principali, al fine di non appesantire la descrizione con dettagli superflui o comportamenti banali dell'applicativo: nel caso dell'ambiente tridimensionale infatti, ho mantenuto il focus sulla visualizzazione delle strutture e l'interazione con esse, aspetti essenziali per la funzionalità di creazione degli ordini di movimentazione.

## Ambiente tridimensionale:

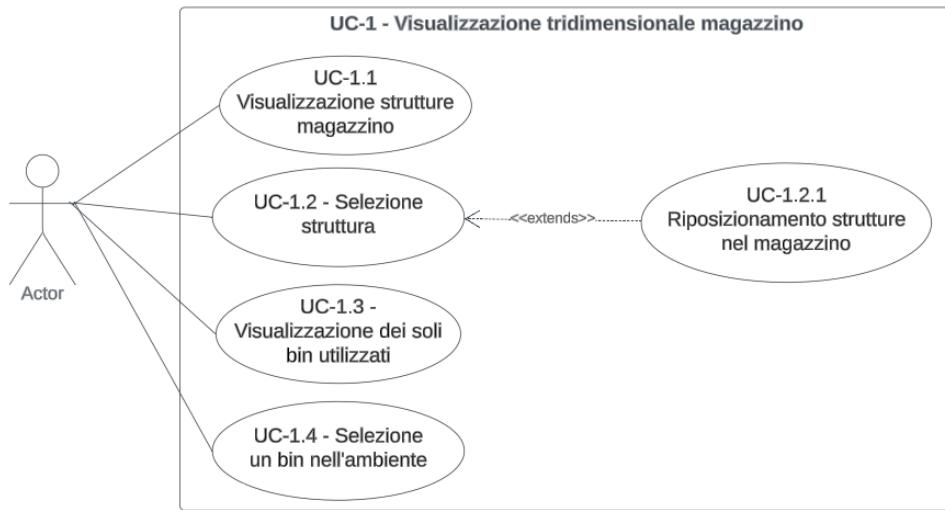


Immagine 28: Casi d'uso per l'ambiente tridimensionale

- **Nome:** Visualizzazione tridimensionale magazzino;
- **Attore primario:** Utente;
- **Precondizioni:** L'ambiente 3D deve essere correttamente caricato e configurato;
- **Postcondizioni:** L'ambiente 3D viene visualizzato correttamente;

### UC-1.1 Visualizzazione strutture magazzino

L'utente visualizza le strutture del magazzino all'interno dell'ambiente tridimensionale. L'utente deve poter navigare nell'ambiente, avendo una visione chiara del posizionamento delle strutture del magazzino.

- **Attore primario:** Utente;
- **Precondizioni:** L'ambiente 3D deve essere correttamente caricato e configurato;
- **Postcondizioni:** Le strutture del magazzino viene visualizzata correttamente.

### UC-1.2 Selezione struttura

L'utente seleziona una struttura del magazzino all'interno dell'ambiente tridimensionale, avendo modo di modificarla, riposizionarla o visualizzarne i dettagli.

- **Attore primario:** Utente;
- **Precondizioni:** L'ambiente 3D deve essere correttamente caricato e configurato;
- **Postcondizioni:** La struttura selezionata viene evidenziata, cambiandone il colore in base al tema dell'applicativo.
- **Estensioni:**
  - **UC-1.2.1:** Riposizionamento strutture magazzino;

---

### **UC-1.2.1 Riposizionamento della struttura nel magazzino**

L’utente riposiziona la struttura del magazzino all’interno dell’ambiente tridimensionale.

- **Attore primario:** Utente;
- **Precondizioni:** L’ambiente 3D deve essere correttamente caricato e configurato;
- **Postcondizioni:** La struttura del magazzino vengono riposizionate correttamente.

### **UC-1.3 Visualizzazione dei soli $bin_G$ utilizzati**

L’utente visualizza solo i  $bin_G$  utilizzati all’interno dell’ambiente tridimensionale, ossia i  $bin_G$  contenenti almeno un saldo al loro interno.

- **Attore primario:** Utente;
- **Precondizioni:** L’ambiente 3D deve essere correttamente caricato e configurato;
- **Postcondizioni:** Vengono visualizzati solo i  $bin_G$  utilizzati all’interno dell’ambiente tridimensionale, nascondendo i  $bin_G$  vuoti.

### **UC-1.4 Selezione $bin_G$ nell’ambiente**

L’utente seleziona un  $bin_G$  all’interno dell’ambiente tridimensionale, avendo modo di visualizzare i saldi presenti al suo interno.

- **Attore primario:** Utente;
- **Precondizioni:** L’ambiente 3D deve essere correttamente caricato e configurato;
- **Postcondizioni:** Il  $bin_G$  selezionato viene evidenziato, cambiandone il colore in base al tema dell’applicativo.

## **Creazione dell’ordine di movimentazione**

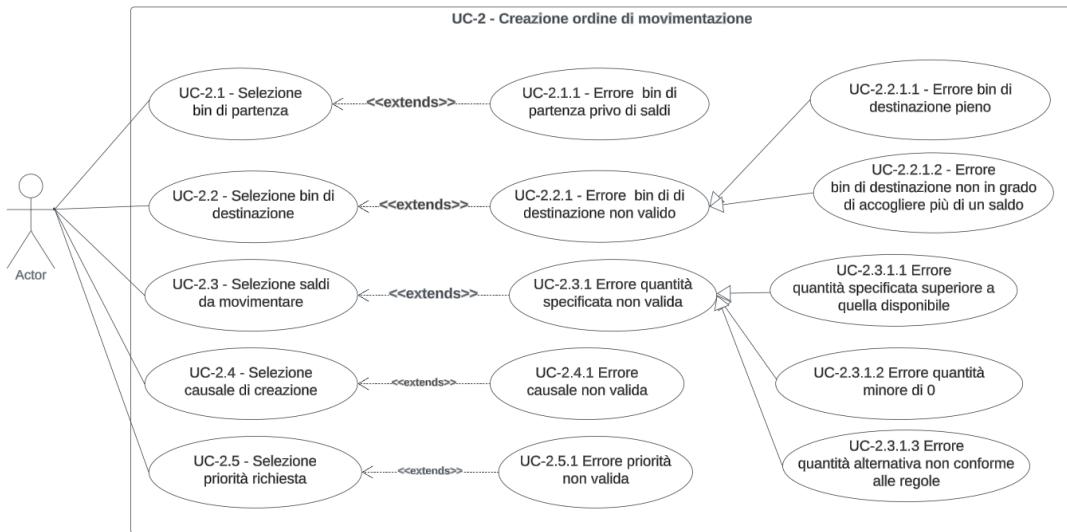


Immagine 29: Casi d'uso per la creazione dell'ordine di movimentazione

- **Nome:** Creazione dell'ordine di movimentazione;
- **Attore primario:** Utente;
- **Precondizioni:**
  - L'ambiente 3D deve essere correttamente caricato e configurato;
  - Deve essere presente almeno un  $bin_G$  di partenza valido da cui iniziare l'operazione di *dragging*;
  - Deve essere presente almeno un  $bin_G$  valido da utilizzare come  $bin_G$  di destinazione diverso dal  $bin_G$  di partenza;
  - Il  $bin_G$  di destinazione deve essere in grado di accogliere i saldi che si intendono movimentare;
  - In caso si movimenti più di un saldo, il  $bin_G$  di destinazione deve essere in grado di accogliere più saldi differenti.
- **Postcondizioni:**
  - Viene generata una richiesta di ordine di movimentazione che sarà preso in carico dagli operatori in magazzino.

### **UC-2.1 Selezione $bin_G$ di partenza**

L'utente seleziona il  $bin_G$  di partenza da cui iniziare l'operazione di *drag&drop*.

- **Attore primario:** Utente;
- **Precondizioni:** L'ambiente 3D deve essere correttamente caricato e configurato;
- **Postcondizioni:** Il  $bin_G$  di partenza viene “Selezionato”, cambiandone il colore in base al tema dell'applicativo.
- **Esezioni:**
  - **UC-2.1.1:** Errore  $bin_G$  di partenza privo di saldi;

---

## **UC-2.2 Selezione $bin_G$ di destinazione**

L'utente seleziona il  $bin_G$  di destinazione in cui posizionare i saldi movimentati.

- **Attore primario:** Utente;
- **Precondizioni:** L'ambiente 3D deve essere correttamente caricato e configurato;
- **Postcondizioni:** Il  $bin_G$  di destinazione viene “Evidenziato”, cambiandone il colore in base al tema dell'applicativo.
- **Esezioni:**
  - UC-2.2.1: Errore  $bin_G$  di destinazione non valido;

### **UC-2.2.1 Errore $bin_G$ di destinazione non valido**

L'utente seleziona un  $bin_G$  di destinazione non valido, e pertanto non è possibile procedere con la selezione dei saldi da movimentare.

- **Attore primario:** Utente;
- **Precondizioni:** Il  $bin_G$  di destinazione selezionato deve essere non valido;
- **Postcondizioni:** Il  $bin_G$  viene resettato, tornando allo stato iniziale.
- **Generalizzazioni:**
  - UC-2.2.1.1:  $bin_G$  di destinazione completo;
  - UC-2.2.1.2:  $bin_G$  di destinazione non in grado di accogliere i saldi movimentati;

## **UC-2.3 Selezione saldi da movimentare**

L'utente seleziona i saldi da movimentare dal  $bin_G$  di partenza al  $bin_G$  di destinazione.

- **Attore primario:** Utente;
- **Precondizioni:** Il  $bin_G$  di partenza e il  $bin_G$  di destinazione devono essere specificati e validi;
- **Postcondizioni:** Viene mostrato il resoconto di quali saldi si vuole movimentare e in che quantità, e la quantità rimanente del saldo dal  $bin_G$  di partenza.
- **Esezioni:**
  - UC-2.3.1: Errore quantità selezionata non valida;

## **UC-2.4 Specifica della causale di creazione**

L'utente specifica la causale per la creazione dell'ordine di movimentazione.

- **Attore primario:** Utente;
- **Precondizioni:** Il  $bin_G$  di partenza e il  $bin_G$  di destinazione devono essere specificati e validi;
- **Postcondizioni:** La causale di creazione viene specificata e associata all'ordine di movimentazione.

- 
- **Estensioni:**
    - UC-2.4.1: Causale non valida;

### UC-2.5 Specifica della priorità della richiesta

L'utente specifica la priorità per la creazione dell'ordine di movimentazione.

- **Attore primario:** Utente;
- **Precondizioni:** Il  $bin_G$  di partenza e il  $bin_G$  di destinazione devono essere specificati e validi;
- **Postcondizioni:** La priorità della richiesta viene specificata e associata all'ordine di movimentazione.
- **Estensioni:**
  - UC-2.5.1: Errore priorità non valida;

### 3.3.2 Tracciamento dei requisiti

Ho tracciato i requisiti con un codice identificativo definito nel seguente modo:

TC-I

dove:

- **T** rappresenta la tipologia. Può assumere i valori:
  - **F:** funzionale;
  - **Q:** di qualità;
  - **P:** prestazionale;
  - **V:** di vincolo.
- **C** rappresenta la classificazione. Può assumere i valori:
  - **M:** *mandatory*, obbligatorio;
  - **D:** desiderabile;
  - **O:** opzionale.
- **I:** intero positivo identificativo del requisito<sub>G</sub>.

#### Requisiti funzionali:

Codice	Classe	Descrizione	Fonte
F-M1	Obbligatorio	L'utente può visualizzare le strutture del magazzino all'interno dell'ambiente tridimensionale.	UC-1.1 Obiettivi aziendali

F-M2	Obbligatorio	L'utente può selezionare una struttura del magazzino all'interno dell'ambiente tridimensionale.	UC-1.2 Obiettivi aziendali
F-M3	Obbligatorio	L'utente può riposizionare la struttura del magazzino all'interno dell'ambiente tridimensionale.	UC-1.2.1 Obiettivi aziendali
F-M4	Obbligatorio	L'utente può visualizzare solo i <i>bin<sub>G</sub></i> utilizzati all'interno dell'ambiente tridimensionale.	UC-1.3 Obiettivi aziendali
F-M5	Obbligatorio	L'utente può selezionare un <i>bin<sub>G</sub></i> all'interno dell'ambiente tridimensionale.	UC-1.4 Obiettivi aziendali
F-M6	Obbligatorio	L'utente può selezionare il <i>bin<sub>G</sub></i> di partenza da cui iniziare l'operazione di <i>drag&amp;drop</i> .	UC-2.1 Obiettivi aziendali
F-M7	Obbligatorio	L'utente può selezionare il <i>bin<sub>G</sub></i> di destinazione in cui posizionare i saldi movimentati.	UC-2.2 Obiettivi aziendali
F-M8	Obbligatorio	L'utente può selezionare i saldi da movimentare dal <i>bin<sub>G</sub></i> di partenza al <i>bin<sub>G</sub></i> di destinazione.	UC-2.3 Obiettivi aziendali
F-M9	Obbligatorio	L'utente può specificare la causale per la creazione dell'ordine di movimentazione.	UC-2.4 Obiettivi aziendali
F-M10	Obbligatorio	L'utente può specificare la priorità per la creazione dell'ordine di movimentazione.	UC-2.5 Obiettivi aziendali
F-D11	Desiderabile	L'utente deve visualizzare un errore se il <i>bin<sub>G</sub></i> di partenza è privo di saldi.	UC-2.1.1
F-D12	Desiderabile	L'utente deve visualizzare un errore se il <i>bin<sub>G</sub></i> di destinazione non è valido.	UC-2.2.1
F-D13	Desiderabile	L'utente deve visualizzare un errore se la quantità selezionata non è valida.	UC-2.3.1

F-D14	Desiderabile	L'utente deve visualizzare un errore se la causale specificata non è valida.	UC-2.4.1
F-D15	Desiderabile	L'utente deve visualizzare un errore se la priorità specificata non è valida.	UC-2.5.1

Tabella 5: Requisiti funzionali

### Requisiti di qualità:

Codice	Classe	Descrizione	Fonte
Q-M1	Obbligatorio	Il prodotto deve rispettare le convenzioni aziendali.	Azienda
Q-O2	Opzionale	Devono essere consegnati i diagrammi UML <sub>G</sub> delle classi implementate.	Obiettivi aziendali
Q-O3	Opzionale	Devono essere consegnata la documentazione delle funzionalità implementate.	Obiettivi aziendali
Q-O4	Opzionale	Devono essere consegnata la documentazione dei servizi REST implementati.	Obiettivi aziendali

Tabella 6: Requisiti di qualità

### Requisiti prestazionali:

Codice	Classe	Descrizione	Fonte
P-M1	Obbligatorio	Il tempo di caricamento dell'ambiente 3D a seguito del refactoring deve rimanere sotto i 4 secondi	Azienda

Tabella 7: Requisiti prestazionali

### Requisiti di vincolo:

Codice	Classe	Descrizione	Fonte
V-M1	Obbligatorio	Il prodotto deve essere sviluppato in Angular 17.	Azienda

V-M2	Obbligatorio	Il prodotto deve essere sviluppato in Java 21 seguendo le convenzioni imposte dal <i>framework</i> Synergy.	Azienda
V-M3	Obbligatorio	L'ambiente tridimensionale deve essere sviluppato in Three.js.	Azienda
V-M4	Obbligatorio	Il browser utilizzato per accedere al prodotto deve supportare WebGL 2.0	V-M3

Tabella 8: Requisiti di vincolo

### Riepilogo requisiti:

Tipologia	Obbligatori	Desiderabili	Opzionali	Totale
Funzionali	10	5	0	15
Qualità	1	0	3	4
Prestazionali	1	0	0	1
Vincolo	4	0	0	4

Tabella 9: Riepilogo requisiti

## 3.4 Progettazione

### 3.4.1 Tecnologie utilizzate

Come stabilito dai vincoli di progetto ([paragrafo 2.2.3.2](#)), trattandosi di un'estensione delle funzionalità di un prodotto esistente, ho utilizzato le tecnologie e gli strumenti già in uso dall'azienda, garantendo un'interoperabilità e una coerenza con il prodotto esistente.

Nel dettaglio, le tecnologie utilizzate sono le seguenti:

- **Frontend:**

- **Angular:** *framework* TypeScript per lo sviluppo di applicazioni *web*. Ho utilizzato Angular per la creazione dell'interfaccia utente e per la gestione delle funzionalità *frontend* del prodotto WMS; si tratta di un *framework* che già conoscevo, ma che non avevo mai utilizzato in un contesto professionale, e che mi ha permesso di approfondire le mie conoscenze e di metterle in pratica in un contesto aziendale.

---

Angular si basa su un'architettura *component-based*, che permette di creare componenti riutilizzabili e modulabili, garantendo una struttura chiara e ben organizzata del codice.

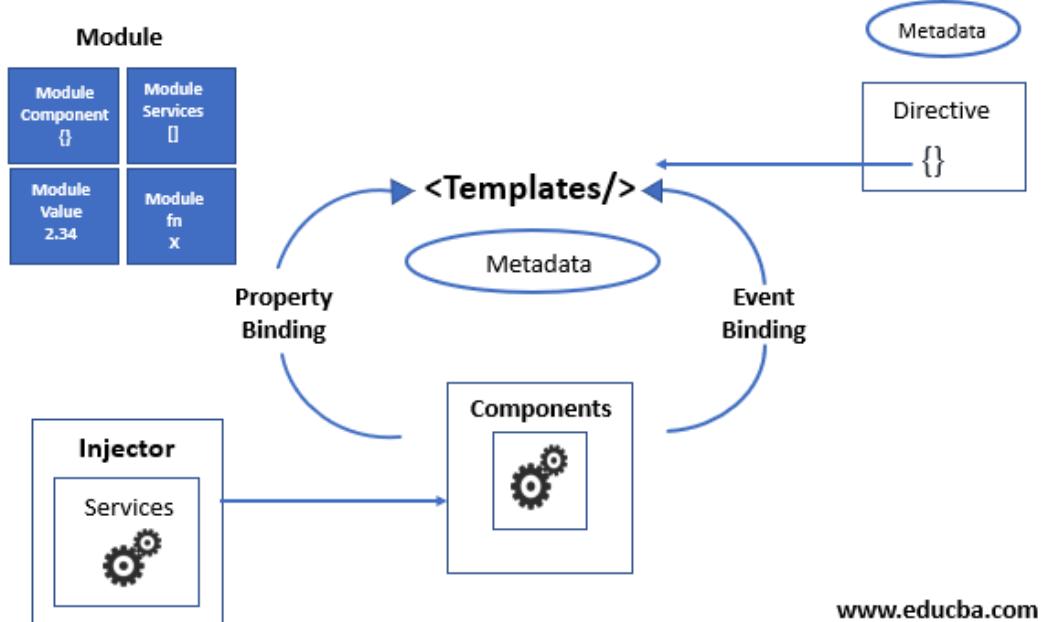


Immagine 30: Struttura Angular

Fonte: <https://www.educba.com/how-angular-works/>

Come mostrato nell'[immagine 30](#), ogni componente fa riferimento da un *template* HTML, a cui sono associati le proprietà e gli eventi, assumendo un comportamento dinamico e interattivo. Un ruolo cruciale è svolto dall'*injector* (secondo il pattern *dependency injection*), che permette di iniettare le dipendenze necessarie per il funzionamento del componente, in particolar modo dei servizi.

- **Three.js**: libreria JavaScript per la creazione di ambienti 3D. Ho utilizzato Three.js per la creazione dell'ambiente tridimensionale, garantendo una visualizzazione realistica e interattiva dell'ambiente di lavoro.

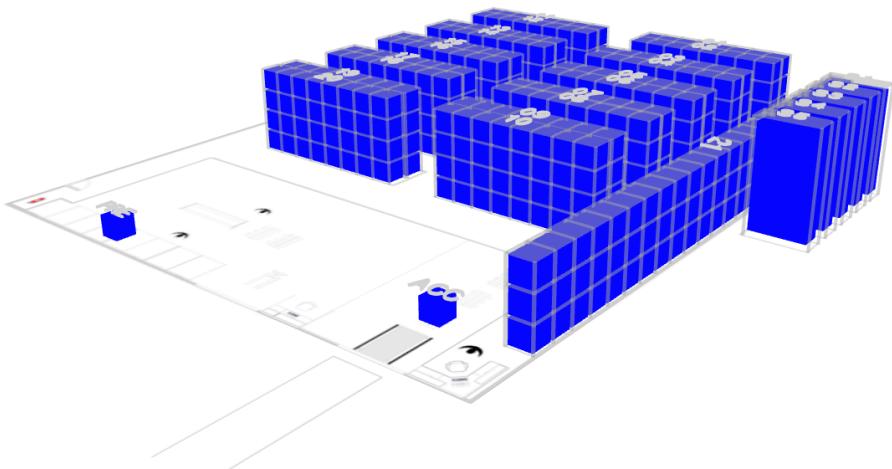


Immagine 31: Ambiente 3D realizzato durante il tirocinio con Three.js

Sviluppare un ambiente 3D presenta delle sfide specifiche, in particolar modo riguardo alle *performance* e alla gestione delle risorse, che durante il mio percorso ho dovuto affrontare e risolvere, al fine di garantire un prodotto fluido e reattivo.

- **Backend:**

- **Java con framework Synergy:** nello sviluppo del lato *backend* dell'applicativo ho utilizzato il *framework* proprietario Synergy sviluppato in Java. Il suo comportamento è similare al *framework* Spring, ed entrambi si occupano di semplificare lo sviluppo e la configurazione delle applicazioni, permettendo agli sviluppatori di concentrarsi sulla logica di *business*.

Synergy risulta essere versatile e in grado di predisporre un ambiente di sviluppo completo, curando gli aspetti relativi al *testing*, alla *build*, alla gestione delle dipendenze e la comunicazione con il *database*.

- **Database:**

- **PostgreSQL:** *database* relazionale utilizzato per la memorizzazione dei dati dell'applicativo. PostgreSQL è un *database* relazionale che garantisce un'alta affidabilità e un'ottima scalabilità, garantendo un ambiente di lavoro stabile e performante anche in presenza di un grande volume di dati.

Il database è stato utilizzato per la memorizzazione dei dati relativi all'ambiente di tridimensionale e alla creazione degli ordini di movimentazione.

Nel complesso le tecnologie si integrano come mostro nell'[immagine 32](#):

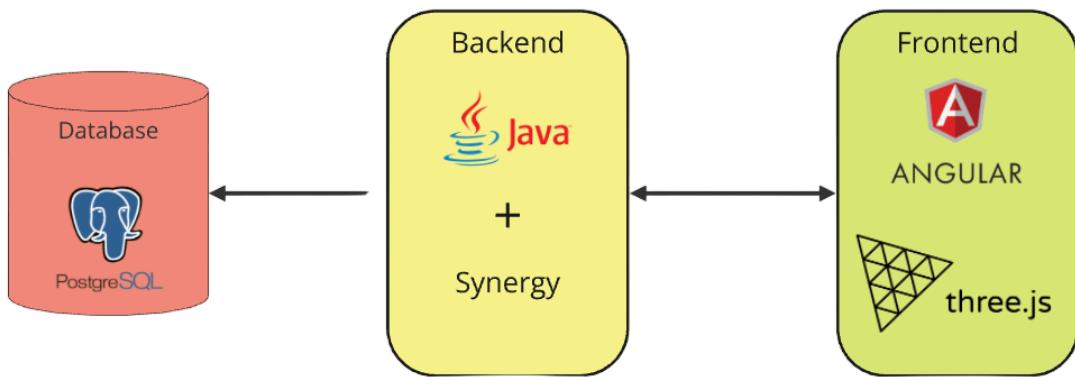


Immagine 32: Integrazione delle tecnologie utilizzate

Le tecnologie utilizzate mi hanno permesso di sviluppare un prodotto conforme alle aspettative e alle esigenze dell'azienda, garantendo che quanto prodotto fosse integrabile e interoperabile con il prodotto esistente.

### 3.4.2 *Workspace e widget*

Per poter comprendere come ho implementato l'ambiente tridimensionale, è necessario che siano chiari i concetti di *workspace* e di *widget*:

- **Widget:** rappresenta un componente autonomo all'interno del *workspace*, che può essere configurato e personalizzato in base alle esigenze dell'utente. Ogni *widget* è responsabile di una specifica funzionalità, e può comunicare con gli altri *widget* presenti nel *workspace*. L'ambiente tridimensionale è di fatto un *widget* all'interno del *workspace*, che permette di visualizzare le strutture del magazzino e i saldi presenti al suo interno;
- **Workspace:** rappresenta l'ambiente di lavoro dell'operatore di *backoffice*, in cui vengono visualizzati i diversi *widget*, personalizzandone dimensione e posizione. L'operatore può configurare il *workspace* in base alle proprie esigenze, organizzando i *widget* in modo da avere una visione chiara e completa delle informazioni necessarie per svolgere le proprie attività.

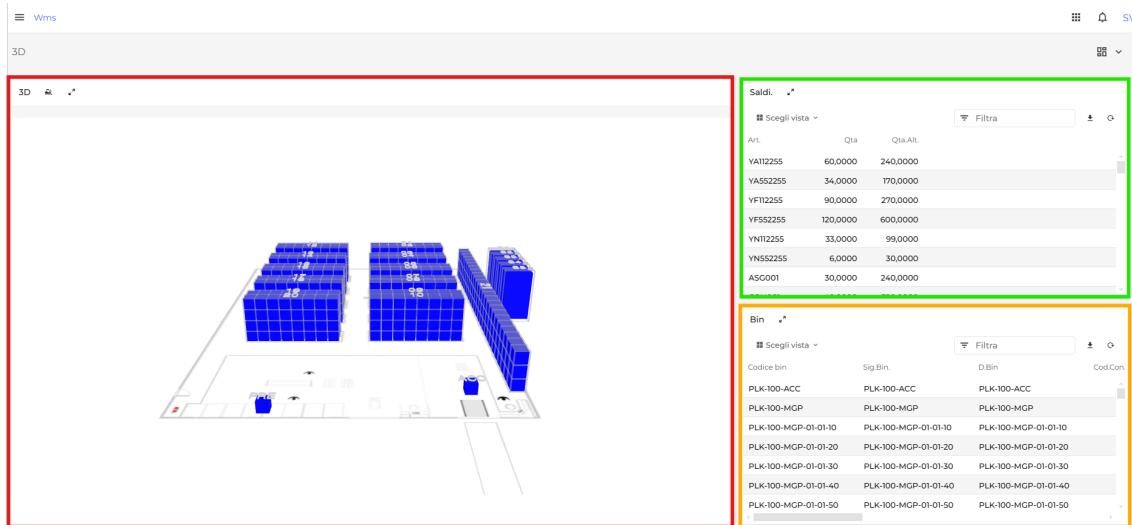


Immagine 33: Esempio dei *widget* presenti nel *workspace*

Nell'[immagine 33](#) mostro un esempio dei *widget* presenti nel *workspace*, tra cui l'ambiente tridimensionale, la lista dei saldi presenti nel magazzino e la lista dei *bin<sub>G</sub>* disponibili.

### 3.4.3 Progettazione dell'ambiente tridimensionale

L'applicativo WMS inizialmente prevedeva una gestione dell'ambiente 3D basata sull'istanziazione di un'unica *mesh* globale, ossia un'unica istanza grafica rappresentante tutte le strutture del magazzino: tale approccio, seppur funzionante, risultava potenzialmente poco flessibile nel caso di future estensioni in merito al comportamento di aree, strutture e *bin<sub>G</sub>*. Infatti, al momento dell'interazione, non si andava direttamente ad interagire con l'oggetto della logica corrispondente, bensì con la *mesh* globale, che si occupava successivamente, mediante l'indice dell'istanza selezionata, di definire quale *bin<sub>G</sub>* fosse stato selezionato. L'immagine seguente rappresenta graficamente il cambiamento apportato:

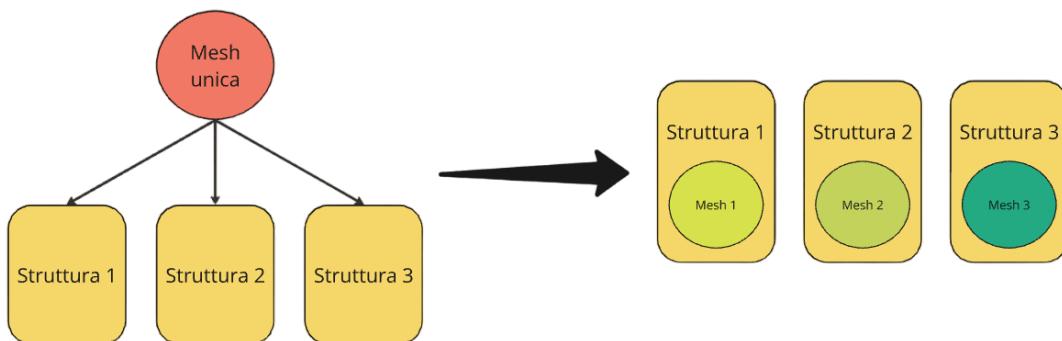


Immagine 34: Cambiamento apportato durante il *refactor* dell'ambiente 3D

La ristrutturazione dell'ambiente 3D, pertanto, si pone come obiettivo la creazione di un sistema di gestione più flessibile e scalabile, che permetta di interagire di-

---

rettamente con gli oggetti dell’ambiente 3D, avendo un riferimento concreto al bin<sub>G</sub>, alla struttura o all’area selezionata.

Durante l’implementazione ho lavorato con diversi *design pattern* derivanti dall’utilizzo di Angular e Synergy, con l’obiettivo di produrre un codice manutenibile e scalabile. Infatti, grazie al pattern *dependency injection* di Angular, mi sono occupato dell’aggiornamento dei servizi necessari alla rappresentazione dell’ambiente 3D, mantenendo una struttura del codice chiara e ben organizzata: infatti questo paradigma consente di incorporare le dipendenze richieste direttamente nel processo di costruzione degli oggetti, semplificando notevolmente la gestione e l’utilizzo dei servizi all’interno del codice sorgente.

In sinergia con esso ho utilizzato il *pattern decorator*: Angular si distingue per l’adozione sistematica del concetto di decoratori, come ad esempio l’annotazione `@Injectable()`. Tali costrutti consentono di arricchire dinamicamente le funzionalità degli oggetti esistenti, ampliandone le capacità senza introdurre eccessivi vincoli di dipendenza tra le varie componenti del sistema. Un altro esempio concreto è il decoratore `@ExhaustingEvent()`, che ho utilizzato per la gestione degli eventi “iscrivibili” nell’ambito degli *Observable* per le chiamate asincrone dei servizi REST: in questo modo, tali eventi vengono gestiti in modo trasparente da Synergy, che si occuperà automaticamente delle operazioni di *subscribe* e *unsubscribe*. Il comportamento di tali *pattern* lo mostro nella seguente immagine:

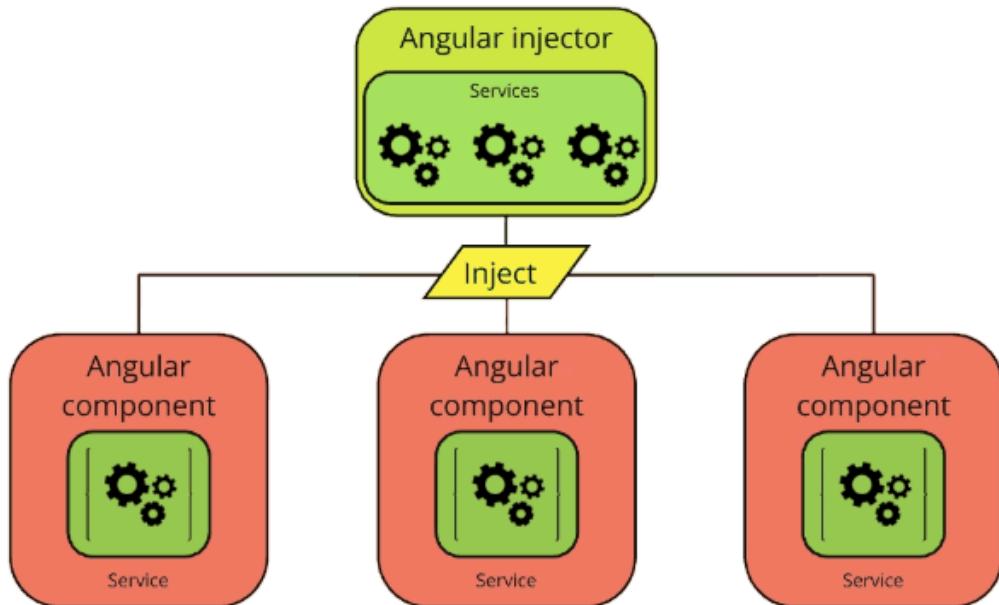


Immagine 35: Comportamento del *dependency injector* in Angular

### **Mesh**

La libreria Three.js mette a disposizione diverse tipologie di *mesh* per la visualizzazione di oggetti 3D, ciascuna caratterizzata da un consumo di risorse differente e da un comportamento diverso all’interno dell’ambiente. Durante il mio percorso

---

ho svolto un'attenta analisi delle prestazioni del prodotto, cercando di ottimizzare le operazioni di *rendering* aligerendo il carico di lavoro che la GPU doveva sostenere.

A tale scopo ho utilizzato una tipologia particolare di *mesh* chiamata `InstancedMesh`, che permette di istanziare un oggetto 3D a partire da un modello base, replicandolo in base a un insieme di parametri specificati. La particolarità di questa tipologia di *mesh* è che è possibile applicarla a diversi oggetti 3D (in questo caso per i diversi  $bin_G$  di ogni struttura), in un'unica operazione svolta dalla GPU, garantendo un approccio più efficiente rispetto all'istanziazione di una *mesh* per ogni oggetto.

### Vantaggi

La gestione centralizzata della *mesh* dei  $bin_G$  per ciascuna struttura semplifica notevolmente la selezione e l'interazione da parte dell'utente. In futuro, qualora si desiderasse implementare funzionalità aggiuntive (es. la disabilitazione di una struttura, la modifica del colore di un determinato gruppo di strutture per evidenziare diverse aree del magazzino, ...) si disporrebbe di un punto di accesso centralizzato per gestire tutte le strutture e i  $bin_G$  a esse associati.

### Considerazioni aggiuntive

A differenza di quanto avveniva in precedenza, in cui la *mesh* globale veniva caricata una sola volta, ora viene istanziata una `InstanceMesh` per ogni struttura. Questo aspetto è stato tenuto in particolare attenzione durante l'implementazione, controllando che il carico di risorse e la gestione della memoria fossero ottimizzati e che il prodotto risultante fosse performante e reattivo. Maggiori dettagli nel [paragrafo 3.6.3](#).

#### 3.4.4 Progettazione della funzionalità di creazione degli ordini di movimentazione

L'obiettivo è la gestione dei diversi aspetti che compongono le pratiche logistiche e amministrative di un magazzino. Durante il corso del mio tirocinio mi sono occupato di implementare la funzionalità per la generazione degli ordini di movimentazione mediante un'operazione di *drag & drop*, al fine di rendere tale processo più intuitivo e veloce.

L'operazione consiste nel selezionare un  $bin_G$  e trascinarlo in un'altra posizione dell'ambiente 3D, generando una richiesta di movimentazione manuale. Questo permetterebbe una gestione rapida ed intuitiva delle movimentazioni, avendo una visione concreta e diretta dell'ambiente 3D.

Nello specifico, la funzionalità generà una richiesta di movimentazione, i dettagli della richiesta e i  $task_G$  corrispondenti, uno per ogni saldo movimentato. In futuro, i  $task_G$  verranno presi in carico dagli operatori in magazzino, che si occuperanno di eseguire la movimentazione fisica dei saldi.

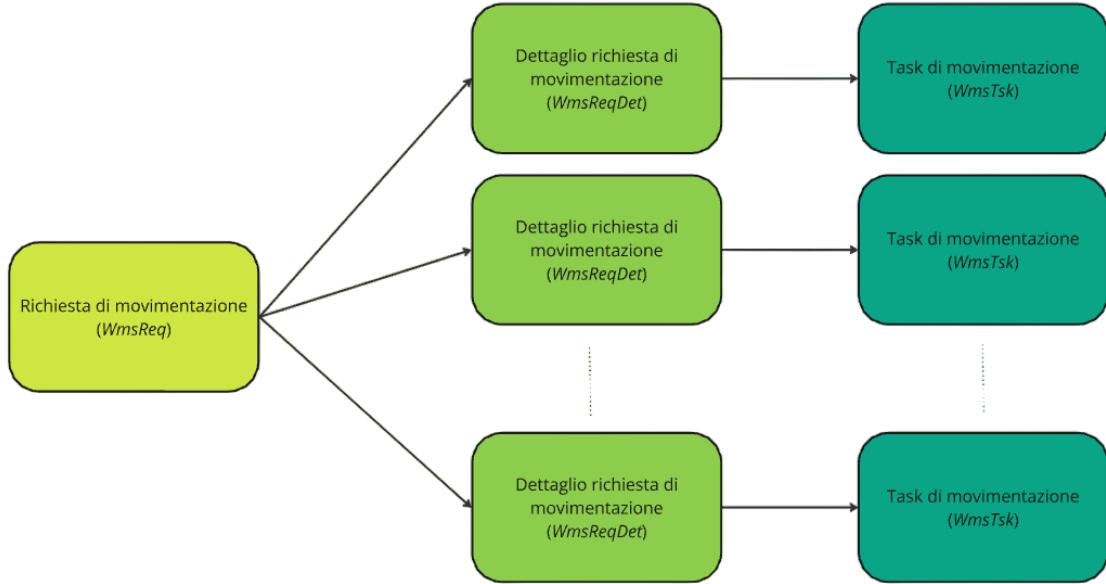


Immagine 36: Comportamento creazione dell'ordine di movimentazione

Come prima cosa, io e il team abbiamo definito le tabelle coinvolte, in modo da avere una visione chiara e completa delle informazioni necessarie per la creazione degli ordini di movimentazione. In particolare, abbiamo definito le seguenti tabelle:

- $WmsReq$ : tabella contenente le informazioni relative alla richiesta di movimentazione, la priorità, la tipologia e se si tratta di una richiesta manuale;
- $WmsReqDet$ : tabella contenente i dettagli della richiesta di movimentazione, uno per ogni saldo movimentato. Contiene le informazioni relative al prodotto movimentato e la quantità;
- $WmsTsk$ : tabella contenente i  $task_G$  corrispondenti alla richiesta di movimentazione, direttamente associato alla tabella  $WmsReqDet$ . Anche questa tabella contiene le informazioni relative al prodotto e la quantità movimentata, aggiungendo inoltre dettagli come la tipologia del  $task_G$ , il  $bin_G$  di partenza, il  $bin_G$  di destinazione e lo stato del  $task_G$ .



Immagine 37: Tabelle coinvolte nella creazione degli ordini di movimentazione

La gestione della connessione con il *database* è gestita dal *framework* Synergy mediante l'utilizzo di classi che utilizzano il *pattern singleton* per garantire un'unica istanza della connessione al *database*. Al fine di eseguire le operazioni di lettura e scrittura dei dati ho definito delle classi che implementassero il *pattern DAO* (*Data Access Object*) in grado di interfacciarsi con il *database* e fornire i metodi per l'esecuzione delle operazioni richieste. Questo mi ha permesso di separare la logica di accesso ai dati dalla logica di *business*, garantendo una maggiore modularità e scalabilità del codice.

### 3.4.5 Architettura del sistema

L'applicativo WMS è una *web application* che si compone di due parti principali: il *frontend*, sviluppato in Angular, e il *backend*, sviluppato in Java con il *framework* Synergy. Le due parti comunicano tra loro mediante servizi REST esposti.

Durante il mio tirocinio ho avuto modo di sviluppare funzionalità sia lato *frontend* che lato *backend*, adattandomi ai *pattern* architetturali presenti, in particolare:

- **Lato frontend:** ho seguito il *pattern MVVM* (*Model-View-ViewModel*), derivato dall'utilizzo di Angular, che permette di separare la logica di *business* dalla presentazione, garantendo una struttura chiara e ben organizzata del codice. In particolare, ho sviluppato i *componenti* e i *servizi* necessari per la visualizzazione dell'ambiente 3D e la creazione degli ordini di movimentazione.

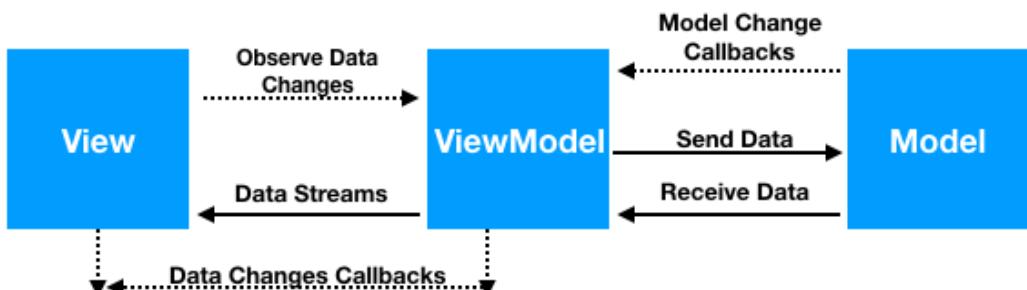
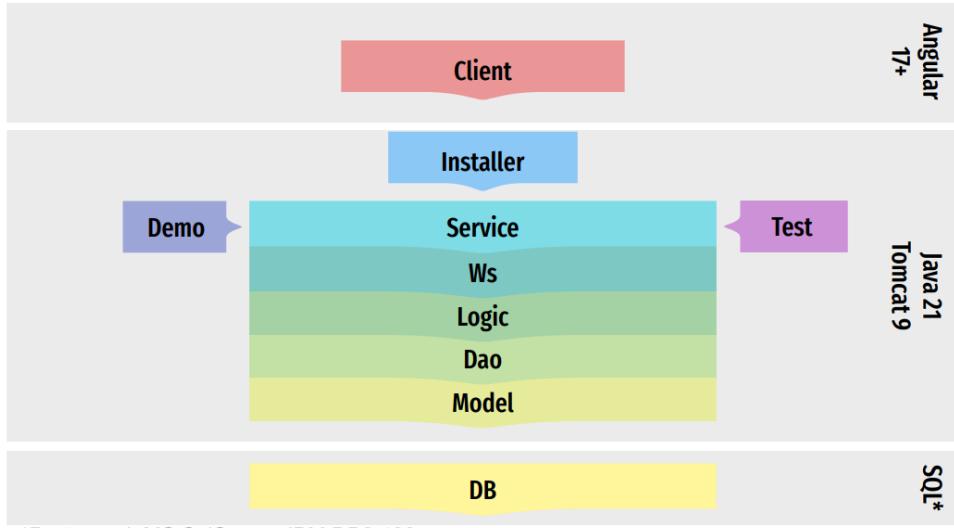


Immagine 38: Pattern MVVM del *frontend* con Angular

Fonte: <https://medium.com/@mutebibrian256/mastering-android-mvvm-architecture-developers-guide-3271e4c8908b>

- **Lato backend:** ho seguito il *pattern* architetturale a *layer* di Synergy, che permette di separare le diverse responsabilità del sistema in *layer* distinti, garantendo una struttura modulare e scalabile, come mostrato dall'immagine seguente:



\*Postgresql, MS SqlServer, IBM DB2 400

Immagine 39: Architettura a *layer* del backend con Synergy

Nell'implementazione delle funzionalità, ho lavorato attivamente a tutti i *layer* dell'architettura, rispettando le responsabilità di cui ciascun livello si fa carico:

- **Service layer:** *layer* che si occupa di esporre i servizi REST all'esterno, controllare l'autenticazione e di chiamare il *layer WS* per l'esecuzione delle operazioni richieste;
- **WS layer:** *layer* che si occupa di gestire i parametri delle chiamate REST, controllando inoltre le autorizzazioni per la scrittura e la lettura dei dati. Si occupa di chiamare il *layer* di logica per l'esecuzione delle operazioni richieste;
- **Logic layer:** *layer* che contiene la logica di *business* dell'applicativo. Si occupa di eseguire la logica di *business*, richiamando il DAO (*Data Access Object*) per interagire con il database.
- **DAO layer:** *layer* che si occupa di gestire l'accesso ai dati, interfacciandosi con il *database* e fornendo i metodi per le operazioni CRUD (*Create, Read, Update, Delete*);
- **Model layer:** *layer* che contiene le classi che rappresentano il modello dei dati dell'applicativo.

## 3.5 Codifica

### 3.5.1 Visualizzazione tridimensionale

Al fine di apportare le modifiche necessarie all'ambiente 3D definite durante il processo di progettazione, ho implementato due nuovi classi, responsabili della gestione delle strutture e dei  $bin_G$  all'interno dell'ambiente tridimensionale:

- **Bin3D**: la classe Bin3D rappresenta un  $bin_G$  all'interno dell'ambiente 3D, contenente le informazioni relative all'oggetto della *business logic* del  $bin_G$  stesso e le operazioni di interazione. La classe Bin3D contiene un riferimento alla InstancedMesh della struttura a cui appartiene, permettendo di visualizzare e interagire con il  $bin_G$  all'interno della struttura.
- **Struct3D**: la classe Struct3D rappresenta una struttura all'interno dell'ambiente 3D, contenente i diversi  $bin_G$  e le informazioni relative alla struttura stessa. La classe Struct3D contiene un'istanza di InstancedMesh per tutti i  $bin_G$  presenti al suo interno, permettendo di visualizzare e interagire con i  $bin_G$  all'interno della struttura.

Il rapporto che intercorre l'ho rappresentato con un diagramma UML<sub>G</sub>, come mostrato nell'immagine seguente:

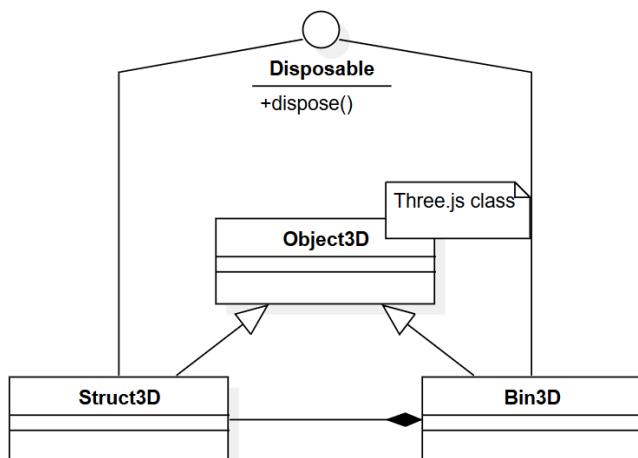


Immagine 40: Diagramma UML<sub>G</sub> delle classi Bin3D e Struct3D

Sia la classe Bin3D che la classe Struct3D sono specializzazioni della classe ‘Object3D’ di Three.js, che rappresenta un oggetto 3D all'interno della scena, ed entrambe implementano l’interfaccia Disposable, che permette di liberare le risorse allocate in memoria quando l’oggetto non è più utilizzato richiamando il metodo `dispose()` alla distruzione dell’oggetto.

L’interfaccia Disposable svolge un ruolo fondamentale nell’ottimizzazione delle risorse e nella gestione della memoria, garantendo che le risorse allocate vengano liberate correttamente evitando così eventuali *memory leak*.

---

All'interno dell'ambiente 3D, le due classi verranno mostrate come segue:

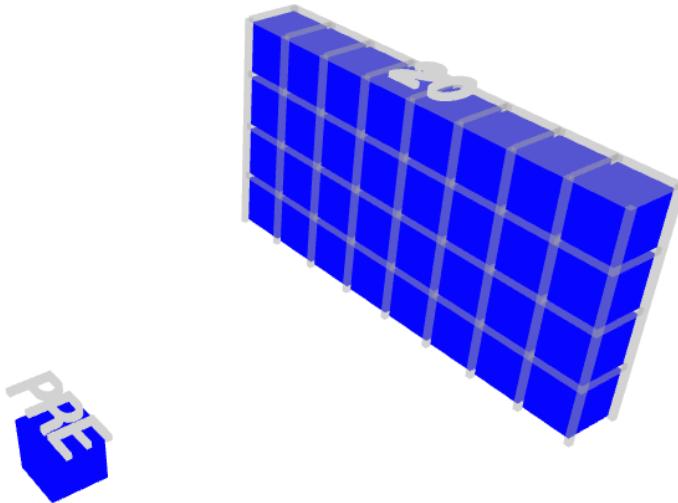


Immagine 41: Bin3D e Struct3D all'interno dell'ambiente 3D

Questo approccio ha permesso di centralizzare la logica di gestione dei  $bin_G$  e delle strutture all'interno dell'ambiente 3D, garantendo una struttura chiara e ben organizzata del codice. Un esempio significativo di come ho utilizzato queste classi è la gestione dell'interazione con i  $bin_G$  e le strutture: Struct3D e Bin3D espongono infatti i seguenti metodi per gestire l'interazione:

- **highlight ()**: permette di evidenziare il  $bin_G$  o la struttura selezionata, modificandone il colore;
- **select ()**: permette di selezionare il  $bin_G$  o la struttura, attivando la possibilità di operazioni di *drag & drop* e di creazione dell'ordine di movimentazione;
- **reset ()**: permette di ripristinare lo stato iniziale del  $bin_G$  o della struttura, rimuovendo l'evidenziazione e la selezione.

Nel momento in cui con il *mouse* ci si posiziona sopra la struttura o il  $bin_G$ , questo viene evidenziato e al click viene selezionato. Nel modo originale, la selezione avveniva direttamente sulla *mesh* globale, dovendo identificare l'indice dell'istanza della *mesh* corrispondente al  $bin_G$  selezionato. Con l'introduzione delle classi Bin3D e Struct3D, la selezione avviene direttamente sull'oggetto corrispondente, semplificando notevolmente la gestione dell'interazione e garantendo una maggiore flessibilità e scalabilità del codice.

Nel momento in cui si vuole evidenziare o selezionare un  $bin_G$ , ora che l'interazione avviene direttamente con l'oggetto Bin3D corrispondente, è sufficiente richiamare i metodi **highlight ()** e **select ()** dell'oggetto stesso, garantendo una gestione più chiara e modulare dell'interazione.



Immagine 42: Interazione con i bin<sub>G</sub>

Allo stesso modo, qualora si volesse evidenziare o selezionare una struttura, è sufficiente richiamare i metodi `highlight ()` e `select ()` dell’oggetto `Struct3D` corrispondente, che si occuperanno poi di gestire l’evidenziazione e la selezione dei `binG` al suo interno:

```
// evidenziazione struttura
public highlight (): void {
    for (const currentBin of this.binList) {
        currentBin.highlight ();
    } // for
} // highlight

// selezione struttura
public select (): void {
    for (const currentBin of this.binList) {
        currentBin.select ();
    } // for
} // select
```

### 3.5.2 Drag & Drop e creazione ordini di movimentazione

Lo sviluppo della funzionalità di *drag & drop* e di creazione degli ordini di movimentazione ha richiesto la creazione di un nuovo componente e dei servizi necessari per la gestione dell’interazione e la comunicazione con il *backend*.

Nello specifico ho implementato il componente `WmsNewMovementFormComponent`, responsabile della visualizzazione del *form* per la creazione degli ordini di movimentazione secondo le specifiche definite durante la progettazione.

Questo componente viene visualizzato al termine dell’operazione di *drag & drop* di un `binG`, permettendo all’utente di specificare la causale, la priorità per la creazione dell’ordine di movimentazione e di selezionare i saldi da movimentare e in che quantità.

Nell’immagine che segue mostro l’interfaccia del *form* per la creazione degli ordini di movimentazione:

	Art.	Quantità disponibile	Quantità r... / Unità...	Quantità alternativa dispo...	Quantità ... / Unità...
1	ASG002	35.0000 → 30.0000	5.0000 PZ	315.0000 → 270.0000	45.0000 KG
2	GRN001	40.0000	0.0000 PZ	320.0000	0.0000 KG
3	ASG001	30.0000 → 0.0000	30.0000 PZ	240.0000 → 0.0000	240.0000 KG
4	YN552255	6.0000	0.0000 PZ	30.0000	0.0000 KG
5	YN112255	33.0000 → 13.0000	20.0000 PZ	99.0000 → 39.0000	60.0000 KG
6	YF552255	120.0000	0.0000 PZ	600.0000	0.0000 KG
7	YF112255	90.0000	0.0000 PZ	270.0000	0.0000 KG
8	YA552255	34.0000	0.0000 PZ	170.0000	0.0000 KG

Immagine 43: *Dialog* per la creazione degli ordini di movimentazione

## Classi e servizi

I servizi in Angular sono classi che gestiscono in modo centralizzato i dati e la logica dell’applicazione, mettendo in comunicazione i diversi componenti e permettendo la comunicazione con il *backend*. Durante il mio tirocinio ho utilizzato i servizi per poter recuperare le informazioni necessarie alla visualizzazione dell’ambiente 3D e alla creazione degli ordini di movimentazione. In particolare, nell’implementazione di quest’ultima funzionalità ho sviluppato le seguenti classi e servizi:

- **InteractioService**: servizio responsabile della comunicazione con il *backend* per la creazione degli ordini di movimentazione. L’*InteractioService* si occupa di aprire il *dialog WmsNewMovementFormComponent* e successivamente di inviare le informazioni relative all’ordine al *backend*, gestendo la comunicazione con il servizio REST esposto;
- **DragManager**: classe responsabile della gestione dell’operazione di *drag & drop* all’interno dell’ambiente 3D. Il *DragManager* si occupa di gestire l’inizio e la fine dell’operazione di *drag & drop*, permettendo di selezionare e trascinare i *bin* <sub>G</sub> all’interno dell’ambiente 3D. Al momento del rilascio del *bin* <sub>G</sub>, il *DragManager* si occupa di generare l’evento di movimentazione e di richiamare il servizio *InteractioService* per la creazione dell’ordine, passando le informazioni relative al *bin* <sub>G</sub> di partenza e di destinazione;

---

L'operazione avviene come mostro nella seguente immagine:

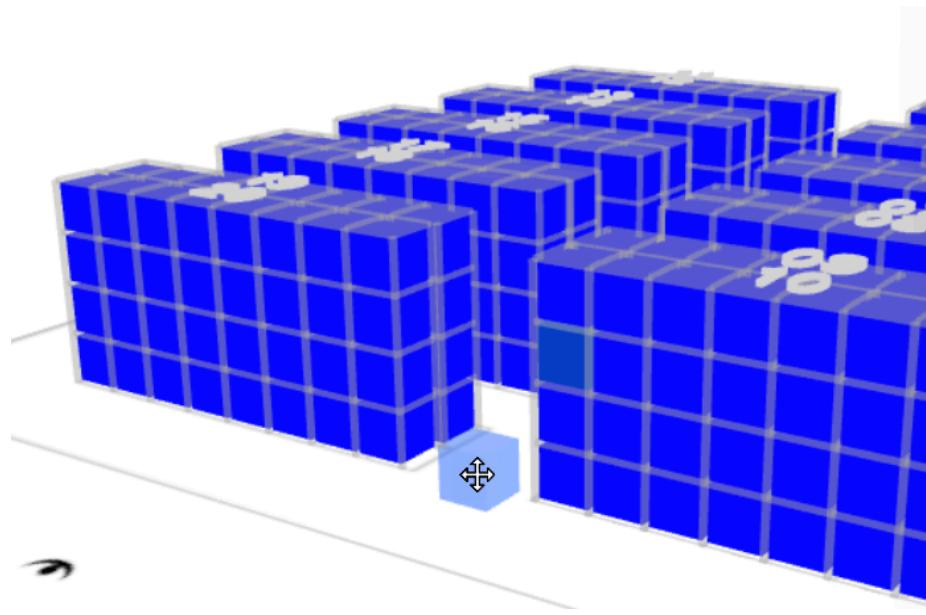


Immagine 44: *Drag & drop* nell'ambiente 3D

## Servizi REST

I servizi REST sono esposti dal *backend* per permettere la comunicazione con il *frontend* e la gestione delle operazioni di lettura e scrittura dei dati. Durante il mio tirocinio mi sono occupato di implementare i servizi REST necessari per la creazione degli ordini di movimentazione, secondo le convenzioni definite dal framework Synergy.

Come mostrato nell'[immagine 39](#), il *layer Service* espone i principali servizi disponibili lato *backend*, e, successivamente, le richieste vengono inoltrate al *layer WS*, il quale si occuperà a sua volta di richiamare i servizi della *business logic* per l'esecuzione delle operazioni richieste.

Il flusso delle chiamate inizia pertanto dal *layer Service* che espone il servizio REST in questo modo:

```
@Path("/WmsReq/createManualRequest") @antsan +1
@POST
@Consumes(MediaType.APPLICATION_JSON)
@Produces(MediaType.APPLICATION_JSON)
public Response createManualRequest (ApiInput<WmsReq> input, @SynergyRestContext() ExecutionContext context) {
    try (SWriteCon wCon = SConnectionFactory.get ().getWriteConnection ()) {
        RestOutput output = new RestOutput (WmsReqWs.get ().createManualRequest (input.entity, context, wCon), context);
        wCon.commit ();
        return Response.ok ().entity (output).build ();
    } catch (Throwable e) {
        return ErrorResponse.get ().manageError (e, context, rCon: null);
    } // try - catch
} // createManualRequest
```

Immagine 45: Servizio REST per la creazione dell'ordine esposto dal *backend*

---

Successivamente viene richiamato il *layer* WS, che controllerà i permessi di accesso alle tabelle:

```
/** {@link WmsReqLogic#createManualRequest (WmsReq, ExecutionContext, SWriteCon)} */
public WmsReq createManualRequest (WmsReq request, ExecutionContext context, SWriteCon wCon) throws SException {
    super.checkCreatePermission (context, wCon);
    return WmsReqLogic.get ().createManualRequest (request, context, wCon);
} // createManualRequest
```

Immagine 46: *Layer* WS per la creazione dell'ordine

Accertati i permessi, il *layer* WS richiamerà il *layer* di logica per l'esecuzione dell'operazione richiesta.

## 3.6 Verifica e validazione

### 3.6.1 Test di unità

Come menzionato in precedenza nel [paragrafo 1.4.4.2](#), ho svolto i *test* di unità per verificare il corretto funzionamento delle singole componenti e delle classi sviluppate, garantendo che le funzionalità implementate rispettassero le specifiche definite durante l'analisi.

Infatti, grazie al supporto del *framework* Synergy, per ogni classe del modello vengono predisposti i *test* di unità (e di integrazione) passando dei parametri di input e verificando che l'output sia conforme alle aspettative. Seppur non sono riuscito a seguire la pratica *test driven development* (TDD), ho comunque implementato i *test* di unità per garantire la correttezza delle funzionalità implementate, focalizzandomi sugli aspetti critici e i *corner case* che avrebbero potuto generare errori.

L'implementazione di questi *test* mi ha permesso pertanto di implementare un codice più robusto e di ridurre il numero di errori, garantendo che le funzionalità implementate fossero conformi alle specifiche definite.

### 3.6.2 Test di integrazione

I *test* di integrazione si occupano di verificare il corretto funzionamento delle componenti e dei servizi sviluppati, garantendo che le varie parti dell'applicativo interagiscano correttamente tra loro. Durante il mio tirocinio ho svolto i *test* di integrazione per verificare che le componenti sviluppate fossero in grado di comunicare tra loro e di scambiarsi i dati necessari, in particolar modo per verificare che l'interazione tra il *frontend* e il *backend* avvenisse correttamente.

Infatti, un aspetto cruciale che ho tenuto in considerazione durante lo sviluppo è stato il corretto funzionamento dei diversi servizi e chiamate REST che eseguivo per la creazione degli ordini di movimentazione, garantendo che i dati fossero conformi e che rispettassero i requisiti della *business logic*.

---

I *test* mi hanno permesso di verificare che i componenti che richiamavo nella *business logic*, come ad esempio i DAO per l’accesso ai dati, funzionassero correttamente e che i dati fossero correttamente scritti e letti dal *database*.

A tal fine è stato fondamentale comprendere il meccanismo di *mocking*, che mi ha permesso di simulare il comportamento delle classi e dei servizi che interagivano con il *backend*, garantendo che i *test* di integrazione fossero eseguiti in modo isolato e che non dipendessero da fattori esterni, rendendoli di fatto ripetibili e affidabili.

### 3.6.3 Test prestazionali

I *test* prestazionali sono stati svolti per verificare il corretto funzionamento dell’ambiente 3D e la gestione delle risorse durante l’interazione con l’utente, evitando che a seguito del *refactor* effettuato, il prodotto potesse riscontrare rallentamenti o problemi di *rendering*.

Questa tipologia di *test* è stata svolta grazie agli strumenti offerti da *Google Chrome* per la profilazione delle prestazioni, che mi hanno permesso di monitorare il carico di lavoro della GPU e la quantità di risorse utilizzate durante l’interazione con l’ambiente 3D.

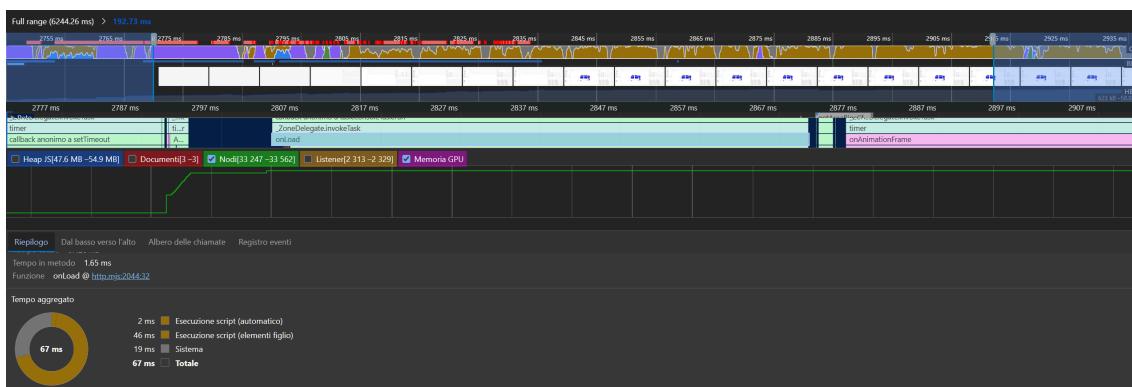


Immagine 47: *Test* prestazionali dell’ambiente 3D

Nell’immagine 47 mostro un esempio di *test* prestazionali dell’ambiente 3D, in cui è possibile visualizzare il carico di lavoro della GPU e la quantità di risorse utilizzate. Il tempo di esecuzione del metodo `onLoad()`, responsabile del caricamento delle strutture e dei `binG` all’interno dell’ambiente 3D, risulta in una media di valori tra i 60 e i 70 ms, garantendo un’esperienza utente fluida e reattiva, dimostrando che la ristrutturazione dell’ambiente 3D non ha compromesso le prestazioni del prodotto.

Inoltre, se andassimo a considerare il tempo complessivo di caricamento dell’intera pagina (quindi non solo del componente 3D ma di tutta l’applicazione), il tempo medio di caricamento risulta essere di circa 3 secondi, al di sotto dunque

---

del limite imposto da requisiti prestazionali (4s) precedentemente definiti ([paragrafo 3.3.2](#)).



Prima del caricamento dell'ambiente 3D      Dopo il caricamento dell'ambiente 3D

Immagine 48: Uso della GPU prima e dopo il caricamento dell'ambiente 3D

Come mostro nell'[immagine 48](#), il carico di lavoro della GPU risulta essere inferiore al 20%, comportando un carico di lavoro di circa 200MB di memoria che garantisce un'esperienza fluida e priva di rallentamenti.

### 3.6.4 Test di sistema

I *test* di sistema sono stati svolti al fine di garantire che il prodotto rispettasse i requisiti definiti durante l'analisi e che le funzionalità implementate fossero conformi alle specifiche richieste. Durante il mio tirocinio ho svolto i *test* di sistema in prima persona, in modo manuale: l'implementazione di *test* automatici, soprattutto considerando l'interazione con l'ambiente 3D, avrebbe potuto impedire la consegna di un prodotto completo e pronto per essere integrato nella *codebase<sub>G</sub>* aziendale.

Questa tipologia di *test*, seppur eseguita manualmente, l'ho svolta con costanza, in modo che ad ogni progresso delle funzionalità implementate, potessi verificare che le operazioni di *drag & drop* e di creazione degli ordini di movimentazione avvenissero in modo conforme alle aspettative. A tal fine, per verificare che i dati fossero letti e scritti correttamente nel *database*, ho utilizzato lo strumento DBeaver ([immagine 14](#)), che mi ha permesso di monitorare le operazioni effettuate e di assicurarmi che le richieste di movimentazione e i relativi dettagli fossero correttamente salvati.

### 3.6.5 Test di accettazione

I *test* di accettazione sono stati svolti direttamente dal *tester* del *team* di sviluppo e dal referente aziendale, che si sono occupati di verificare che il prodotto rispettasse i requisiti definiti durante l'analisi e che il prodotto rispecchiasse le aspettative del cliente. Nel contesto di tirocinio, questi *test* servono a verificare che gli obiettivi aziendali ([paragrafo 2.2.2.1](#)) siano stati raggiunti e che il percorso di *stage* abbia prodotto i risultati attesi.

---

Con i *test* di accettazione si conclude il processo di verifica e validazione del prodotto, assicurando che il lavoro che ho svolto durante il mio tirocinio possa soddisfare le aspettative dell'azienda e del cliente.

## 3.7 Risultati raggiunti

### 3.7.1 Il prodotto realizzato

Il prodotto che ho realizzato supera quanto atteso dall'azienda, risultando completo non solo dal punto di vista funzionale ma anche dal punto di vista dell'esperienza utente. L'implementazione della funzionalità di *drag & drop* e di creazione degli ordini di movimentazione ha permesso di semplificare e velocizzare le operazioni di movimentazione all'interno del magazzino, garantendo una gestione intuitiva e diretta delle operazioni.

Quando l'utente aprirà la *workspace* contenente il *widget* per la visualizzazione 3D del magazzino, avrà una visione chiara e completa della struttura dello stesso, evidenziando in modo chiaro i diversi scaffali presenti, identificati dal loro codice. L'utente è in grado di esplorare il magazzino muovendo la visuale con il mouse, e di modificare l'organizzazione delle strutture riposizionandole all'interno dell'ambiente 3D.

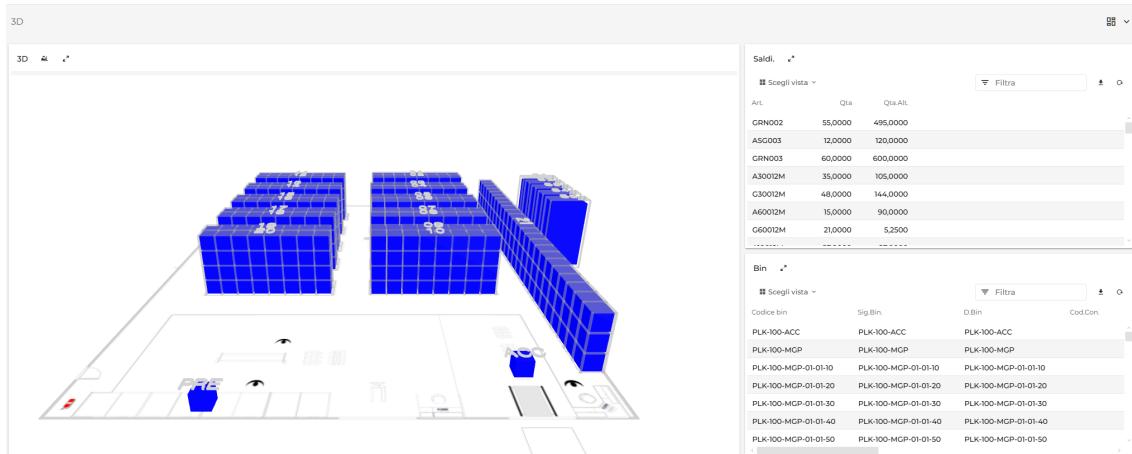


Immagine 49: Interfaccia finale della *workspace*

Nell'[immagine 49](#) mostro l'interfaccia finale del *workspace* utilizzato dall'utente: l'ambiente 3D che ho sviluppato si integra con gli altri *widget* presenti nell'ambiente di lavoro, offrendo funzionalità che semplificano la gestione del magazzino e che forniscono informazioni utili sul suo stato. In particolare l'utente può interagire con le strutture, selezionando i *bin<sub>G</sub>* di cui si vuole ispezionare il contenuto ([immagine 50](#)), oppure mostrare solo i *bin<sub>G</sub>* concretamente utilizzati aventi almeno un saldo al proprio interno ([immagine 51](#)). L'integrazione con gli altri *widget* inoltre, permette anche la selezione di un saldo dal *widget* dedicato, per

mostrare il suo collocamento all'interno dell'ambiente tridimensionale ([immagine 52](#)).

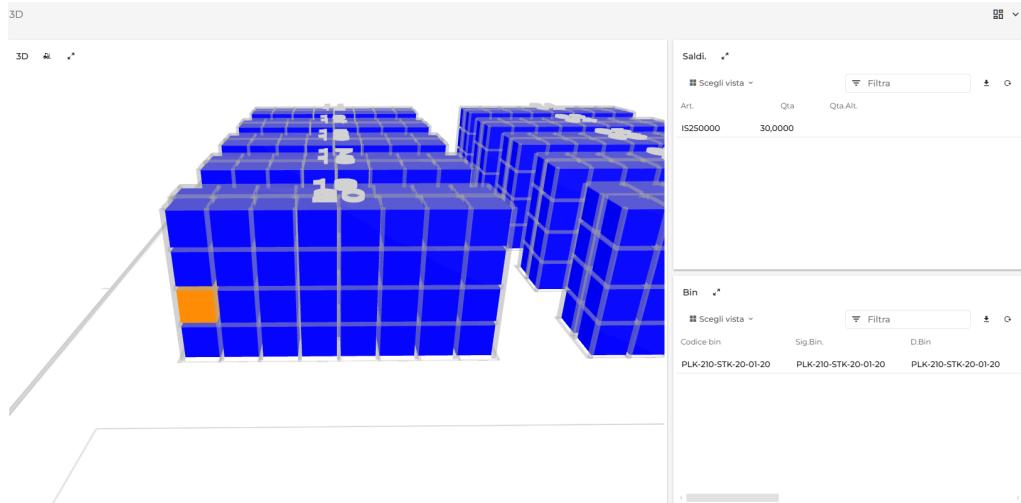


Immagine 50: Visualizzazione dei saldi contenuti all'interno del  $bin_G$  selezionato

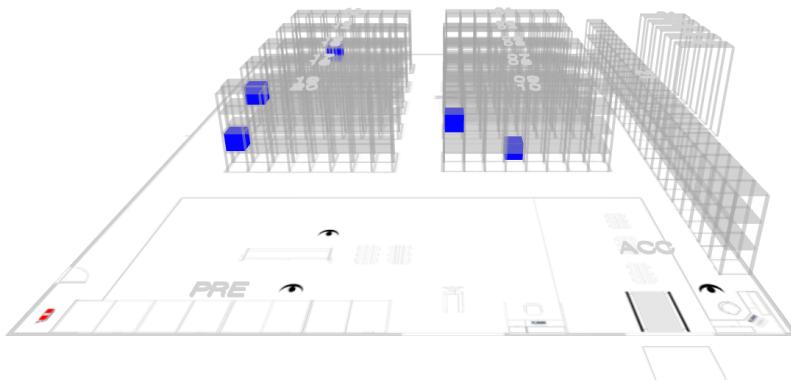


Immagine 51: Visualizzazione dei  $bin_G$  con almeno un saldo

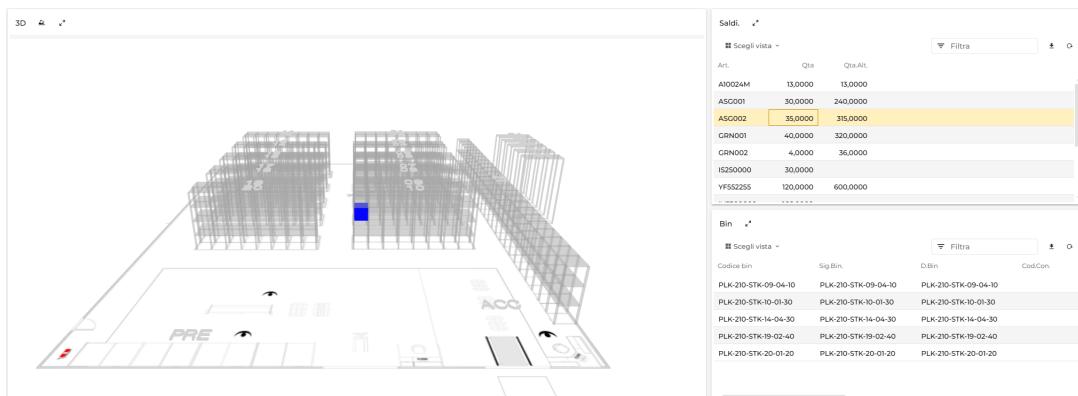


Immagine 52: Visualizzazione posizione del saldo selezionato dal widget dei saldi

Anche per quanto riguarda la funzionalità di *drag & drop* per la creazione dell'ordine di movimentazione, l'ho implementata in modo che tale operazione fosse rapida ed intuitiva: l'utente può selezionare un *bin* e trascinarlo in un'altra posizione dell'ambiente 3D. Questa operazione aprirà un *dialog* che gli permetterà di specificare i dettagli della richiesta con un'interfaccia semplice e facilmente comprensibile, come mostro nella seguente immagine:

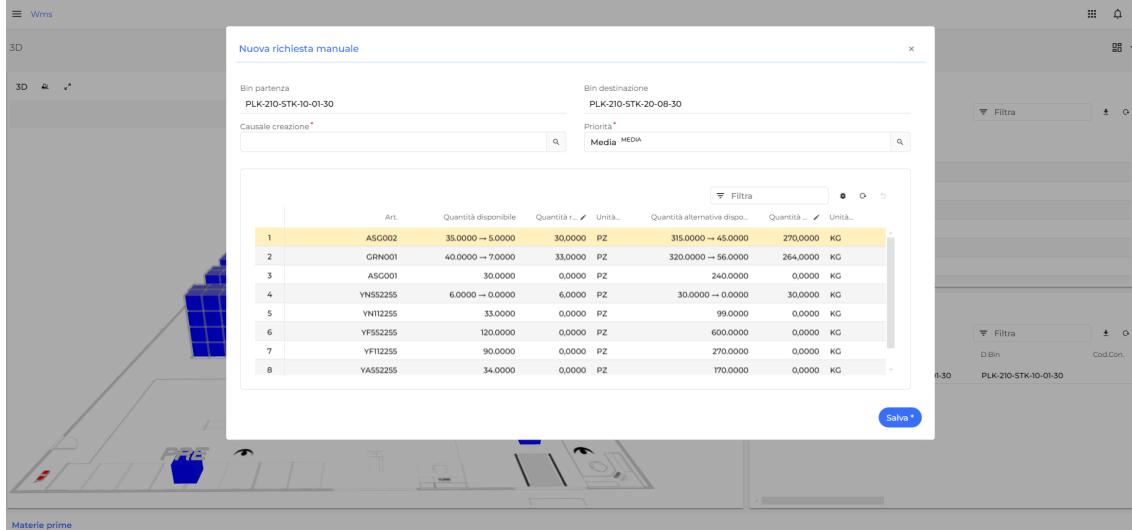


Immagine 53: *Dialog* di creazione dell'ordine di movimentazione

Alla conferma di creazione, l'utente visualizzerà la richiesta all'interno del *widget* dedicato.

### 3.7.2 Copertura dei requisiti

L'applicativo che ho sviluppato copre l'interezza dei requisiti delineati nel [paragrafo 3.3.2](#), garantendo che le funzionalità implementate rispettassero le specifiche definite durante l'analisi e che il prodotto rispecchiasse le aspettative dell'azienda e del cliente.

Tipologia	Individuati	Soddisfatti	Percentuale
Funzionali	15	15	100%
Qualità	4	4	100%
Prestazionali	1	1	100%
Vincolo	4	4	100%

Tabella 10: Copertura dei requisiti

### 3.7.3 Copertura di testing<sub>G</sub>

Durante lo svolgimento del tirocinio, mi sono impegnato ad implementare i *test* necessari a garantire un prodotto che fosse conforme alle aspettative e un che il codice fosse robusto e mantenibile.

I *test* implementati, grazie anche alla predisposizione offerta dal *framework Synergy*, hanno una copertura del 100% delle codice prodotto lato *backend*, con particolare attenzione ai *test* di integrazione nella verifica del corretto funzionamento dei servizi REST coinvolti.

Nella seguente tabella riporto il livello di copertura dei *test* implementati:

Tipologia	Esecuzione	Implementati	Copertura
Unità	Automatica	Predisposti dal <i>framework Synergy</i>	100%
Integrazione	Automatica	22	100%
Performance	Manuale	2	100%
Sistema	Manuale	15 (uno per ogni requisito <sub>G</sub> funzionale)	100%

Tabella 11: Copertura dei *test*

### 3.7.4 Materiali prodotti

Il mio percorso di tirocinio ha visto la produzione di diversi materiali necessari al fine di garantire un'esperienza formativa sia sul profilo personale sia sul profilo tecnico lavorativo. Di seguito, riporto una valutazione quantitativa dell'esperienza svolta, in base ai diversi aspetti considerati:

Organizzativoe	
<i>Daily meeting</i> svolti	30
<i>Sprint<sub>G</sub> review<sub>G</sub></i> svolte	3
<i>Sprint<sub>G</sub> retrospective</i> svolte	3
Tecnico	
<i>Ticket<sub>G</sub></i> di Jira <sub>G</sub> risolti	11
<i>Bug<sub>G</sub></i> risolti	7
Classi implementate	4
Servizi implementati	2
Servizi REST implementati	4

---

<b>Documentale</b>	
Resoconti giornalieri redatti	36
Resoconti settimanali redatti	8
Documentazione tecnica prodotta	1 (omnicomprensiva)
<b>Temporale</b>	
Ore totali di tirocinio	320

Tabella 12: Materiali complessivamente prodotti durante il tirocinio

---

# **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.

---

### **R**

---

**REST:** Representational State Transfer.

---

### **U**

---

**UC:** Use Case.

**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.

### **Bin**

Unità minima di spazio del magazzino atta a contenere prodotti. I bin sono posizionati in automatico all'interno degli scaffali al momento della loro creazione, e possono contenere al massimo un tipo di prodotto. È possibile creare bin, eliminarli se vuoti, e modificarne le dimensioni.

### **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

---

## **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.

---

# 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.

---

# **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

### A

---

- Sito web ufficiale di Angular  
<https://angular.io/> (*ultimo accesso 08/08/2024*)

### D

---

- Documentazione ufficiale di Three.js  
<https://threejs.org/docs/> (*ultimo accesso 08/08/2024*)

### 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*)

### P

---

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

### 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*)