Università degli Studi di Padova
Dipartimento di Matematica "Tullio Levi Civita"
Corso di Laurea in Informatica

Padova, 16 Dicembre 2022

Angular 12 e TypeScript per la Realizzazione di un Componente per il Workflow della Ricerca dello Storico Nell'Inventory Chain Platform

Laureanda
Greta Cavedon, 1123054

Relatrice
Prof.ssa Ombretta Gaggi

A.A. 2021 — 2022

A Chicco, per esserci sempre stato ed aver creduto in me,
Ai miei nonni, Checco e Fernanda,
per avermi permesso di avere un futuro migliore.

Ringraziamenti

In primis, desidero ringraziare immensamente mia mamma Sonia per avermi sopportata e supportata in questo lungo percorso (oltre che per il treno delle 6:51)!

Ringrazio mio nonno Rino per essere sempre stato un esempio di vita e mia nonna Mariuccia per avermi trasmesso la sua solarità ed il suo ottimismo.

Un grazie sincero anche alla mia relatrice, la prof.ssa Ombretta Gaggi, per la sua immensa disponibilità e per avermi dato dei consigli preziosi, al di là del mero esame di laurea o del tirocinio svolto.

Infine, ringrazio i miei amici e compagni di corso (in particolar modo, Giusp, Chime, GioInc, Luca Mkal, Trov e Fif), per avermi aiutata nei momenti di difficoltà, per aver reso questo percorso meno tortuoso e ricco di momenti di gioia! Vi voglio bene!

Indice

Elenco delle figure

Elenco delle tabelle

Sommario

Il presente documento ha lo scopo di riassumere il periodo di tirocinio formativo della laureanda Greta Cavedon presso l'azienda Sopra Steria svolto dal 6 Giugno 2022 al 2 Agosto 2022 in modalità duale: da remoto e nella sede di Milanofiori (Assago, Milano).

L'obiettivo dello stage è stato quello di sviluppare un componente frontend[G]sfruttando il framework Angular 12 e il linguaggio di programmazione TypeScript, per una piattaforma web relativa alla gestione di beni della filiera agroalimentare.

Nelle prime settimane di tirocinio sono state studiate le diverse tecnologie utilizzate dal team di sviluppo, al fine di realizzare un componente, realizzato nella seconda parte del periodo di stage, che si integrasse in maniera efficiente all'intero del prodotto e che seguisse le richieste del committente.

1 Introduzione

In questo capitolo verrà descritta Sopra Steria Group, l'azienda ospitante presso la quale ho svolto il tirocinio formativo, ed alcune norme tipografiche usate per la stesura del documento.

1.1 L'azienda

The World is How We Shape It.

Sopra Steria è una società francese (con logo mostrato in figura 1), con sede centrale a Parigi, che si occupa di consulenza nella trasformazione digitale di aziende ed organizzazioni.

Figura 1: Il logo di Sopra Steria

L'azienda nasce nel gennaio 2015 dalla fusione di due società francesi che si sono sempre occupate di servizi digitali e dell'industria dei servizi informatici, Sopra, creata nel 1968, e Steria, creata nel 1969, con un obiettivo comune: rispondere alle esigenze di clienti di rilievo presenti nel mercato nazionale ed internazionale con prodotti e servizi innovativi. Ad oggi, Sopra Steria è presente in oltre 30 paesi del mondo, vanta di oltre 47.000 risorse (di cui circa 1.000 in Italia) e, nel 2020, ha registrato un ricavo pari a 4,7mld di euro (di cui 75M in Italia).
In Italia ha ben sette sedi:

- Assago,
- Asti,
- Padova,
- Collecchio,
- Roma,
- Ariano Irpino,
- Napoli.

e cinque divisioni:

- Retail, Fashion & Industry Italy,
- Financial Services & Insurance Italy (la divisione su cui sono stata inserita per lo sviluppo del progetto),
- Public Sector, Energy & Telco Italy,
- Service Center Salesforce Italy,
- Innovation.

1.1.1 Servizi e Standard

Sopra Steria offre servizi di consulenza tecnologia, progettazione, costruzione e gestione, oltre a prodotti per il mercato bancario, immobiliare e per le risorse umane su misura delle esigenze del cliente.

Inoltre, Sopra Steria ha ottenuto anche diversi certificati, quali:

- **ISO 14001:2015**: che identifica se una determinata organizzazione ha un sistema di gestione adeguato a tenere sotto controllo gli impatti ambientali delle proprie attività e ne ricerca, sistematicamente, il miglioramento in modo coerente, efficace e soprattutto sostenibile. Si tratta di una certificazione di processo;

- **ISO 14064-1:2018**: certificazione relativa alle emissioni di gas serra, nonché strumento di controllo e riduzione delle emissioni sia come modalità per comunicare la sensibilità dell'azienda sulle tematiche della sostenibilità
- **ISO 20001:2018**: standard internazionale sviluppato specificatamente per la gestione dei servizi informatici, che mira al miglioramento dell'erogazione/fruizione dei servizi IT, ponendosi come obiettivo il raggiungimento della massima qualità dei servizi erogati e un massimo contenimento di costi;
- **ISO 27001-004:2015**: norma internazionale che contiene i requisiti per impostare e gestire un sistema di gestione della sicurezza delle informazioni. Oltre ad includere la sicurezza legata all'ambito informatico, include anche la sicurezza fisica/ambientale e la sicurezza organizzativa;
- **ISO 37001:2016**: identifica uno standard di gestione per aiutare le organizzazioni nella lotta contro la corruzione, istituendo una cultura di integrità, trasparenza e conformità. La norma può fornire un importante aiuto nell'implementazione di misure efficaci per prevenire ed affrontare fenomeni di corruzione;
- **ISO 45001:2018**: si tratta di una norma internazionale che specifica i requisiti per un sistema di gestione della salute e sicurezza sul lavoro. Consente alle organizzazioni di fornire posti di lavoro sicuri e salubri, prevenendo infortuni sul lavoro e problemi di salute, nonché di migliorare la salute e sicurezza sul lavoro in modo proattivo;
- **ISO 9001:2015**: identifica una serie di linee guida che definiscono i requisiti per la realizzazione all'interno di un'organizzazione di un sistema di gestione della qualità, al fine di condurre i processi aziendali, migliorare l'efficacia e l'efficienza nella realizzazione del prodotto e nell'erogazione dei servizi offerti, al fine di ottenere ed incrementale la soddisfazione del cliente;
- **SA 8000:2014**: identifica uno standard internazionale volto a certificare alcuni aspetti della gestione aziendale attinenti alla responsabilità sociale d'impresa, ossia: il rispetto dei diritti umani, il rispetto al diritto del lavoro, la tutela contro lo sfruttamento minorile e le garanzie di sicurezza e salubrità sul posto di lavoro.

1.2 Descrizione dello stage

Questo capitolo è relativo al progetto di stage ed alla stesura del piano di lavoro concordato assieme al relatore esterno, messo in atto solo dopo l'approvazione del relatore interno.

1.2.1 Il Progetto

Il prodotto su cui ho avuto l'opportunità ed il piacere di lavorare si chiama "*Inventory Chain Platform*" (abbreviato "ICP") e consiste in una applicazione web il cui scopo è quello di "*digitalizzare la gestione del pegno non possessorio, assicurando un'adeguata sorveglianza del bene oggetto della garanzia, mediante l'impiego della blockchain*", dove per bene oggetto si intendono delle forme di Grana Padano e Parmigiano Reggiano.

Detto in altri termini, non è altro che uno strumento che permette di monitorare i lotti di formaggio ubicati all'interno di un magazzino, il relativo valore economico (che può variare in base alla stagionatura) e le garanzie ad essi associate.

Fondamentalmente, questa piattaforma è uno strumento utile a tre attori:

- La latteria, per monitorare il valore economico dei beni messi a pegno,
- Il magazziniere, che si occupa di gestire le forme vere e proprie all'interno di un magazzino fisico,
- La banca, per monitorare i beni, il relativo valore e rilasciare finanziamenti.

La parte del progetto interessata dalle mie modifiche è stata la sezione legata all'iter MGT (acronimo di "*Magazzino delle Tagliate*"); si tratta di una personalizzazione realizzata per un particolare magazzino e cliente, la quale gestisce il carico, lo scarico ed il pignoramento della merce in quel determinato deposito.

Infine, per annotare tutte le operazioni rilevanti svolte dai diversi partecipanti, viene sfruttata la blockchain su un registro distribuito.

1.2.2 Lo Stage

Come anticipato, il tirocinio si è svolto in modalità duale: da remoto e presso la sede di Assago (Milano). Il tirocinio ha avuto una durata di otto settimane (per un totale effettivo di 312 ore), dal lunedì al venerdì, dalle 09:00 alle 13:00 e dalle 14:00 alle 18:00.

Le prime settimane si sono concentrate sullo studio individuale delle tecnologie adottate per lo sviluppo del componente da realizzare per il prodotto, mentre nella seconda parte del tirocinio è stato messo in pratica quanto appreso per la realizzazione del componente richiesto.

I tools e le tecnologie affrontate nelle prime settimane di lavoro sono stati i seguenti:

- Angular 12 e, in particolar modo, la libreria Angular Material,
- TypeScript,
- Flowable,
- Keyclock,
- Swagger,
- Pattern architetturale MVC,
- Open-API,
- Jira,
- GitLab.

Nella seconda parte dello stage è stato realizzato un componente capace di sfruttare tutte queste tecnologie e tecniche per il progetto "*Inventory Chain Project*".

1.2.3 Obiettivi

Affinché il tirocinio potesse dirsi concluso con successo, è stato necessario soddisfare gli obiettivi accordati con il tutor.

Notazione Al fine di identificare gli obiettivi in maniera univoca, verranno utilizzate le seguenti notazioni:

$$\{O\}.\{X\}.\{Y\}$$

Dove:

- O: sta per "Obiettivo",
- X: sta per la tipologia di requisito,

- Y: è il numero di requisito per tipologia.

I requisiti possono essere:

- O: per i requisiti obbligatori,
- D: per i requisiti desiderabili,
- F: per i requisiti facoltativi.

Obiettivi fissati Gli obiettivi accordati a monte del tirocinio sono i seguenti:

- Requisiti obbligatori:
  - OO1: Studio e comprensione del framework Angular 12 e del linguaggio TypeScript;
  - OO2: Abilità nel raggiungere gli obiettivi predisposti nel piano di lavoro in autonomia;
  - OO3: Capacità di collaborare con altri membri del gruppo in maniera proattiva e produttiva.
- Requisiti desiderabili:
  - OD1: Utilizzo a livello avanzato del framework Angular 12.
- Requisiti facoltativi:
  - OF1: Imparare ad effettuare tutte le operazioni di *DevOps*^{G} per la gestione dell'infrastruttura.

1.3 Convenzioni Tipografiche

Per la stesura del documento sono state adottate le seguenti convenzioni tipografiche:

- I termini che potrebbero risultare ambigui o di uso non comune menzionati in seguito saranno contraddistinti dalla lettera ^{G} accanto alla parola avranno una breve descrizione nel glossario, situato alla fine del presente documento;
- I termini in lingua straniera o facenti parte del gergo tecnico e le citazioni sono evidenziati in carattere *corsivo*.

2 Analisi Dei Requisiti

In questo capitolo verrà descritto, in maniera precisa e mediante dei diagrammi relativi ai casi d'uso, il componente che è stato realizzato per la piattaforma Inventory Chain Platform.

2.1 Casi D'Uso

Attore Primario

Utente autenticato: un utente che ha effettuato l'autenticazione ad ICP ed ha la possibilità di accedere al WMS per la gestione degli iter. Più nello specifico, un utente autenticato potrebbe essere una latteria, il magazziniere o la banca.

Comprensione del contesto

Lo storico iter non è altro che una pagina nella quale visualizzare, dopo aver impostato determinati filtri, la cronologia degli iter e, per ciascuno di essi, un'anteprima dei dettagli più rilevanti. Un iter non è altro che una procedura (in questo caso, uno scarico, un deposito o un pignoramento di beni da/verso un magazzino) che un produttore può avviare mediante la compilazione di un form.

Questo form presenta diversi step e ciascuno step viene valorizzato da un attore diverso (il reparto amministrativo del produttore, magazziniere e l'impiegato della banca che si occupa del finanziamento o del deposito merci) a seconda dello scopo.

2.1.1 UC 0 – Storico Iter

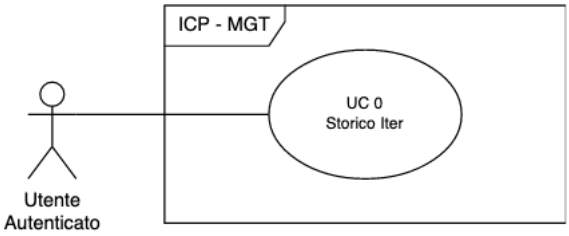


Figura 2: UC 0 - Storico Iter

- **Attore primario**: Utente autenticato;
- **Descrizione**: L'utente autenticato nella piattaforma va alla pagina relativa allo Storico Iter;
- **Precondizione**: L'utente è autenticato nella piattaforma ed ha la possibilità di accedere alla sezione dedicata agli iter;
- **Postcondizione**: L'utente accede alla pagina relativa allo Storico Iter.
- **Scenario principale**:
  1. L'utente ha effettuato l'accesso al sistema;
  2. L'utente si trova nella pagina principale di gestione iter;
  3. L'utente clicca sull'icona relativa allo "Storico Iter".

2.1.2 UC 1 - Visualizza Form Storico Iter

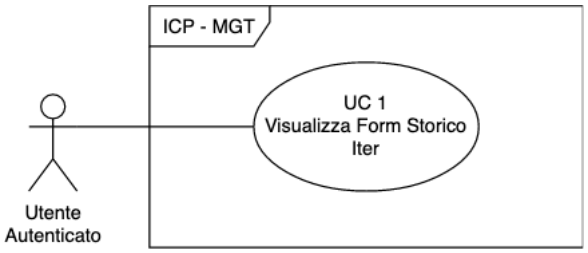


Figura 3: UC 1 - Visualizza Form Storico Iter

- **Attore primario**: Utente autenticato;
- **Descrizione**: L'utente autenticato nella piattaforma e visualizza il form relativo allo storico iter, che permette di filtrare i risultati;
- **Precondizione**: L'utente è autenticato nella piattaforma ed ha aperto la pagina dedicata allo Storico Iter;
- **Postcondizione**: L'utente ha la possibilità di visualizzare il form relativo allo storico iter.
- **Scenario principale**:
  1. L'utente ha effettuato l'accesso al sistema;
  2. L'utente si trova nella pagina principale di gestione iter;
  3. L'utente clicca sull'icona relativa allo "Storico Iter";
  4. L'utente inserisce i filtri necessari per effettuare la ricerca;
  5. Affinché la ricerca vada a buon fine, dopo aver inserito i filtri l'utente deve cliccare su "Cerca" (oppure "Pulisci" nel caso in cui voglia re-impostare i filtri).
- **Sottocasi**:
  1. Scelta del filtro per Tipologia Iter (§2.1.3 UC 1.1),
  2. Scelta del filtro per Stato Iter (§2.1.4 UC 1.2),
  3. Visualizzare il cliente che ha fatto partire l'iter (§2.1.5 UC 1.3)
  4. Scelta del filtro per data inizio (§2.1.6 UC 1.4),
  5. Scelta del filtro per data fine (§2.1.7 UC 1.5).

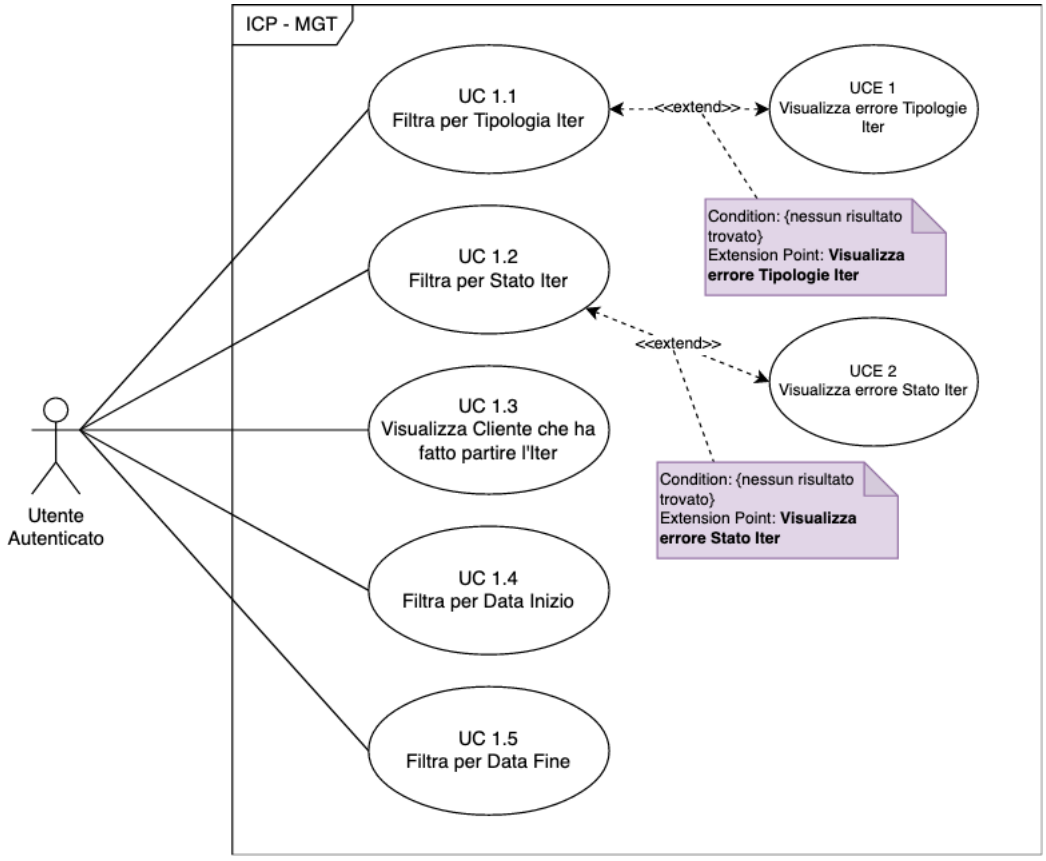


Figura 4: Sottocasi Visualizza Storico Iter

2.1.3 UC 1.1 - Filtra per Tipologia Iter

- **Attore primario**: Utente autenticato;
- **Descrizione**: L'utente autenticato nella piattaforma clicca sul menù a tendina per selezionare la tipologia iter (che può essere: deposito forme, pignoramento e riscatto forme, scarico forme);
- **Precondizione**: L'utente è autenticato nella piattaforma, si trova nella pagina relativa allo storico Iter ed ha la possibilità di filtrare gli iter in base alla tipologia;
- **Postcondizione**: L'utente ha impostato il filtro relativo alla tipologia dell'iter nell'opportuno campo del modulo.
- **Scenario principale**:
  1. L'utente ha effettuato l'accesso al sistema;
  2. L'utente si trova nella pagina principale di gestione iter;
  3. L'utente clicca sull'icona relativa allo "Storico Iter";
  4. L'utente seleziona dal menù a tendina la tipologia iter.
- **Estensioni**:
  1. Visualizza errore Tipologia Iter (§2.1.18 UCE 1).

2.1.4 UC 1.2 - Filtra per Stato Iter

- **Attore primario**: Utente autenticato;
- **Descrizione**: L'utente autenticato nella piattaforma clicca sul menù a tendina per selezionare lo stato iter (che può essere: in corso, terminato, declinato o non autorizzato);
- **Precondizione**: L'utente è autenticato nella piattaforma, si trova nella pagina relativa allo storico Iter ed ha la possibilità di filtrare gli iter in base allo stato iter;
- **Postcondizione**: L'utente ha impostato il filtro relativo allo stato dell'iter nell'opportuno campo del modulo.
- **Scenario principale**:
  1. L'utente ha effettuato l'accesso al sistema;
  2. L'utente si trova nella pagina principale di gestione iter;
  3. L'utente clicca sull'icona relativa allo "Storico Iter";
  4. L'utente seleziona dal menù a tendina lo stato iter.
- **Estensioni**:
  1. Visualizza errore Stato Iter (§2.1.8 UCE 2).

2.1.5 UC 1.3 - Visualizzare cliente che ha fatto partire l'iter

- **Attore primario**: Utente autenticato;
- **Descrizione**: L'utente autenticato nella piattaforma visualizza la denominazione del cliente che ha inizializzato l'iter;
- **Precondizione**: L'utente è autenticato nella piattaforma e si trova nella pagina relativa allo storico Iter;
- **Postcondizione**: L'utente visualizza il cliente che ha fatto partire l'iter nell'opportuno campo del modulo.
- **Scenario principale**:
  1. L'utente ha effettuato l'accesso al sistema;
  2. L'utente si trova nella pagina principale di gestione iter;
  3. L'utente clicca sull'icona relativa allo "Storico Iter";
  4. L'utente visualizza il nome del cliente che ha fatto partire l'iter.

2.1.6 UC 1.4 - Filtra per Data Inizio

- **Attore primario**: Utente autenticato;
- **Descrizione**: L'utente autenticato nella piattaforma clicca sul calendario per selezionare la data di inizio dell'iter;
- **Precondizione**: L'utente è autenticato nella piattaforma e si trova nella pagina relativa allo storico Iter;
- **Postcondizione**: L'utente ha impostato il filtro relativo alla data di inizio dell'iter nell'opportuno campo del modulo.
- **Scenario principale**:

1. L'utente ha effettuato l'accesso al sistema;
2. L'utente si trova nella pagina principale di gestione iter;
3. L'utente clicca sull'icona relativa allo "Storico Iter";
4. L'utente seleziona dal calendario la data di inizio iter.

2.1.7 UC 1.5 - Filtra per Data Fine

- **Attore primario**: Utente autenticato;
- **Descrizione**: L'utente autenticato nella piattaforma clicca sul calendario per selezionare la data di fine dell'iter;
- **Precondizione**: L'utente è autenticato nella piattaforma e si trova nella pagina relativa allo storico Iter;
- **Postcondizione**: L'utente ha impostato il filtro relativo alla data di fine dell'iter.
- **Scenario principale**:
  1. L'utente ha effettuato l'accesso al sistema;
  2. L'utente si trova nella pagina principale di gestione iter;
  3. L'utente clicca sull'icona relativa allo "Storico Iter";
  4. L'utente seleziona dal calendario la data di fine iter.

2.1.8 UC 2 - Visualizza Storico Iter

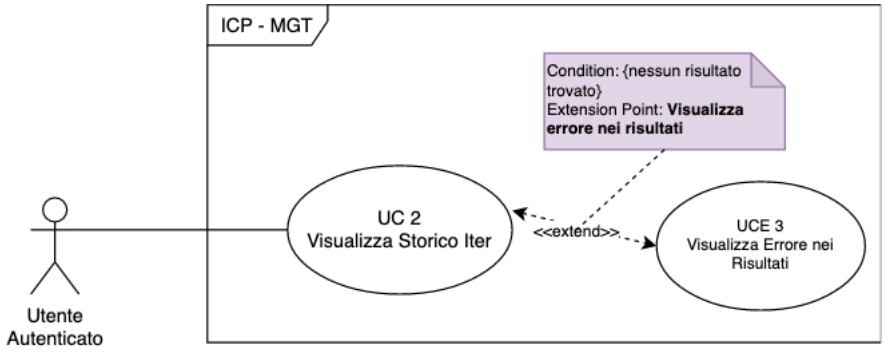


Figura 5: UC 2 - Visualizza Storico Iter

- **Attore primario**: Utente autenticato;
- **Descrizione**: L'utente autenticato nella piattaforma visualizza i risultati prodotti dalla ricerca;
- **Precondizione**: L'utente è autenticato nella piattaforma e si trova nella pagina relativa allo Storico Iter;
- **Postcondizione**: L'utente autenticato visualizza i risultati prodotti dalla ricerca.
- **Scenario principale**:
  1. L'utente ha effettuato l'accesso al sistema;
  2. L'utente si trova nella pagina principale di gestione iter;
  3. L'utente clicca sull'icona relativa allo "Storico Iter";

4. L'utente ha selezionato i filtri necessari per la ricerca;
5. Se la ricerca ha prodotto dei risultati, l'utente visualizza una lista di iter in base ai filtri inseriti.

- **Estensioni**:
  1. Visualizza Errore nei Risultati (§2.1.20 UCE 3).

2.1.9 UC 2.1 - Visualizza Singolo Iter

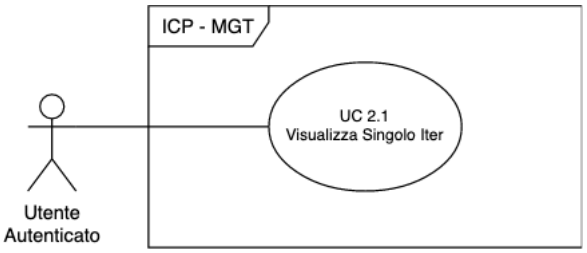


Figura 6: UC 2.1 - Visualizza Singolo Iter

- **Attore primario**: Utente autenticato;
- **Descrizione**: L'utente autenticato nella piattaforma visualizza un singolo iter presente nei risultati prodotti dalla ricerca;
- **Precondizione**: L'utente è autenticato nella piattaforma, si trova nella pagina relativa allo storico Iter ed ha ottenuto dei risultati dalla ricerca effettuata;
- **Postcondizione**: L'utente visualizza un singolo iter.
- **Scenario principale**:
  1. L'utente ha effettuato l'accesso al sistema;
  2. L'utente si trova nella pagina principale di gestione iter;
  3. L'utente clicca sull'icona relativa allo "Storico Iter";
  4. L'utente ha selezionato i filtri necessari per la ricerca;
  5. Se la ricerca ha prodotto dei risultati, l'utente visualizza una lista di iter in base ai filtri inseriti;
  6. L'utente sceglie di visualizzare un singolo iter presente nella lista dei risultati.
- **Sottocasi**:
  1. Visualizza ID Iter (§2.1.10 UC 2.1.1),
  2. Visualizza cliente (§2.1.11 UC 2.1.2),
  3. Visualizza Tipologia Iter (§2.1.12 UC 2.1.3),
  4. Visualizza Percorso Iter (§2.1.13 UC 2.1.4),
  5. Visualizza Stato Iter (§2.1.14 UC 2.1.5),
  6. Visualizza Data Inizio (§2.1.15 UC 2.1.6),
  7. Visualizza Data Fine (§2.1.16 UC 2.1.7),
  8. Visualizza Link Iter (§2.1.17 UC 2.1.8)

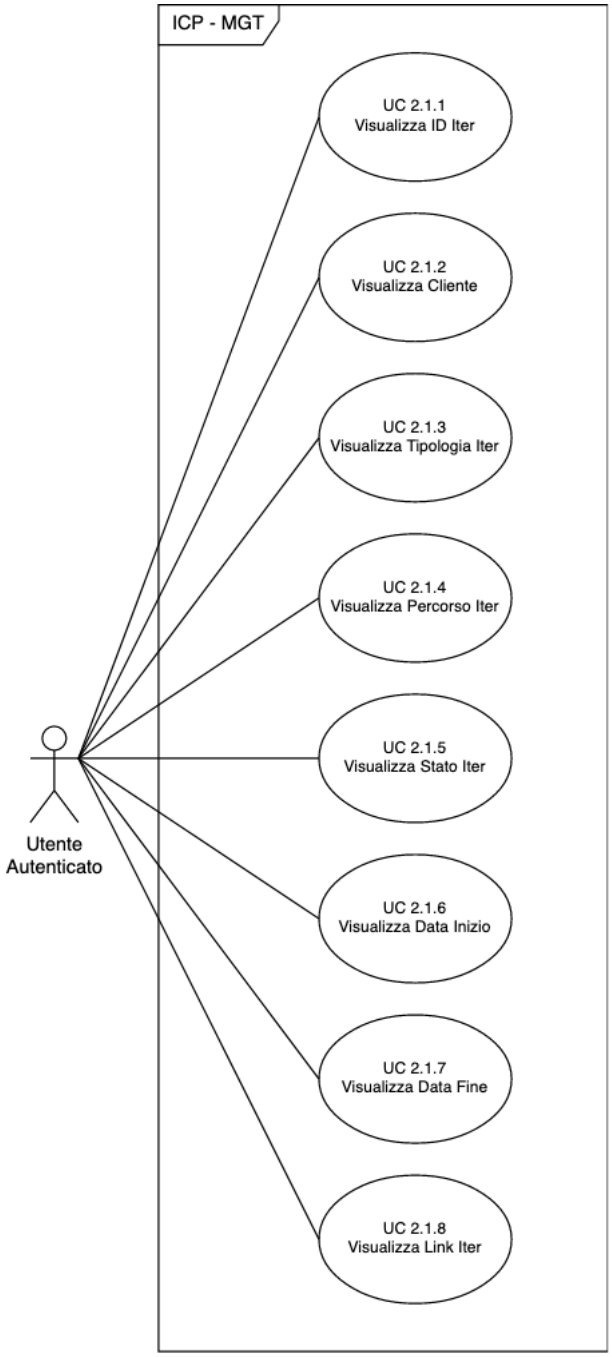


Figura 7: Sottocasi Visualizza Singolo Iter

2.1.10 UC 2.1.1 - Visualizza ID Iter

- **Attore primario**: Utente autenticato;
- **Descrizione**: L'utente autenticato nella piattaforma visualizza l'ID dell'iter del singolo iter;
- **Precondizione**: L'utente è autenticato nella piattaforma, si trova nella pagina relativa allo storico Iter e sta visualizzando un singolo iter presente nello storico;
- **Postcondizione**: L'utente autenticato visualizza l'ID del singolo Iter.
- **Scenario principale**:

1. L'utente ha effettuato l'accesso al sistema;
2. L'utente si trova nella pagina principale di gestione iter;
3. L'utente clicca sull'icona relativa allo "Storico Iter";
4. L'utente ha selezionato i filtri necessari per la ricerca;
5. Se la ricerca ha prodotto dei risultati, l'utente visualizza una lista di iter in base ai filtri inseriti;
6. L'utente sceglie di visualizzare un singolo iter presente nella lista dei risultati;
7. L'utente visualizza l'ID iter dello specifico iter.

2.1.11 UC 2.1.2 - Visualizza Cliente

- **Attore primario**: Utente autenticato;
- **Descrizione**: L'utente autenticato nella piattaforma visualizza la denominazione del cliente che ha avviato il singolo iter;
- **Precondizione**: L'utente è autenticato nella piattaforma, si trova nella pagina relativa allo storico Iter e sta visualizzando un singolo iter presente nello storico;
- **Postcondizione**: L'utente autenticato visualizza la denominazione del cliente che ha avviato il singolo Iter.
- **Scenario principale**:
  1. L'utente ha effettuato l'accesso al sistema;
  2. L'utente si trova nella pagina principale di gestione iter;
  3. L'utente clicca sull'icona relativa allo "Storico Iter";
  4. L'utente ha selezionato i filtri necessari per la ricerca;
  5. Se la ricerca ha prodotto dei risultati, l'utente visualizza una lista di iter in base ai filtri inseriti;
  6. L'utente sceglie di visualizzare un singolo iter presente nella lista dei risultati;
  7. L'utente visualizza la denominazione del cliente che ha avviato il singolo iter.

2.1.12 UC 2.1.3 - Visualizza Tipologia Iter

- **Attore primario**: Utente autenticato;
- **Descrizione**: L'utente autenticato nella piattaforma visualizza la tipologia del singolo iter;
- **Precondizione**: L'utente è autenticato nella piattaforma, si trova nella pagina relativa allo storico Iter e sta visualizzando un singolo iter presente nello storico;
- **Postcondizione**: L'utente autenticato visualizza la tipologia del singolo iter.
- **Scenario principale**:
  1. L'utente ha effettuato l'accesso al sistema;
  2. L'utente si trova nella pagina principale di gestione iter;
  3. L'utente clicca sull'icona relativa allo "Storico Iter";
  4. L'utente ha selezionato i filtri necessari per la ricerca;
  5. Se la ricerca ha prodotto dei risultati, l'utente visualizza una lista di iter in base ai filtri inseriti;
  6. L'utente sceglie di visualizzare un singolo iter presente nella lista dei risultati;
  7. L'utente visualizza la tipologia di quello specifico iter.

2.1.13 UC 2.1.4 - Visualizza Percorso Iter

- **Attore primario**: Utente autenticato;
- **Descrizione**: L'utente autenticato nella piattaforma visualizza il percorso di quel singolo iter;
- **Precondizione**: L'utente è autenticato nella piattaforma, si trova nella pagina relativa allo storico Iter e sta visualizzando un singolo iter presente nello storico;
- **Postcondizione**: L'utente autenticato visualizza il percorso del singolo iter.
- **Scenario principale**:
  1. L'utente ha effettuato l'accesso al sistema;
  2. L'utente si trova nella pagina principale di gestione iter;
  3. L'utente clicca sull'icona relativa allo "Storico Iter";
  4. L'utente ha selezionato i filtri necessari per la ricerca;
  5. Se la ricerca ha prodotto dei risultati, l'utente visualizza una lista di iter in base ai filtri inseriti;
  6. L'utente sceglie di visualizzare un singolo iter presente nella lista dei risultati;
  7. L'utente visualizza il percorso di quello specifico iter.

2.1.14 UC 2.1.5 - Visualizza Stato Iter

- **Attore primario**: Utente autenticato;
- **Descrizione**: L'utente autenticato nella piattaforma visualizza lo stato dell'iter nella pagina relativa alla visualizzazione dello storico iter;
- **Precondizione**: L'utente è autenticato nella piattaforma, si trova nella pagina relativa allo storico Iter e sta visualizzando un singolo iter presente nello storico;
- **Postcondizione**: L'utente autenticato visualizza lo stato di quello specifico iter.
- **Scenario principale**:
  1. L'utente ha effettuato l'accesso al sistema;
  2. L'utente si trova nella pagina principale di gestione iter;
  3. L'utente clicca sull'icona relativa allo "Storico Iter";
  4. L'utente ha selezionato i filtri necessari per la ricerca;
  5. Se la ricerca ha prodotto dei risultati, l'utente visualizza una lista di iter in base ai filtri inseriti;
  6. L'utente sceglie di visualizzare un singolo iter presente nella lista dei risultati;
  7. L'utente visualizza lo stato di quello specifico iter.

2.1.15 UC 2.1.6 - Visualizza Data Inizio

- **Attore primario**: Utente autenticato;
- **Descrizione**: L'utente autenticato nella piattaforma visualizza la data di inizio dell'iter nella pagina relativa alla visualizzazione dello storico iter;
- **Precondizione**: L'utente è autenticato nella piattaforma, si trova nella pagina relativa allo storico Iter e sta visualizzando un singolo iter presente nello storico;

- **Postcondizione**: L'utente autenticato visualizza la data in cui è stato avviato quello specifico iter.
- **Scenario principale**:
    1. L'utente ha effettuato l'accesso al sistema;
    2. L'utente si trova nella pagina principale di gestione iter;
    3. L'utente clicca sull'icona relativa allo "Storico Iter";
    4. L'utente ha selezionato i filtri necessari per la ricerca;
    5. Se la ricerca ha prodotto dei risultati, l'utente visualizza una lista di iter in base ai filtri inseriti;
    6. L'utente sceglie di visualizzare un singolo iter presente nella lista dei risultati;
    7. L'utente visualizza la data di inizio di quello specifico iter.

2.1.16 UC 2.1.7 - Visualizza Data Fine

- **Attore primario**: Utente autenticato;
- **Descrizione**: L'utente autenticato nella piattaforma visualizza la data di fine dell'iter nella pagina relativa alla visualizzazione dello storico iter;
- **Precondizione**: L'utente è autenticato nella piattaforma, si trova nella pagina relativa allo storico Iter e sta visualizzando un singolo iter presente nello storico;
- **Postcondizione**: L'utente autenticato visualizza la data in cui si è concluso/è stata fatta l'ultima modifica (nel caso in cui l'iter non è ancora terminato) a quello specifico iter.
- **Scenario principale**:
    1. L'utente ha effettuato l'accesso al sistema;
    2. L'utente si trova nella pagina principale di gestione iter;
    3. L'utente clicca sull'icona relativa allo "Storico Iter";
    4. L'utente ha selezionato i filtri necessari per la ricerca;
    5. Se la ricerca ha prodotto dei risultati, l'utente visualizza una lista di iter in base ai filtri inseriti;
    6. L'utente sceglie di visualizzare un singolo iter presente nella lista dei risultati;
    7. L'utente visualizza la data di fine di quello specifico iter.

2.1.17 UC 2.1.8 - Visualizza Link Iter

- **Attore primario**: Utente autenticato;
- **Descrizione**: L'utente autenticato nella piattaforma visualizza il link alla pagina di dettaglio dell'iter nella pagina relativa alla visualizzazione dello storico iter;
- **Precondizione**: L'utente è autenticato nella piattaforma, si trova nella pagina relativa allo storico Iter e sta visualizzando un singolo iter presente nello storico;
- **Postcondizione**: L'utente autenticato visualizza il link che rimanda alla pagina di dettaglio di quello specifico iter.
- **Scenario principale**:
    1. L'utente ha effettuato l'accesso al sistema;
    2. L'utente si trova nella pagina principale di gestione iter;

3. L'utente clicca sull'icona relativa allo "Storico Iter";
4. L'utente ha selezionato i filtri necessari per la ricerca;
5. Se la ricerca ha prodotto dei risultati, l'utente visualizza una lista di iter in base ai filtri inseriti;
6. L'utente sceglie di visualizzare un singolo iter presente nella lista dei risultati;
7. L'utente visualizza il link che rimanda alla pagina di dettaglio di quello specifico iter.

2.1.18 UCE 1 - Visualizza Errore Tipologia Iter

- **Attore primario**: Utente autenticato;
- **Descrizione**: L'utente autenticato nella piattaforma visualizza un errore nel caso in cui il microservizio associato alle tipologie da mostrare nel menù a tendina non risponde correttamente;
- **Precondizione**: L'utente è autenticato nella piattaforma, si trova nella pagina relativa allo storico Iter ed ha la possibilità di filtrare i risultati in base alla tipologia iter;
- **Postcondizione**: Viene visualizzato un errore a schermo nel caso in cui non ci siano tipologie associate; quindi, l'utente non è in grado di selezionare una voce né effettuare una ricerca per visualizzare lo storico iter.
- **Scenario principale**:
  1. L'utente ha effettuato l'accesso al sistema;
  2. L'utente si trova nella pagina principale di gestione iter;
  3. L'utente clicca sull'icona relativa allo "Storico Iter";
  4. All'utente viene mostrato un messaggio d'errore nel caso in cui il servizio associato alle tipologie non risponda correttamente;
  5. Per risolvere questo problema, l'utente dovrà ricaricare la pagina o contattare il servizio clienti della piattaforma.

2.1.19 UCE 2 - Visualizza Errore Stato Iter

- **Attore primario**: Utente autenticato;
- **Descrizione**: L'utente autenticato nella piattaforma visualizza un errore nel caso in cui il microservizio[G]associato agli stati da mostrare nel menù a tendina non risponde correttamente;
- **Precondizione**: L'utente è autenticato nella piattaforma, si trova nella pagina relativa allo storico Iter ed ha la possibilità di filtrare i risultati in base allo stato in cui si trova l'iter in quel momento;
- **Postcondizione**: Viene visualizzato un errore a schermo nel caso in cui non ci siano stati associati; quindi, l'utente non è in grado di selezionare una voce né effettuare una ricerca per visualizzare lo storico iter.
- **Scenario principale**:
  1. L'utente ha effettuato l'accesso al sistema;
  2. L'utente si trova nella pagina principale di gestione iter;
  3. L'utente clicca sull'icona relativa allo "Storico Iter";
  4. All'utente viene mostrato un messaggio d'errore nel caso in cui il servizio associato allo stato iter non risponda correttamente;
  5. Per risolvere questo problema, l'utente dovrà ricaricare la pagina o contattare il servizio clienti della piattaforma.

2.1.20 UCE 3 - Visualizza Errore nei Risultati

- **Attore primario**: Utente autenticato;
- **Descrizione**: L'utente è autenticato nella piattaforma e riceve un errore nel caso in cui la ricerca degli iter non è andata a buon fine;
- **Precondizione**: L'utente è autenticato nella piattaforma, si trova nella pagina relativa allo storico Iter ed ha la possibilità di filtrare i risultati;
- **Postcondizione**: L'utente ha impostato i filtri ed ha avviato la ricerca, ma non ha ottenuto alcun risultato.
- **Scenario principale**:
  1. L'utente ha effettuato l'accesso al sistema;
  2. L'utente si trova nella pagina principale di gestione iter;
  3. L'utente clicca sull'icona relativa allo "Storico Iter";
  4. All'utente viene mostrato un messaggio d'errore nel caso in cui il servizio associato alla ricerca degli iter non ritorna alcun risultato;
  5. Per risolvere questo problema, l'utente dovrà ricaricare la pagina o contattare il servizio clienti della piattaforma.

2.2 Tracciamento dei Requisiti

In questa sezione verranno tracciati i requisiti, in seguito ai casi d'uso studiati.

2.2.1 Notazione

Al fine di identificare i requisiti in maniera univoca, verranno utilizzate le seguenti notazioni:

R. {A}.{B}.{UC.X}

Dove:

- R sta per "requisito",
- A può essere:
  1. Obbligatorio,
  2. Desiderabile,
  3. Facoltativo.
- B può essere:
  1. F: Funzionale,
  2. V: di Vincolo,
  3. Q: di Qualità,
  4. P: di Prestazione.
- UC indica il numero del caso d'uso associato (e, eventualmente, con X si indica il sottocaso).

2.2.2 Requisiti Funzionali

Requisito	Descrizione	Classificazione	Fonti
R1F0	Il componente deve essere accessibile dalla Homepage della sezione dedicata allo storico iter	Obbligatorio	UC 0
R1F1	Nello storico iter deve essere presente il componente che permetta di visualizzare lo storico iter	Obbligatorio	UC 1
R1F1.1	Il componente deve permettere all'utente di effettuare la ricerca in base alla tipologia di iter da cercare	Obbligatorio	UC 1.1
R1FE1	Il componente deve mostrare un messaggio d'errore nel caso in cui la piattaforma non riesca a ritornare la lista di tipologie di iter da cercare	Obbligatorio	UCE 1
R1F1.2	Il componente deve permettere all'utente di effettuare la ricerca in base allo stato in cui si trova l'iter da cercare	Obbligatorio	UC 1.2

R1FE2	Il componente deve mostrare un messaggio d'errore nel caso in cui la piattaforma non riesca a ritornare la lista con gli stati relativi all'iter da cercare	Obbligatorio	UCE 2
R1F1.3	Il componente deve permettere all'utente di visualizzare il cliente che ha fatto partire l'iter	Obbligatorio	UC 1.3, UC 2.1.2
R2F1.4	Il componente deve permettere all'utente di effettuare la ricerca in base alla data di inizio dell'iter da cercare	Desiderabile	UC 1.4
R2F1.5	Il componente deve permettere all'utente di effettuare la ricerca in base alla data di fine dell'iter da cercare	Desiderabile	UC 1.5
R1FE3	Il componente deve mostrare un messaggio d'errore nel caso in cui la ricerca non produca alcun risultato	Obbligatorio	UCE 3
R1F2	Il componente deve permettere all'utente di visualizzare i risultati ottenuti dalla ricerca	Obbligatorio	UC 2
R1F2.1	Il componente deve permettere all'utente di visualizzare un singolo iter dai risultati ottenuti dalla ricerca	Obbligatorio	UC 2.1
R1F2.1.1	Il componente deve permettere all'utente di visualizzare l'ID del singolo iter	Obbligatorio	UC 2.1.1
R1F2.1.2	Il componente deve permettere all'utente di visualizzare il cliente che ha avviato il singolo iter	Obbligatorio	UC 2.1.2
R1F2.1.3	Il componente deve permettere all'utente di visualizzare la tipologia del singolo iter	Obbligatorio	UC 2.1.3
R2F2.1.4	Il componente deve permettere all'utente di visualizzare il percorso del singolo iter	Desiderabile	UC 2.1.4
R1F2.1.5	Il componente deve permettere all'utente di visualizzare lo stato del singolo iter	Obbligatorio	UC 2.1.5
R1F2.1.6	Il componente deve permettere all'utente di visualizzare la data di inizio in cui è stato avviato il singolo iter	Obbligatorio	UC 2.1.6
R1F2.1.7	Il componente deve permettere all'utente di visualizzare la data di fine del singolo iter	Obbligatorio	UC 2.1.7
R1F2.1.8	Il componente deve permettere all'utente di accedere al link per riprendere il singolo iter o visualizzare i dettagli nel caso in cui il singolo iter sia concluso	Obbligatorio	UC 2.1.8

Tabella 1: Requisiti Funzionali

2.2.3 Requisiti di Vincolo

Requisito	Descrizione	Classificazione
R1V1	Il componente dovrà essere sviluppato sfruttando il framework Angular 12	Obbligatorio
R1V2	È necessario sfruttare la libreria Angular Material per lo stile del componente	Obbligatorio
R1V3	È necessario sfruttare il linguaggio TypeScript per la parte logica del componente	Obbligatorio
R1V4	È necessario fare il push del codice utilizzando il repository aziendale	Obbligatorio
R3V5	È necessario utilizzare l'IDE Visual Studio Code	Facoltativo
R1V6	Il componente dovrà contenere dei dati scritti in lingua italiana	Obbligatorio

Tabella 2: Requisiti di Vincolo

2.2.4 Requisiti di Qualità

Requisito	Descrizione	Classificazione
R1Q1	È necessario realizzare il componente seguendo le indicazioni grafiche date dal tutor	Obbligatorio
R1Q2	Lo stagista dovrà rispettare i tempi di sviluppo prefissati e non superare la data di scadenza prevista	Obbligatorio
R1Q3	L'interfaccia grafica del componente realizzato dovrà essere accessibile e coerente con la grafica dell'intero applicativo	Obbligatorio
R1Q4	Il componente deve avere una grafica intuitiva	Obbligatorio
R1Q5	Il componente realizzato deve integrarsi con la parte back-end e fare in modo che i microservizi ad esso associati siano ben collegati	Obbligatorio
R1Q6	Il componente non dovrà contenere alcun bug	Obbligatorio

Tabella 3: Requisiti di Qualità

2.2.5 Requisiti Prestazionali

Requisito	Descrizione	Classificazione
R1P1	Il componente dovrà funzionare sui browser[G]più recenti	Obbligatorio
R1P2	È necessario avere JavaScript abilitato nel browser[G]per un corretto funzionamento della piattaforma web e del componente realizzato	Obbligatorio

Tabella 4: Requisiti Prestazionalli

3 Strumenti

3.1 Microsoft Visual Studio

Visual Studio Code (logo in figura 8) non è altro che un ambiente di sviluppo integrato (IDE[G]) multi linguaggio sviluppato da Microsoft (la sua prima versione risale al 1997), grazie al quale è possibile sviluppare applicazioni di qualsiasi dimensione e tipologia (ad esempio WebApp e Mobile App).

Inoltre, Visual Studio supporta numerosi linguaggi di programmazione (tra cui il TypeScript, l'HTML ed il CSS usati per la realizzazione del componente richiesto).

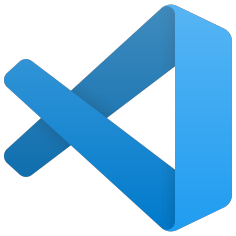


Figura 8: Il logo di Microsoft Visual Studio

Tra le tante funzionalità offerte da Visual Studio Code c'è anche il debugger, uno strumento utile per l'individuazione di errori nel codice (o bug).

Le motivazioni che hanno spinto alla scelta di Visual Studio Code come IDE[G]per lo sviluppo del progetto sono:

- **Uniformità**: anche gli altri membri del team, compreso il tutor, utilizzano Visual Studio Code come IDE[G], il che mi ha permesso di discutere di quanto svolto in maniera più rapida ed efficiente;
- **Costo**: a differenza di altri IDE[G], Visual Studio Code è completamente gratuito;
- **Supporto a diversi linguaggi di programmazione**: aspetto che mi ha permesso di sviluppare nei linguaggi scelti per la realizzazione del progetto e ne ha facilitato la fase di coding (grazie ai suggerimenti);
- **Integrazione con Git[G]**: Visual Studio Code offre la possibilità di gestire i repository[G]Git[G]ed il relativo versionamento[G].

Alcune alternative a Visual Studio sono:

- **Atom**: è un editor di testo ed IDE[G]open source sviluppato da GitHub e rilasciato nel 2014. Il motivo per cui non è stato scelto Atom rispetto a Visual Studio Code è che presenta meno funzionalità (è necessario installare dei plugin o delle estensioni per poter sfruttare appieno Atom) e su determinati sistemi operativi – ad esempio Windows – risulta essere meno performante;
- **WebStorm**: IDE[G]realizzato da JetBrains, software house[G]nata nel 2000 in Repubblica Ceca, pensato appositamente per lo sviluppo web (in particolar modo, per JavaScript e tecnologie affini). Lo svantaggio di WebStorm è il fatto di essere a pagamento;

- **Sublime Text**: si tratta di un editor di testo veloce, leggero e particolarmente indicato per molti linguaggi (tra cui quelli di Markup[G] come: HTML o LaTex); tuttavia, è molto limitato perché non offre tutte le funzionalità di un IDE[G], ma, semplicemente, permette di editare il codice.

Sebbene non avessi mai utilizzato Visual Studio prima, questo si è rivelato una scelta molto azzeccata e, grazie anche ai suggerimenti, mi ha permesso di sviluppare in autonomia, nonostante non avessi piena conoscenza della sintassi dei linguaggi richiesti.

3.2 GitLab

GitLab (logo in figura 9) è una piattaforma web open source che permette di creare dei repository[G] pubblici o privati, nei quali gli sviluppatori possono caricare il proprio codice e gestire le modifiche alle varie versioni in contemporanea, aspetto essenziale per chi lavora in team.

Infatti, mediante GitLab più persone possono lavorare in maniera asincrona ed autonoma ad uno stesso progetto senza generare conflitti, oltre a caricare ed unire le modifiche fatte da tutti i componenti in un unico progetto.

Figura 9: Il logo di GitLab

Come altri sistemi di controllo di versione[G] (VCS), tra le tante funzionalità GitLab permette di effettuare le seguenti operazioni (ampiamente utilizzate per il il versionamento[G] del repository[G] su cui ho lavorato per ICP):

- **Pull**, scaricare in locale il codice presente in un repository[G];
- **Push**, caricare sul repository[G] le modifiche effettuate in locale;
- **Merge[G]**, unire le modifiche apportate ad un branch[G].

Ad ogni commit viene avviato un servizio di integrazione continua[G], il quale crea automaticamente una build ed avvia dei test di unità sulle modifiche più recenti del codice, al fine di individuare eventuali errori.

Ho utilizzato quotidianamente GitLab per la realizzazione del componente e per salvare in remoto (sul repository[G] del FrontEnd[G]) le modifiche fatte; ciò ha permesso al tutor di monitorare tutti i miei interventi e di verificare che quanto fatto non creasse conflitti con quanto già svolto.

Ad esempio: sono state modificate delle classi nel foglio di stile comune a tutti i file, tramite l'integrazione continua[G] è stato possibile verificare - in maniera automatica - che quelle modifiche non andassero in conflitto con quanto scritto dal tutor sullo stesso file nello stesso periodo.

Infine, il salvataggio delle versioni del codice su GitLab permette di ricostruire l'intera storia del codice ed, eventualmente, ripristinare vecchie versioni (aspetto molto interessante per confrontare il vecchio codice con le nuove modifiche).

3.3 Microsoft Azure

Microsoft (logo in figura 10) è una piattaforma cloud realizzata da Microsoft che offre servizi di cloud computing[G].

Figura 10: Il logo di Microsoft Azure

Con questo servizio di Microsoft vengono erogati servizi appartenenti a diverse categorie, come:

- Risorse di elaborazione,
- Archiviazione e memorizzazione dati,
- Trasmissione dati,
- Interconnessione di reti,
- Analisi,
- Intelligence,
- Apprendimento automatico,
- Sicurezza.

Tuttavia, il numero di servizi erogati viene modificato con cadenza periodica a seconda dell'utilizzo e di eventuali aggiornamenti.

Le aree su cui operano i servizi messi a disposizione di Microsoft Azure sono tre e si differenziano dalla modalità di erogazione adottata:

- Infrastructure as a Service (IaaS),
- Platform as a Service (PaaS),
- Software as a Service[G](SaaS) - modello utilizzato per "Inventory Chain Platform".

Sopra Steria oltre a sfruttare la suite di Microsoft 365, ha attivato anche un piano Azure come servizio di cloud computing[G]che si utilizza per ICP.

In particolare, i database di ICP si trovano su Azure, così come i documenti che vengono caricati e scaricati dai vari attori che sfruttano la piattaforma.

3.4 Keycloak

Keycloak (logo in figura 11) è un software open source pensato per gestire l'accesso, tramite single sign-on, ad una applicazione o servizio.

Figura 11: Il logo di Keycloak

Mediante Keycloak è possibile assegnare o meno determinati ruoli e "coni di visibilità[G]" ad un utente: ciò significa che, per quanto riguarda lo "Storico Iter", solo se un un utente ha il ruolo associato alla lettura degli iter, può consultare lo storico, altrimenti no.

Lo stesso vale per i coni di visibilità[G]: se su keycloak non vengono associati i "tenant[G]" relativi agli iter, l'utente che ha effettuato l'accesso alla piattaforma non può fare operazioni né accedere allo storico iter.

3.5 Jira

Jira (logo in figura 12) è una suite di strumenti software realizzati per il tracciamento delle segnalazioni e sviluppato da Atlassian; software house[G]australiana fondata nel 2002 a Sydney, che realizza prodotti per sviluppatori di software, project management e per la gestione dei contenuti.

Figura 12: Il logo di Jira

Tra gli strumenti offerti da Jira, Sopra Steria utilizza l'omonimo software Jira per la gestione dei ticket e per verificare lo stato di avanzamento del progetto. I principali vantaggi offerti da Jira, che permettono di lavorare in team in maniera efficace ed efficiente sono:

- **Monitoraggio attività** (o Kanban Board[G]): tramite la dashboard di Jira è possibile verificare lo stato di avanzamento del progetto sia in termini generali che di dettaglio (ad esempio tramite la checklist presente in un ticket), poiché ciascun ticket è assegnato ad una o più persone e per ogni ticket creato è possibile modificarne lo stato di avanzamento;
- **Creazione di ticket**: permette di assegnare compiti ai vari membri del team di un progetto. Questa funzionalità è molto utile, perché permette a tutti i membri del team di realizzare una lista di attività da svolgere, conoscere il relativo stato di avanzamento, indicarne la priorità ed il tempo di esecuzione stimato;
- **Allegare documenti**: in ciascun ticket è possibile allegare documenti di vario tipo (vedi file in formato JSON), il che permette di avere un ambiente di lavoro ordinato e fruibile da chiunque abbia necessità di trovare informazioni;

- **Attivare sprint**: che consiste nell'attivare degli slot temporali (noti anche con il nome di "time box") che definiscono il periodo massimo in cui è possibile lavorare ad un determinato obiettivo (ad esempio il rilascio di una versione dell'applicazione che si sta sviluppando). All'interno di ciascuno sprint è possibile associare singole attività (come task, user story, spike e tutte quelle mansioni relative alla metodologia agile).

Altre alternative a Jira:

- **Trello**: si tratta di un software di project management acquistato da Atlassian, software house[G]che ha prodotto anche Jira, nel 2017. La differenza tra i due software è il target di riferimento; Trello è pensato per la pianificazione di attività poco complesse e più generiche, nella quale partecipano figure professionali con diverse specializzazioni, mentre Jira è più improntato per aziende di dimensioni medio-grandi e per la gestione di progetti IT più nel dettaglio;
- **ClickUp**: è simile agli altri strumenti, ma pensato per team composti da poche persone per qualsiasi tipologia di progetto, mentre Jira è più improntato per lo sviluppo software e per la modalità di sviluppo agile. Essendo uno strumento "cross" per diverse tipologie di business, offre numerose funzionalità per qualsiasi tipologia di progetto, tra cui: calendario, possibilità di caricare documenti e mappe, gestione di eventi e relativi alert, creare fogli di calcolo, diagrammi di Gantt, automazioni, to-do list, sprint, oltre a permettere ai team di avere una conversazione attiva attraverso la chat presente nell'applicazione stessa di Click Up.

Ho utilizzato quotidianamente Jira per inserire nuove attività che mi venivano assegnate e marcare quelle svolte, monitorare il flusso di lavoro, assegnare nuovi ticket ad altri membri del team (come nel caso della creazione del microservizio[G]per lo Storico Iter, nel quale ho allegato un mockup[G]di request e response) e verificare l'andamento del progetto.

3.6 Microsoft 365

Sopra Steria utilizza la Suite di Microsoft 365 in versione aziendale per la gestione del lavoro (logo in figura 13).

Figura 13: Il logo di Microsoft 365

Gli strumenti di supporto realizzati da Microsoft 365 utilizzati per il progetto e per l'allineamento interno con tutti i membri del team sono stati:

- **Microsoft Teams**, una piattaforma di comunicazione e collaborazione unificata, che permette a diversi membri di un'organizzazione di comunicare – tramite chat, chiamate o videochiamate - e scambiare file in tempo reale. A differenza delle e-mail, permette una comunicazione più rapida, veloce ed informale;
- **Outlook** è un servizio di posta elettronica via web, tramite il quale è possibile inviare e ricevere messaggi di posta, pianificare riunioni e gestire il proprio calendario lavorativo;
- **Power Point** per visualizzare presentazioni come quelle relative al progetto;
- **Excel** come foglio di calcolo per la produzione e la gestione dei fogli elettronici. Excel è stato usato per la visualizzazione delle schede cliente;

- **OneDrive**, si tratta di un servizio di cloud storage e backup offerto da Microsoft, fruibile da browser[G]o applicazione. Quest'ultimo servizio è stato utilizzato per consultare i documenti inerenti al progetto, all'azienda e come servizio di archiviazione dati personale.

3.7 Swagger

Swagger (logo in figura 14) è una suite di strumenti per sviluppatori realizzati dall'azienda americana SmartBear Software e, fondamentalmente, permettono di visualizzare le interfacce delle API[G].

Per quanto concerne il componente che ho realizzato durante il periodo di tirocinio, ho utilizzato lo Swagger Editor, l'editor della piattaforma swagger.io, per visualizzare request e response dei microservizi.

Figura 14: Il logo di Swagger

In figura 15 è possibile visualizzare il corpo di request e response del servizio che si occupa di estrapolare lo storico iter dati dei filtri.

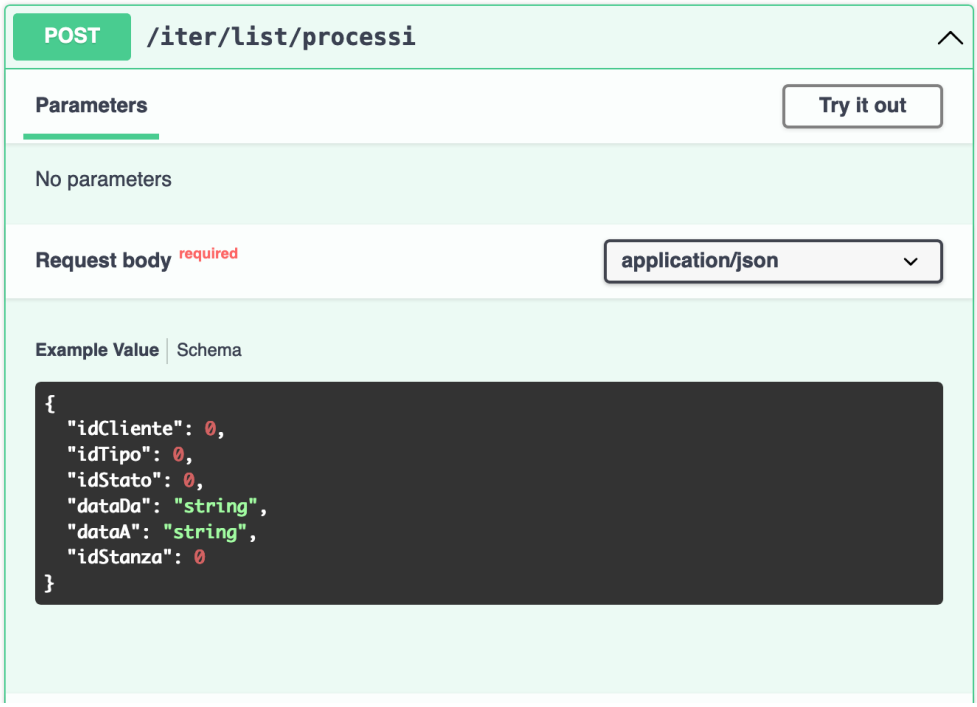


Figura 15: Body della request, ossia dei dati da passare al microservizio[G]

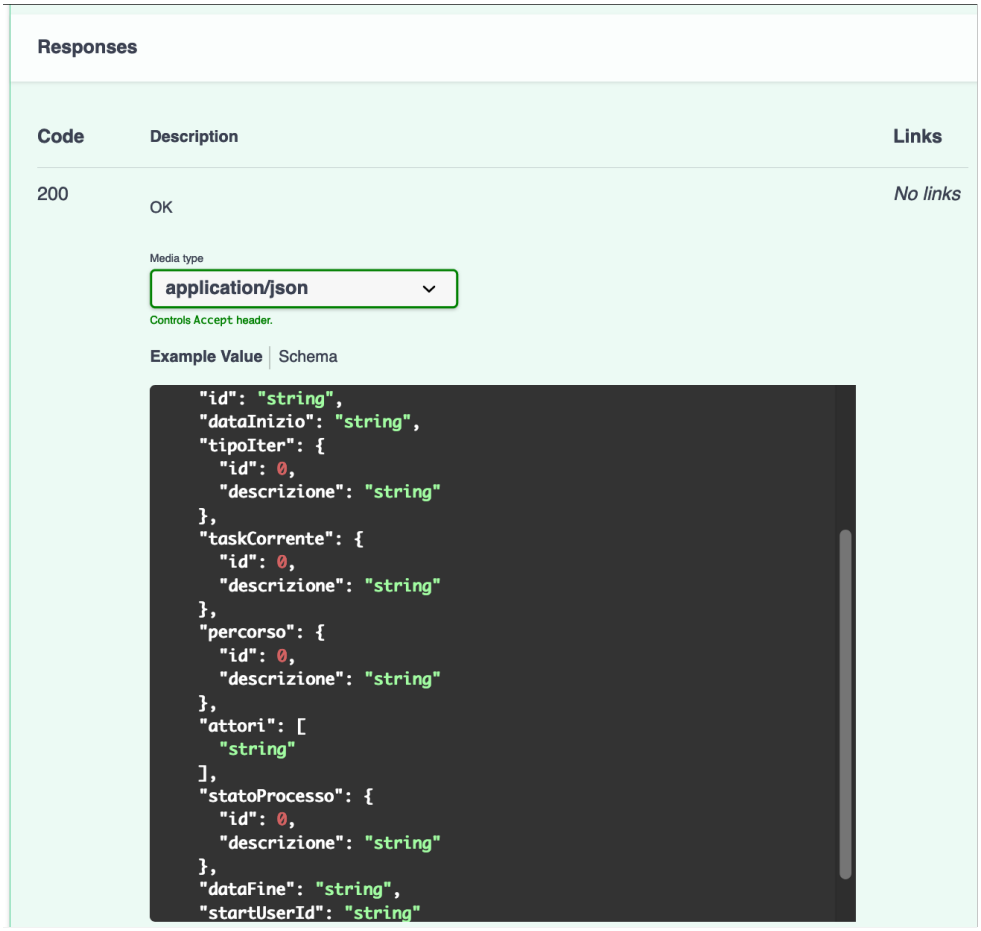


Figura 16: Body della response, ossia dei dati ritornati dal microservizio[G]

Le attività svolte per far funzionare e sfruttare la potenzialità di swagger a fini progettuali sono:

1. Una volta pronto un nuovo microservizio[G], nel caso in cui fosse stato creato un nuovo oggetto o aggiornata l'interfaccia di un oggetto preesistente lato Back-end, è necessario scaricare lo swagger (in formato "YAML[G]") generato in fase di "install" della parte BE e relativo al microservizio[G], che contiene la definizione delle interfacce di tutte le API[G]contenute all'interno di esso;
2. A questo punto, tramite l'editor di swagger è possibile visualizzare l'interfaccia della request (ossia, la struttura dell'oggetto da inviare per chiamare il servizio - come mostrato in Figura 15) e della response (l'oggetto ritornato dal servizio chiamato - come mostrato in figura 16) nel caso in cui il servizio chiamato risponda correttamente (codice 200), oltre ad eventuali errori ritornati dai codici di stato nel caso in cui il servizio vada in "bad request[G]" (esempio: 500, 400, 404 o, più in generale, 40x).

L'editor di swagger è risultato molto utile perché mi ha permesso di avere una traccia su come definire la struttura della request e di cosa aspettarmi dalla response, oltre a verificare che i dati trasmessi e ricevuti rispettassero le richieste.

3.8 Postman

Postman (logo in figura 17) è una nota piattaforma API[G]per creare, testare e progettare API[G]. Lato Front-end[G], non ho fatto grande uso di Postman, ma è stato utile per provare il corretto funzionamento di request e response passate ai microservizi.

Figura 17: Il logo di Postman

Per quanto riguarda ICP, ho utilizzato postman per capire nel dettaglio quali valori venivano ritornati da un microservizio[G] (come la chiamata al metodo GET che ritornava gli stati associati all'iter o la tipologia dei processi, dati presi dal database).

Inoltre, Postman mi è servito - in maniera marginale - per verificare il corretto funzionamento dei dati inviati al microservizio[G]della ricerca (un metodo POST).

In figura 18 è presente una schermata d'esempio di Postman con la chiamata al microservizio[G] che ritorna la lista degli iter da mostrare nello storico (nella request è stato inserito l'idStanza associato ad un cliente).

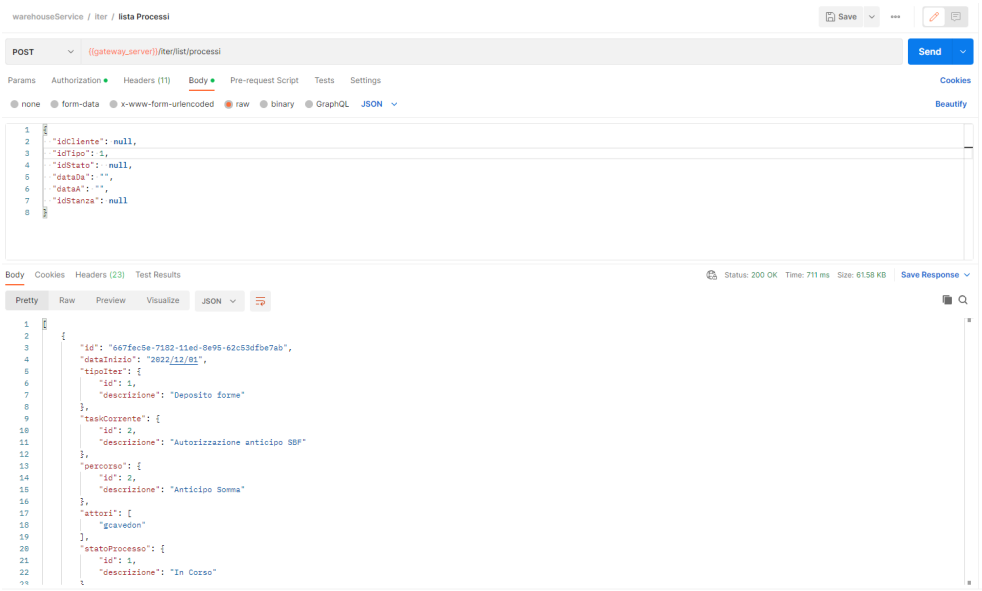


Figura 18: Request e Response del microservizio[G]che ritorna lo storico iter tramite Postman

4 Tecnologie utilizzate

In questo paragrafo verranno descritte le tecnologie utilizzate in fase di sviluppo del progetto e relative funzionalità.

4.1 <angular/>

Angular (logo in figura 19) è un framework open source sviluppato principalmente da Google ed utilizzato per lo sviluppo di applicazioni web con licenza MIT.

Figura 19: Il logo di Angular

La storia di Angular inizia nel 2008-2009, quando Miško Hevery (sviluppatore presso Google) ed il suo amico Adam Abrons iniziarono a lavorare part-time su un progetto il cui scopo originale era quello di semplificare il lavoro ai web designer, i quali non avevano competenze in materia di sviluppo web.

A tal proposito, Hevery e Abrons decisero di realizzare questa nuova tecnologia per la creazione di applicazioni (Web APP) e pagine Web, basata sul modello architetturale Model View Controller (MVC) e sul linguaggio open source JavaScript (la cui prima versione risale al Dicembre 1995), poiché all'epoca lo sviluppo web era dominato dai framework Java (Spring, JSF, Struts, JSP) e .NET, linguaggi che rendevano tutt'altro che semplice lo sviluppo di interfacce web.

Oltre alla difficoltà legata all'implementazione, ad ogni modifica era necessario eseguire nuovamente l'applicazione, per visualizzare i componenti realizzati, il data binding e la validazione di form.

Una volta messo in piedi il progetto, i due autori del framework decisero di realizzare il progetto e registrare il dominio "getangular.com" (in quanto "angular.com" all'epoca non era disponibile) ed all'interno di questo sito web era possibile trovare una sorta di guida con dimostrazioni sul funzionamento di questa nuova tecnologia.

Tuttavia, nemmeno Hevery aveva una visione generale di cosa ne sarebbe uscito da questo progetto e quando gli fu chiesto quale fosse l'utilità di Angular, lui rispose così:

"*It's a kind of Spread sheet in cloud so that we can bind data and save it without worrying about security persistent etc and UI will be in HTML*".

In figura 20 viene illustra la pagina iniziale del sito web GetAngular.com.

Figura 20: Homepage del sito getangular.com - Fonte: aulab.it

4.1.1 La sfida ed il primo rilascio

Successivamente, a Hevery venne chiesto di lavorare ad un progetto denominato "Google Feedback Tool[G]" (GWT), ma dopo sei mesi di sviluppo divenne complesso lavorare su oltre 17.000 righe di codice - specie per la difficoltà nel testare i file generati dai sorgenti realizzati con dei framework che si basavano su Java per generare file HTML e JavaScript.

Di conseguenza, Miško chiese di riscrivere il codice sfruttando Angular: aveva promesso che avrebbe riscritto tutto il codice in sole due settimane.

Purtoppo la promessa di Miško non fu rispettata, poiché furono necessarie ben tre settimane per portare a termine il lavoro, ma le righe di codice si ridussero a 1500.

Questo fatto impressionò il manager di Hevery - Brad Green -, che chiese di lavorare ulteriormente al progetto sfruttando Angular - tecnologia che, per l'occasione, fu rinominata in "AngularJS" (o Angular 1.x).

Data l'efficienza di questa nuova tecnologia, Brad Green decise di rilasciare AngularJS il 20 Ottobre 2010: AngularJS era il primo framework JavaScript ad introdurre la Dependency Injection ed il focus era HTML5 (rilasciato a Gennaio 2008) con JS, anziché l'HTML prodotto da JavaScript.

4.1.2 L'avvento di Angular puro

Nell'ottobre 2014 è stata annunciata una versione nuova e completamente riscritta di AngularJS, ossia Angular 2, la prima di una lunga serie di rilasci (ad oggi siamo alla versione 14 rilasciata lo scorso 2 Giugno 2022).

Nonostante la somiglianza nel nome, l'allora nuova versione ed Angular JS hanno avuto, sin da subito, sostanziali differenze.

Sebbene anche Angular 2 fosse un framework open source pensato per lo sviluppo di applicazioni web e sviluppato dallo stesso Google, denotava differenti caratteristiche rispetto al framework padre:

- Il linguaggio di programmazione utilizzato è diverso: anziché utilizzare JavaScript, Angular 2 è stato sviluppato sfruttando **TypeScript** (§4.2), un linguaggio open source di Microsoft la cui prima versione risale all'Ottobre 2012. TypeScript è sempre stato ritenuto un "superset[G]"

di JavaScript (o un JavaScript "tipizzato"), in quanto amplia la suite di quest'ultimo con l'aggiunta di tipi statici, classi, interfacce e generics;

- Angular 2 sfrutta un'architettura basata sul pattern architetturale **Model View View Model** (MVVM), con una View che è in grado di gestire gli eventi, eseguire operazioni ed effettuare il data-binding;
- Angular CLI: si tratta di una interfaccia a riga di comando (o "command line interface") di Angular, la quale permette di eseguire da terminale un insieme di comandi di supporto per la creazione del progetto, oltre a dei componenti per il deploy ed i test;
- È possibile creare applicazioni SEO-Friendly (ottimizzate per i motori di ricerca e relativo posizionamento nella classifica della ricerca);
- RxJS: viene introdotta la libreria per gestire eventi asincroni tramite l'uso del pattern Observer;
- Mobile Friendly[G]: in quanto è possibile creare interfacce grafiche sia per applicazioni web che mobile per qualsiasi sistema operativo (iOS, Android, MacOS, Linux e Windows);
- Generics: sono degli "stampini" (o modelli di codice) che favoriscono il riutilizzo del codice, che permettono, ad esempio, ad una funzione di essere utilizzata con tipi diversi (come numeri e stringhe), senza dover essere riscritta in base al tipo.

Inoltre, rispetto ad AngularJS, questa nuova versione vanta diversi cambiamenti sostanziali in termini di sintassi:

- Vengono aggiornati gli **Hooks[G]** (come: @NgModule, @Component, @Directive, @Service, che prima si scrivevano con "ng-module"), ossia eventi che servono per gestire il ciclo di vita di un componente;
- Introduzione delle parentesi quadrate "[ ]" per il data-binding delle proprietà e le parentesi tonde "( )" per il binding degli eventi.

Infine, con la tipizzazione statica ad ogni dato viene associato un tipo, indicato espressamente in fase di dichiarazione e tale tipo deve essere rispettato anche in fase di assegnazione successiva.

4.1.3 Angular Material

Angular Material è una libreria che permette di utilizzare delle interfacce utente (User Interface) "pre-confezionate" da personalizzare ed implementare in un progetto. Il vantaggio di usare Angular Material è che permette di velocizzare lo sviluppo, offrendo soluzioni eleganti, riutilizzabili ed efficaci.

Tra i tanti componenti di Angular Material sfruttati (specie per capire meglio le funzionalità delle tecnologie scelte) durante il mio tirocinio sono: Form Field, Card, Data Tables, List, Input, Radio Button ed il Paginator.

Oltre alla semplicità d'uso e di apprendimento, Angular Material è una libreria piuttosto usata dagli sviluppatori, aspetto che mi ha permesso di trovare soluzioni a problemi che sono sorti in fase di sviluppo in tempi molto rapidi (è possibile discutere e chiedere aiuto in merito ad Angular Material ed alle sue componenti su StackOverflow, Discord, Gruppi Google, Twitter, Reddit ed altri forum).

4.2 TypeScript

TypeScript (logo in figura 21) è un linguaggio di programmazione open source pensato per lo sviluppo front-end[G]di applicazioni ideato da Microsoft e rilasciato per la prima volta il primo Ottobre 2012. Ad oggi (17 Giugno 2022) siamo arrivati alla versione 4.7.4.

Figura 21: Il logo di Postman

Si tratta di un'estensione del linguaggio di programmazione orientato agli eventi e comunemente usato per la programmazione web JavaScript; TypeScript estende la sintassi di quest'ultimo e ne aggiunge la tipizzazione stretta, mantenendo invariato il funzionamento (qualunque programma realizzato in JavaScript è in grado di funzionare anche con TypeScript senza alcuna modifica).

In questo modo, per coloro che sono abituati ad altri linguaggi ad oggetti (Java, C++ etc.), non devono rinunciare ai tipi.

Anche TypeScript è stato progettato per la realizzazione di applicazioni ed è destinato ad essere compilato in JavaScript per poter essere interpretato da qualsiasi browser web[G]o applicazione.

Ad esempio, per quanto riguarda la realizzazione del componente relativo allo storico iter, per chiamare il microservizio[G]di ricerca, è stato necessario realizzare un mapper con TypeScript che realizzasse l'oggetto richiesto dalla request con i dati selezionati ed inseriti dall'utente.

Il codice del mapper è il seguente:

```
mapperFromFilterToService(filters: FormGroup): ElencoProcessiFiltratiInput {
    let tipologia = filters.controls['tipologia'].value?.id || undefined;
    let stato = filters.controls['stato'].value?.id || undefined;
    let cliente = this.stanza.cliente?.id;
    let dataDaForm = filters.controls['dataDa'].value;
    let dataDa;
    let dataA;
    if (dataDaForm) {
      dataDa = dataDaForm.getFullYear() + '/' + (dataDaForm.getMonth() + 1) + '/' +
          dataDaForm.getDate();
    } else {
      dataDa = null
    }

    let dataAForm = filters.controls['dataA'].value;
    if (dataAForm) {
      dataA = dataAForm.getFullYear() + '/' + (dataAForm.getMonth() + 1) + '/' +
          dataAForm.getDate();
    } else {
      dataA = null;
    }
    return {
      idCliente: cliente,
      idTipo: tipologia,
      idStato: stato,
      dataDa: dataDa ? dataDa : "",
      dataA: dataA ? dataA : "",
      idStanza: this.stanza.id,
    }
  }
```

5 Progettazione e codifica

In questo paragrafo verranno trattati argomenti relativi alla progettazione e codifica del componente realizzato.

5.1 Architettura Generale

Angular si basa sul pattern architetturale "MVVM" ideato dagli architetti di Microsoft, Ken Cooper e Ted Peters, ed è una variante del pattern MVP di Martin Fowler che ha come obiettivo quello di semplificare la programmazione guidata dagli eventi per interfacce grafiche. Il MVVM non è altro che una rifinitura del pattern "MVC", nel quale la parte di View-Model (VM) facilita la separazione dello sviluppo della parte grafica.

Infatti, il View Model è responsabile della conversione dei dati ricevuti dal modello, di modo che questi possano essere gestiti e presentati nella vista.

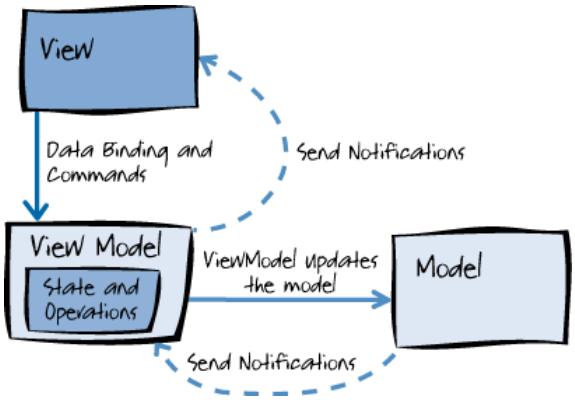


Figura 22: L'architettura Model View View-Model

Come illustrato in Figura 22, il pattern è composto da tre componenti ed il binder:

- **Model** (modello): il Model rappresenta il punto di accesso ai dati, che possono essere letti da un database o un servizio web, e la logica di business;
- **View** (vista): la vista è responsabile della definizione della struttura, il layout e dell'interfaccia grafica che l'utente vede nello schermo. Questa parte non contiene alcuna logica di business, ma il suo unico scopo è renderizzare (in termini grafici) un componente e quanto esposto dal View-Model (come riflettere i suoi cambiamenti di stato o attivare comportamenti);
- **View-Model** (modello della vista): il view-model è il punto d'incontro tra vista e modello, è responsabile per la gesione della logica della vista. In linea generale, questa parte interagisce con il modello invocando metodi presenti nel model. Il view-model fornisce, quindi, i dati dal modello in una forma che la vista può usare facilmente;
- **Binder** (legante): il binder è il meccanismo essenziale del MVVM, poiché permette al view-model ed alla vista di essere costantemente sincronizzati. Ciò implica che le modifiche ai dati apportate dall'utente attraverso la view, verranno automaticamente riportate nel view-model e senza che questo onere lo debba fare il programmatore. Analogamente, le modifiche apportate ai dati contenuti nel view-model verranno aggiornati automaticamente nella view.

5.2 Observable

Un aspetto molto importante da non tralasciare per quanto riguarda la parte di progettazione sono gli **Observable** (ossia, il deisng pattern Observer), che Angular implementa mediante la libreria RxJS.

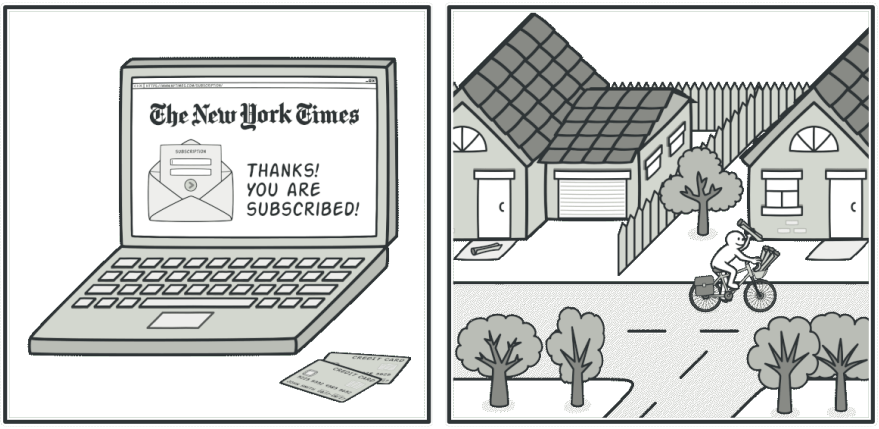


Figura 23: Una analogia dell'uso del design pattern Observer - Fonte: refactoring.guru

Come illustrato in Figura 23, il pattern Observable permette a degli utenti, in questo caso un utente (l'Observer o il Subscriber) che decide di iscriversi alla newsletter di un sito web, di osservare il comportamento di qualcuno o qualcosa, nel caso della posta elettronica, l'utente osserva la casella di posta in attesa che arrivi una email. L'operazione che deve compiere l'utente per ottenere la newsletter (che è l'oggetto osservato, ossia l'Observable) nella propria casella di posta elettronica è detta "Subscribe" (in italiano "sottoscrivere").

RxJS implementa questa funzionalità, che nella realizzazione del componente in oggetto (l'osservatore che apre una "subscribe" verso i microservizi indicati in §5.3.3) viene utilizzata per:

1. Attendere la response della chiamata al MS che ritorna la lista con le tipologie di iter;
2. Attendere la response della chiamata al MS che ritorna la lista con gli stati in cui può trovarsi un iter;
3. Attendere la response della chiamata al MS che ritorna la lista di iter richiesti.

5.3 Scelte Progettuali

È possibile suddividere le scelte progettuali fatte per la realizzazione del componente come segue:

- Interfaccia grafica,
- Servizi relativi al corretto funzionamento dello stesso.

Entrambi gli aspetti verranno analizzati nel dettaglio nei prossimi paragrafi.

5.3.1 Interfaccia Grafica

In accordo con il tutor, per l'interfaccia grafica del componente da realizzare è stato scelto di utilizzare la libreria **Angular Material** ed alcuni suoi componenti.

All'interno del componente mat-form-field, come si evince dalla figura 24, sono stati usati:

- Due componenti mat-select (per la selezione della tipologia (§2.1.3 UC 1.1) e lo stato dell'iter (§2.1.4 UC 1.2)), dal momento che le opzioni tra cui scegliere erano già presenti nel database; quindi, è stato sufficiente trovare un componente che permettesse all'utente di effettuare la selezione di una voce dalla lista di opzioni ritornate dal microservizio[G](il mat-select è da preferire rispetto altri blocchi, come potrebbe essere il radio-button, poiché è una soluzione semplice, intuitiva, pulita, efficiente ed ideale per liste con più di tre/quattro voci);
- Due componenti mat-datepicker, che permettono di selezionare una data (comprensiva di giorno, mese ed anno) da un calendario. Un datepicker è stato utilizzato per la scelta della data di inizio dell'iter, mentre un altro per la data di fine. È stato scelto di utilizzare questo "toggle[G]", rispetto ad un campo di input, per facilitare la scelta all'utente: in questo modo, la probabilità di sbagliare ad inserire una data è più bassa rispetto ad un campo di testo bianco. Inoltre, questo componente è stato usato anche in altri form presenti nella piattaforma; perciò, mantiene la coerenza con il resto dell'applicazione.

È stato utilizzato anche un campo di input per l'inserimento del cliente (in questo caso il caseificio che ha prodotto le forme), nel quale è possibile digitare il nome del produttore. Questo campo è presente nel form, ma è stato disabilitato, poiché il valore inserito è univoco e proviene dal percorso da cui si arriva quando si apre lo storico iter (non è possibile scegliere il produttore, in quanto viene scelto nelle pagine precedenti allo storico).

Il codice scritto in HTML che sfrutta i componenti di angular material per la realizzazione del form è il seguente:

```
<div class="main-container container-fluid cont-padding">
    <div class="row filter-table">
        <mat-accordion class="headers-align">
            <mat-expansion-panel expanded>
                <mat-expansion-panel-header>
                    <mat-panel-title>
                        Filtri
                    </mat-panel-title>
                    <mat-panel-description>
                         
                        <mat-icon>filter_alt</mat-icon>
                    </mat-panel-description>
                </mat-expansion-panel-header>
                <form [formGroup]="filterGroup">
                    <div class="row">
                        <div class="col">
                          // Qui inizia il menu' a tendina per selezionare la tipologia di
                              iter
                           <mat-form-field>
                              <mat-label>Tipologia Iter</mat-label>
                              <mat-select id="tipologia" formControlName="tipologia">
                                  <mat-option [value]="null"></mat-option>
                                  <mat-option class="drop-field" [value]="tipologia"
                                      *ngFor="let tipologia of
                                          listaTipologieProcessi">{{tipologia.descrizione}}
                                  </mat-option>
                              </mat-select>
```

```
        </mat-form-field>
    </div>
    <div class="col">
      // Qui inizia il menu' a tendina per selezionare lo stato in cui
          si puo' trovare l'iter
        <mat-form-field>
            <mat-label>Stato Iter</mat-label>
            <mat-select id="stato" formControlName="stato">
                <mat-option [value]="null"></mat-option>
                <mat-option class="drop-field" [value]="stato"
                    *ngFor="let stato of
                        listaStatiProcessi">{{stato.descrizione}}</mat-option>
            </mat-select>
        </mat-form-field>
    </div>
    <div class="col">
      // Qui inizia il campo di input nel quale viene inserita la
          denominazione del cliente
        <mat-form-field>
            <mat-label>Cliente</mat-label>
            <input matInput #cliente type="text" maxlength="100"
                value={{stanza.cliente?.descrizione}} [disabled]="true">
        </mat-form-field>
    </div>
</div>
<div class="row">
    <div class="col">
      // Qui inizia la parte relativa alla finestra nella quale e'
          possibile selezionare la data di inizio iter
        <mat-form-field>
            <mat-label>Iter Da</mat-label>
            <input matInput [matDatepicker]="dpDa" placeholder="Iter Da"
                formControlName="dataDa"
                (dateChange)="setMaxDateA($event)" [min]="minDateDa"
                    [max]="maxDate">
            <mat-datepicker-toggle matSuffix
                [for]="dpDa"></mat-datepicker-toggle>
            <mat-datepicker #dpDa></mat-datepicker>
        </mat-form-field>
    </div>
    <div class="col">
 // Qui inizia la parte relativa alla finestra nella quale e' possibile
     selezionare la data di fine iter
        <mat-form-field>
            <mat-label>Iter A</mat-label>
            <input matInput [matDatepicker]="dpA" placeholder="Iter A"
                formControlName="dataA"
                [min]="minDate" [max]="maxDate">
            <mat-datepicker-toggle matSuffix
                [for]="dpA"></mat-datepicker-toggle>
            <mat-datepicker #dpA></mat-datepicker>
        </mat-form-field>
    </div>
</div>
<mat-divider></mat-divider>
<div class="row actionButton">
    <div class="col-md-12">
        <div class="pull-right">
          // Bottone di ricerca, disponibile sse e' stato selezionato
              almeno un campo del form
            <button mat-button class="form-btn" class="float-right"
```

```
                                (click)="resetFiltri()"
                                [disabled]="filterGroup.pristine">Pulisci 
                                  <mat-icon>clear</mat-icon></button>
                            // Bottone per pulire i filtri, disponibile sse e' stato
                                selezionato almeno un campo del form
                            <button mat-button class="form-btn" class="float-right"
                                (click)="cerca($event)"
                                [disabled]="filterGroup.pristine">Cerca 
                                  <mat-icon>search</mat-icon></button>
                        </div>
                    </div>
                </div>
            </form>
        </mat-expansion-panel>
    </mat-accordion>
</div>
<div class="row filter-table active-filters" *ngIf="activeFilters.length > 0">
  // Area in cui vengono indicati i filtri selezionati
    <mat-form-field>
        <mat-label>Filtri Attivi</mat-label>
        <mat-chip-list aria-label="Alert Filter">
            <mat-chip *ngFor="let filter of activeFilters">
                {{filter}}
            </mat-chip>
        </mat-chip-list>
    </mat-form-field>
</div>
```

I risultati ritornati dal MS che ritorna la lista di iter sono stati inseriti in una mat-table, la quale presenta una serie di colonne, ciascuna di esse valorizzata con il corrispettivo campo.

Il codice scritto in HTML sfruttando i componenti di angular material per la tabella con i risultati ottenuti dalla ricerca è il seguente:

```
<mat-card class="row-table">
     // Barra di progressione che compare fin tanto che il MS non ritorna un risultato -
         qualsiasi esso sia
      <div class="row" *ngIf="loading">
          <mat-progress-bar mode="indeterminate"></mat-progress-bar>
      </div>
      <mat-table *ngIf="dataSource.data" [dataSource]="dataSource">
       // Colonna relativa all'ID
          <!-- ID Column -->
          <ng-container matColumnDef="ID" class="col-1">
              <mat-header-cell *matHeaderCellDef>ID</mat-header-cell>
              <mat-cell *matCellDef="let iter; let i = index;">{{!iter.id ? "-" :
                  iter.id}}</mat-cell>
          </ng-container>
       // Colonna relativa al nome del cliente/produttore dei beni in oggetto
          <!-- Client Column -->
          <ng-container matColumnDef="Cliente" class="col-1">
              <mat-header-cell *matHeaderCellDef>Cliente</mat-header-cell>
              <mat-cell *matCellDef="">{{stanza.cliente?.descrizione}}</mat-cell>
          </ng-container>
       // Colonna relativa alla tipologia di iter (deposito, pignoramento/svincolo,
           scarico)
          <!-- Iter type Column -->
          <ng-container matColumnDef="Tipo iter" class="col-1">
              <mat-header-cell *matHeaderCellDef>Tipo iter</mat-header-cell>
              <mat-cell *matCellDef="let iter; let i =
```

```
            index;">{{!iter.tipoIter.descrizione ? "-" :
            iter.tipoIter.descrizione}}</mat-cell>
    </ng-container>
// Colonna relativa al percorso dell'iter
    <!-- Iter Path Column -->
    <ng-container matColumnDef="Percorso iter" class="col-1">
        <mat-header-cell *matHeaderCellDef>Percorso iter</mat-header-cell>
        <mat-cell *matCellDef="let iter; let i =
            index;">{{!iter.percorso.descrizione ? "-" :
            iter.percorso.descrizione}}</mat-cell>
    </ng-container>
// Colonna relativa allo stato dell'iter (terminato, in corso, declinato, non
     autorizzato)
    <!-- Iter Status Column -->
    <ng-container matColumnDef="Stato iter" class="col-2">
        <mat-header-cell *matHeaderCellDef>Stato iter</mat-header-cell>
        <mat-cell *matCellDef="let iter; let i =
            index;">{{!iter.statoProcesso.descrizione ? "-" :
            iter.statoProcesso.descrizione}}</mat-cell>
    </ng-container>
// Nel caso di un iter in corso - viene indicato il task in cui l'iter e' fermo
    <!-- Current Task Column -->
    <ng-container matColumnDef="Task corrente" class="col-2">
        <mat-header-cell *matHeaderCellDef>Task corrente</mat-header-cell>
        <mat-cell *matCellDef="let iter; let i =
            index;">{{!iter.taskCorrente.descrizione ? "-" :
            iter.taskCorrente.descrizione}}</mat-cell>
    </ng-container>
// Colonna relativa alla data in cui e' stato aperto l'iter
    <!-- Start date Column -->
    <ng-container matColumnDef="Data inizio" class="col-1">
        <mat-header-cell *matHeaderCellDef>Data inizio</mat-header-cell>
        <mat-cell *matCellDef="let iter; let i = index;"><span>{{!iter.dataInizio
            ? '-' : iter.dataInizio |
                date: 'dd/MM/yyyy' }}</span></mat-cell>
    </ng-container>
// Colonna relativa alla data di fine dell'iter o dell'ultima modifica fatta (nel
     caso di iter in corso)
    <!-- End date Column -->
    <ng-container matColumnDef="Data fine" class="col-1">
        <mat-header-cell *matHeaderCellDef>Data fine</mat-header-cell>
        <mat-cell *matCellDef="let iter; let i = index;">{{!iter.dataFine ? '-' :
            iter.dataFine | date:
            'dd/MM/yyyy' }}</mat-cell>
    </ng-container>
    <mat-header-row *matHeaderRowDef="displayedColumns" class="row
        table-row"></mat-header-row>
    <mat-row *matRowDef="let row; columns: displayedColumns;"></mat-row>
//Bottone che rimanda alla pagina di dettaglio dell'iter
    <!-- Detail Column -->
    <ng-container matColumnDef="Dettaglio" class="col-1">
        <mat-header-cell *matHeaderCellDef>Dettaglio</mat-header-cell>
        <mat-cell *matCellDef="let iter; let i = index;">
            <button *ngIf="iter.statoProcesso?.id != 1; else inCorso"
                mat-icon-button
                matTooltip="Dettaglio Iter" aria-label="dettaglio attore"
                    (click)="dettaglioIter(iter.id)">
                <mat-icon class="vertical-top"> info </mat-icon>
            </button>
            <ng-template #inCorso>
                <button mat-icon-button matTooltip="Riprendi Iter"
```

```
                        aria-label="dettaglio attore"
                        (click)="redirectToIter(iter.id)">
                        <mat-icon class="vertical-top"> input </mat-icon>
                    </button>
                </ng-template>
            </mat-cell>
        </ng-container>
    </mat-table>
    //Paginatore, utile nel caso in cui ci siano tanti risultati da mostrare in
        tabella (al piu' 5 iter per vista/pagina)
    <mat-paginator [pageSizeOptions]="[5, 10, 20]" [length]="dataSource.data.length"
        [pageSize]="5"
        showFirstLastButtons aria-label="Visualizza gli iter">
    </mat-paginator>
</mat-card>
```

In Figura 24 è presente l'interfaccia grafica del componente realizzato.

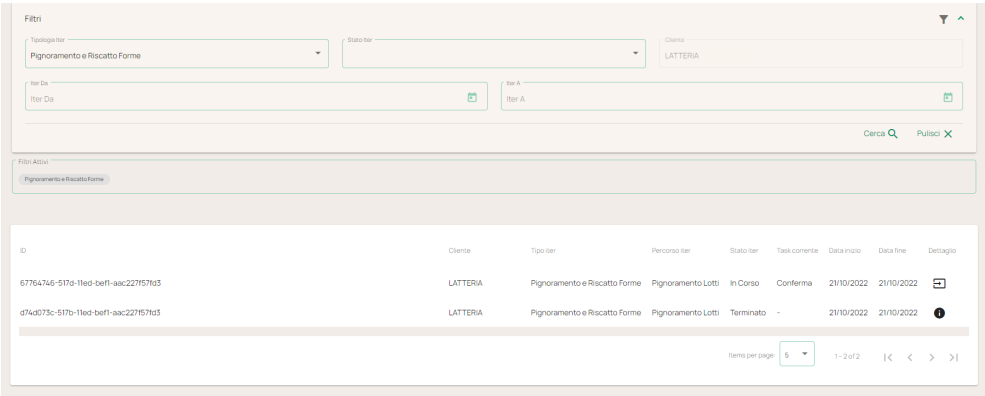


Figura 24: Il componente relativo allo Storico Iter in Inventory Chain Platform

5.3.2 Moduli

Il componente relativo allo storico iter che, all'interno del file TS, ha le sue per chiamare i microservizi ed eseguire operazioni che si riflettono, principalmente, sulla vista.

Questo componente è a sé stante ed è chiamato dal componente padre, mediante relativo modulo (file `module.ts`), che gestisce tutti i componenti relativi all'iter (ad esempio: oltre allo storico iter, questo modulo gestisce anche il componente per inizializzare un deposito).

5.3.3 Servizi

Per la parte logica è stato necessario capire come e da dove prendere le informazioni da mostrare nelle select, oltre alla denominazione del produttore. Inoltre, è stato fatto uno studio su come passare i filtri al microservizio[G]che ritorna i risultati relativi allo storico iter e, di conseguenza, come implementare le chiamate REST[G].

Per la lista di tipologie e gli stati associati all'iter è stato sufficiente chiamare due metodi GET distinti, che mi hanno permesso di popolare le select. Contrariamente, per ottenere la denominazione del produttore e valorizzare il campo di input del cliente è stato sufficiente prelevare l'informazione (sfruttando la libreria ngx-context, che passa i dati di contesto di una determinata pagina) dalle pagine precedenti allo storico iter. Infatti, non avrebbe avuto senso fare una chiamata ad un microservizio[G]per ottenere un dato già disponibile.

Infine, è stato studiato come realizzare il mapper per chiamare il microservizio[G](un metodo POST) che ritorna i risultati relativi allo storico, in base a dei filtri dati.

Il microservizio[G]- come indicato sullo swagger (§3.7 - Figura 15 e Figura 16) - chiedeva di ritornare una request con almeno un dato valorizzato ed i dati erano i seguenti:

- Tipologia iter di tipo intero,
- Stato iter di tipo intero,
- Data di inizio iter di tipo stringa,
- Data di fine iter di tipo stringa.

Per la tipologia e lo stato iter, se selezionati, è stato preso l'id (di tipo intero) del valore scelto, mentre per le date è stata fatta una formattazione del valore passato in input dall'utente: è stato preso l'anno, il mese ed il giorno (ciascuno separato dal carattere speciale "/"), il tutto messo sotto forma di stringa come richiesto dal microservizio[G]. Il risultato ritornato dal microservizio[G], qualora siano presenti risultati, corrisponde alla lista degli iter mostrata nella visualizzazione storico iter (§2.1.8 UC 2).

Routing ed endpoint Angular Routing è una libreria di Angular che consente di spostarsi tra le viste di un'applicazione (in altri termini, tra le pagine web), senza la necessità di dover ottenere una nuova pagina dal server.

Nel componente realizzato è stata inserita una rotta nel dettaglio iter (dalla tabella dei risultati) che, al click del bottone, rimanda alla pagina di dettaglio dell'iter in questione.

Il codice seguente mostra come viene fatto il route verso il modulo relativo alla pagina di dettaglio iter:

```
path: "dettaglio-iter",
      //DettaglioIterComponent e' un altro componente - che mostra il dettaglio di un iter
      component: DettaglioIterComponent,
        resolve: {},
        canActivate: [AuthGuard],
        data: {
          breadcrump: 'Dettaglio iter'
        }
```

L'oggetto Router viene utilizzato anche all'interno del file TS del componente, che viene implementato all'interno di una funzione attivata al click dell'utente sul bottone relativo al dettaglio.

Il codice nel file TS che rimanda alla pagina di dettaglio è il seguente:

```
dettaglioIter(iter: any) {
   //Mediante la chiamata getProcessoByIdProcessFlowable si ottiene l'id dell'iter
    this.processiController.getProcessoByIdProcessFlowable(iter).subscribe({
    // se il servizio risponde con un codice 200 allora viene eseguito questo codice
      next: (res) => {
         //se l'id esiste - al click l'utente verra' indirizzato verso la pagina di
             dettaglio dello specifico iter selezionato
        this.appContextService.insert(AppContextVariables.dettaglioIter,res);
        this.router.navigate(["gestione-iter/dettaglio-iter"]);
      },
      // altrimenti viene ritornato un messaggio d'errore ed il routing alla pagina di
          dettaglio non potra' essere fatto
```

```
      error: (error) => {
        this.errorService.getHttpErrorMsg(error,ServiceName.iterById);
      }
    });
}
```

Un endpoint di comunicazione è un tipo di nodo per la comunicazione in rete e la sua interfaccia viene esposta tramite una parte comunicativa.

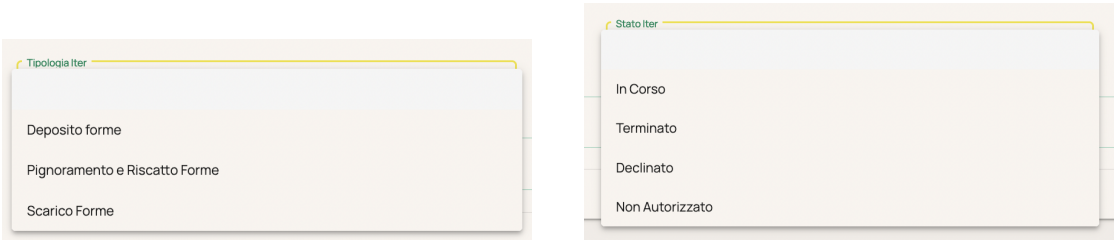
Per realizzare il componente richiesto è stato necessario utilizzare i seguenti endpoint:

- `GET / iter/tipologie/processo`, che ritorna la lista di tipologie di processo (che possono essere: Deposito Forme, Pignoramento e Riscatto Forme, Scarico Forme);
- `GET / iter/stato/processo`, ritorna la lista degli stati in cui si può trovare un task (ossia: In Corso, Terminato, Declinato o Non Autorizzato),
- `POST / iter/list/processi`, è la chiamata vera e propria al microservizio[G]che ritorna la lista di iter, una volta valorizzato almeno un campo del form.

Il codice TS che viene eseguito quando viene aperta la pagina (mediante il metodo ngOnInit di Angular) e che si occupa di ottenere i risultati dai MS realizzati per ottenere le tipologie e lo stato di un processo è il seguente:

```
ngOnInit(): void {
   //Mediante il resolver viene aperta una subscribe - che rimane in attesa fin tanto che
       non ottiene un risultato dal MS chiamato
    this.resolver = this.route.data.subscribe({
      next: (storico) => {
      //La seguente istruzione ritorna la tipologia dei processi
        this.listaTipologieProcessi =
            this.errorService.checkResolverErrors(storico.listaTipologieIter,
            AppContextVariables.lTPro, ResolverName.elTPro, false);
        //Nel caso in cui la lista sia vuota - il filtro viene disabilitato
        if (this.listaTipologieProcessi.length <= 0) {
          this.filterGroup.controls['tipologia'].disable();
        }
        //La seguente istruzione ritorna gli stati su cui si possono trovare i processi
        this.listaStatiProcessi =
            this.errorService.checkResolverErrors(storico.listaStatiIter,
            AppContextVariables.lStat, ResolverName.elStat, false);
      }
    });
  }
```

In Figura 25 (a) è presente la lista dei risultati ottenuti chiamando il MS `GET / iter/tipologie/processo`, mentre in Figura 25 (b) è presente la lista dei risultati ottenuti chiamando il MS `GET / iter/stato/processo`.



(a) (b)

Figura 25: La valorizzazione dei componenti mat-select per tipologia e stato iter

6 Background

In questo capitolo verranno trattati alcuni approfondimenti in merito al pegno rotativo non possessorio e la metodologia agile, due aspetti che riguardano da vicino la piattaforma ICP.

6.1 Che cos'è il pegno rotativo non possessorio?

Nel diritto civile, un pegno non è altro che un diritto reale di garanzia su beni altrui; il titolare del bene concesso in garanzia viene spossessato del medesimo ed il creditore (p.es. la banca) potrà soddisfarsi sullo stesso nel caso di inadempimento al proprio credito.

Con il "**pegno rotativo**" si indica il fatto che le merci impegnate a garanzia possono essere sostituite con altre merci di pari valore, a patto che di tale operazione ne venga a conoscenza anche il creditore.

Nel settore dei prodotti lattiero caseari a lunga stagionatura, così come nel settore vitivinicolo e, più in generale, per i prodotti agroalimentari DOP e IGP, esistono dei registri telematici appositi che annotano tutte le movimentazioni dei beni (ad esempio, il registro telematico che si occupa di prodotti vitivinicoli è il SIAN, acronimo di "Sistema Informativo Agricolo Nazionale").

Questi registri vengono vidimati annualmente da parte di un notaio e contengono tutte le movimentazioni (come l'entrata, l'uscita dei beni e relativi dettaglio come: peso, numero forme, casello di produzione etc.), nonché tutte le fasi di esistenza della garanzia: dalla costituzione all'eventuale sostituzione del bene, fino all'estinzione del bene stesso.

Inoltre, al creditore viene concesso il diritto di ispezionare i prodotti concessi in garanzia attraverso dei campioni o controllando da remoto i beni, attraverso l'utilizzo di nuove tecnologie.

Grazie alla loro efficienza, questi registri telematici permettono al pegno di essere definito pegno "non possessorio"; si tratta di una forma di garanzia introdotta con il decreto-legge n. 59 del 3 maggio 2016 (conv. in l.n. 119/2016) tramite la quale, nel caso dei prodotti lattiero caseari, il pegno resta nella disponibilità del debitore e del suo processo produttivo (ad esempio, all'interno di un magazzino), mentre la garanzia "prestata" viene, di volta in volta, rinnovata, ma rimanendo invariata rispetto ciò che è stato accordato in fase di costituzione del pegno stesso.

In questo modo, chi concede il finanziamento non dovrà preoccuparsi degli oneri di custodia del bene, dei rischi relativi a crisi finanziare e di tutti i problemi che ne conseguono.

6.2 Inventory Chain Platform

Inventory Chain Platform è che una piattaforma web "chiavi in mano[G]" rilasciata nella versione 1.0 all'inizio del 2022, nonché un software SaaS (acronimo di "Software as a Service") che ha come scopo quello di permettere a degli stabilimenti produttivi di ottenere un finanziamento da parte di un istituto finanziario tramite il pegno rotativo non possessorio.

Dal punto di vista del debitore, tramite ICP è possibile mettere a pegno dei beni alimentari e, conseguentemente, ottenere liquidità per fare investimenti a supporto della filiera produttiva, abbattere dei costi logistici per la costituzione del pegno, certificare un elevato controllo di qualità sul ciclo di lavorazione, tracciare il ciclo di produzione ed avere costi minori per la gestione della rotatività della garanzia.

Analogamente, per gli istituti finanziari ICP è una soluzione di finanziamento su misura che sfrutta la blockchain per la vidimazione dei registri, che permette di sostenere le imprese nel loro processo di sviluppo, di azzerare i costi di gestione per l'immagazzinamento diretto dei beni e di monitorare

costantemente la garanzia, oltre a ricevere alert in tempo reale qualora non venissero rispettati i sistemi di qualità standard da parte del debitore.

In Figura 26 è presente una parte della pagina iniziale della piattaforma.

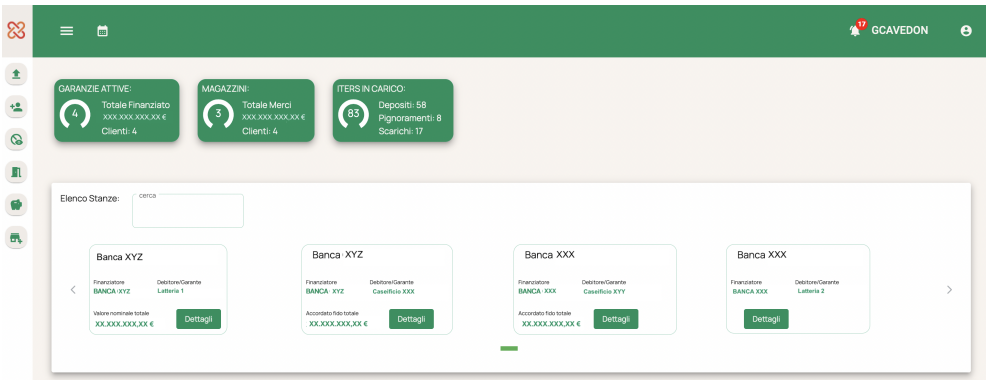


Figura 26: Homepage di ICP, sezione dedicata alle stanze

6.3 Il mindset Agile ed il framework Scrum

Il termine "Agile" fu utilizzato per la prima volta nel 2001, nel "Manifesto per lo Sviluppo Agile del Software" pubblicato da Kent Beck, Robert Cecil Martin e Martin Fowler, e fa riferimento ad un mindset; ossia, una filosofia, un modo di pensare ed un paradigma organizzativo, che nell'ambito dello sviluppo del software si traduce in una distribuzione continua di software efficienti creati in modo rapido ed iterativo.

A differenza di altre metodologie (per esempio: la metodologia Waterfall, nella quale budget, tempi ed obiettivi vengono definiti chiaramente ed in maniera strutturata), la metodologia Agile non prevede regole da rispettare nell'ambito dello sviluppo del software.

Inoltre, questa metodologia si prefigge come obiettivo quello di rilasciare in tempi brevi piccole modifiche al software, al fine di aumentare il grado di soddisfazione del cliente, oltre al fatto di concentrarsi sul miglioramento continuo del prodotto stesso.

I team che si occupano di sviluppo ed adottano una metodologia agile sono composti per lo più da poche persone, si organizzano in autonomia e collaborano direttamente con i rappresentanti aziendali mediante incontri periodici svolti durante tutto l'intero ciclo di vita dello sviluppo del software.

Esistono diversi framework agili per la gestione del ciclo di sviluppo del software e, tra questi, c'è anche lo Scrum, che è il più adottato ed è stato adottato anche in ICP.

6.3.1 Scrum

Scrum è un framework iterativo ed incrementale per la gestione del lavoro ed è stato progettato per team di piccole dimensioni (in genere da 5 a 9 persone), i quali suddividono il carico di lavoro in attività completabili in un arco temporale breve comunemente detto "Sprint".

Al fine di avere una rappresentazione visiva dello stato di avanzamento del ciclo di sviluppo, il team ha scelto di utilizzare la Board Scrum di Jira, che promuove la pianificazione degli sprint, la suddivisione del carico di lavoro e lo sviluppo iterativo.

In figura 27 vengono illustrati i cinque principi dello SCRUM.

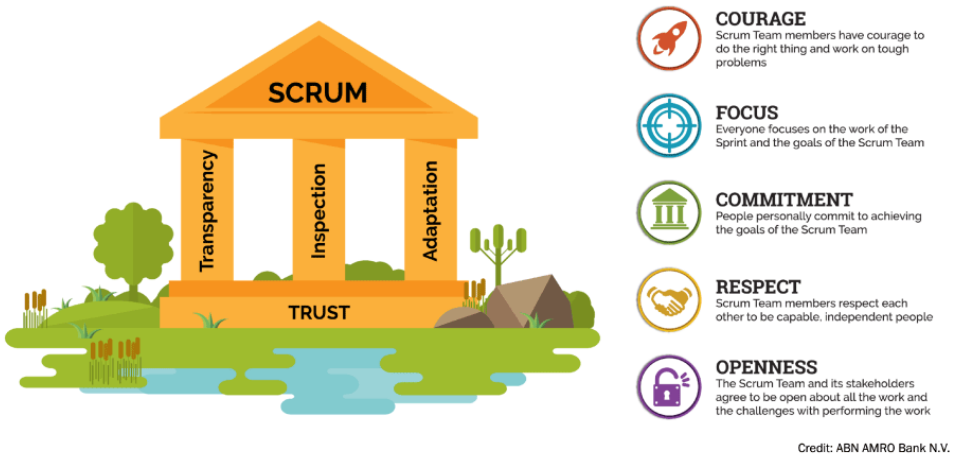


Figura 27: I cinque principi del framework Scrum - Fonte: ABN AMRO bank

I principi illustrati in figura 27 li possiamo sintetizzare come segue:

- **Focalizzazione**, in quanto con lo Scrum è importante concentrarsi su poche cose alla volta, al fine di produrre risultati di qualità e rilasciare più rapidamente;
- **Coraggio**, di affrontare sfide sempre più ambiziose;
- **Apertura mentale**. Lavorando con altre persone è possibile avere un confronto in merito ai progressi ed eventuali problematiche affrontate, attività che permette di risolvere preoccupazioni e di migliorare sempre di più;
- **Impegno**, per avere maggior controllo sul proprio destino ed il proprio successo;
- **Rispetto**, lavorando assieme, condividendo i successi ed i fallimenti, si impara a rispettare il prossimo ed ad aiutare reciprocamente a guadagnare quel rispetto.

I valori del manifesto agile che possono essere applicati allo Scrum sono:

1. **Individui ed interazioni hanno priorità rispetto ai processi ed agli strumenti**. Infatti, alla base dello scrum c'è la fiducia nei confronti di tutti i componenti del team e del modo con cui essi interagiscono; i team che adottano lo scrum, hanno la responsabilità di scoprire cos'è necessario fare e come farlo, oltre al fatto di mettersi all'opera per implementare le soluzioni scelte. Ciascun team ha la responsabilità di identificare eventuali problemi di rallentamento e di trovare soluzioni per risolvere i punti critici;
2. **Software funzionante più che una documentazione esaustiva**. Lo scrum richiede che ci sia un incremento di prodotto finito e rilasciabile a fine di ogni Sprint, poiché è molto più importante avere un prodotto finito, testato e funzionante, rispetto ad avere una buona documentazione esaustiva (che dovrà esserci, ma verrà messa in secondo piano rispetto al software);
3. **La collaborazione col cliente più che la negoziazione dei contratti**. È essenziale che il responsabile dello sviluppo e dell'ottimizzazione di un prodotto lavori con tutti i membri del team, al fine di determinare cosa dovrà essere prodotto realmente. Questa figura ha, quindi, il ruolo di scegliere le cose più di alto valore da fare per il prodotto e deve assicurarsi che il prodotto realizzato abbia il maggior valore possibile in ogni momento del suo sviluppo;
4. **Rispondere al cambiamento più che seguire un piano**. L'ultimo aspetto, ma non di minor importanza, è quello legato al cambiamento: Scrum funziona bene in tutti i team che "ispezionano" apertamente quanto succede in un determinato momento e "adottano" le loro azioni alla realtà.

7 Conclusioni

In quest'ultimo capitolo verranno tratte le somme in merito al tirocinio svolto, al fine di dare una valutazione finale di quanto fatto ed appreso nei due mesi di lavoro in azienda.

7.1 Raggiungimento degli obiettivi

Il piano di lavoro che era stato stilato prima di iniziare il tirocinio formativo è stato rispettato e gli obiettivi prestabiliti sono stati raggiunti.

7.2 Competenze acquisite

Questa esperienza mi ha permesso di crescere sotto il punto di vista professionale ed accrescere le mie competenze nell'ambito dello sviluppo front-end[G], ma non solo.

Senz'altro, ciò che porterò nel mio bagaglio culturare al termine di questo tirocinio sono le competenze acquisite, sia in termini formativi che pratici, delle tecnologie studiate, in particolare Angular 12 e TypeScript, oltre all'applicazione delle stesse per la realizzazione del componente sviluppato.

Inoltre, questo tirocinio mi ha permesso di confrontarmi e collaborare con figure con più esperienza di me nel settore, al fine di portare a termine i compiti richiesti.

Nonostante le difficoltà nell'apprendere nuove tecnologie, gli aspetti più complessi che ho dovuto affrontare sono stati:

- **Comprensione del problema e del contesto**: il progetto in sé non è stato di immediata comprensione, sia dal punto di vista della grandezza che degli argomenti trattati. Non avendo un "background[G]" molto ampio del settore finanziario, certi concetti e soluzioni architetturali attuate dal team sono risultati difficili da comprendere ed hanno richiesto più tempo del previsto in termini di analisi;
- **Rapporto con i colleghi**: a differenza di ciò che accade nell'ambiente accademico, dove tutti noi studenti abbiamo un bagaglio culturale ed un'esperienza più o meno simile, relazionarsi con risorse esperte nel settore alle volte è stato complesso. Infatti, sia nei meeting con tutti i componenti del team che con il tutor, ho sempre cercato di stare al passo con le tematiche affrontate, nonostante mi mancassero delle informazioni in merito al contesto (ad esempio: alcuni concetti affrontati prima del mio arrivo). Tuttavia, qualora le tematiche affrontate non mi fossero chiare, ho sempre chiesto spiegazioni senza alcun problema.

Tra le altre competenze acquisite ci sono anche:

- **"Work for goals"** (o "lavorare per obiettivi"), attività che consiste nel porsi degli obiettivi e raggiungerli nel breve-lungo periodo, di modo da avere sempre in chiaro ciò che si sta facendo ed il contributo che si sta dando al progetto;
- **Prendere in carico un'attività per volta**, di modo da non avere mai vecchie attività in sospeso non completate o con bug non risolti.

Jira è stato un ottimo strumento di supporto per definire obiettivi chiari e comprendere meglio le ultime due competenze descritte.

Analogamente, i "daily", ossia i meeting svolti quotidianamente (dalle 9:30 alle 10:00 circa) con tutti i membri del team, sono stati utili per capire l'andamento del tirocinio, i risultati prodotti e sapere cosa stessero facendo/a cosa stessero lavorando i miei colleghi, analizzare eventuali punti di blocco, definire nuove attività ed aggiornare il Jira.

8 Acronimi

API, Application Programming Interface,
BE, Back-end,
CLI, Angular, Command-Line Interface (Interfaccia a Riga di Comando),
FE, Front-end,
DI, Dependency Injection,
ICP, Inventory Chain Platform,
IDE, Integrated Development Environment (Ambiente di Sviluppo Integrato),
GWT, Google Feedback Tool
MGT, Magazzino delle Tagliate,
MS, Micro-servizio,
MVC, Model View Controller,
MVVM, Model View View Model,
REPO, Repository,
REST, Representational State Transfer,
SaaS, Software as a Service,
SEO, Search Engine Optimization (Ottimizzazione per Motori di Ricerca),
SIAN, Sistema Informativo Agricolo Nazionale,
TS, TypeScript,
UAT, User Acceptance Testing (Test di Accettazione dell'Utente - Collaudo),
UI, User Interface (Interfaccia utente),
VCS, Version Control System (Sistema di Controllo Versione)
WMS, Warehouse Management System

9 Glossario

9.1 A

9.1.1 API

Una API è un'interfaccia di programmazione di una applicazione, ossia un insieme di procedure volte a risolvere uno specifico problema tra computer o software diversi e consentire lo scambio di informazioni.

Ad esempio, in ICP sono state usate le API per fare "*two-way data binding*", ossia estrapolare dati dal database a seguito di una richiesta (come la richiesta di estrapolare solamente i processi relativi allo storico iter che avessero come stato "In Corso").

9.2 B

9.2.1 Back-end

Il termine "Back-end" ("Backend") denota la parte che permette il corretto funzionamento di un programma ed agisce a seconda delle operazioni svolte dall'utente lato frontend[G].

Tra le tante cose, la parte backend di un software definisce il processo di estrazione ed elaborazione dei dati dalle fonti originarie (come potrebbe essere un database), oppure, nel caso di siti web e web app, questa parte permette di amministrare il sito stesso (modificando contenuti e creando pagine), aspetto accessibile solo agli amministratori del sito web.

9.2.2 Background

Il termine "background" inserito in §6.3 si riferisce al fatto che non avessi conoscenze pregresse in merito ad un argomento, in questo caso il settore finanziario.

9.2.3 Bad Request

Un errore "Bad Request" (o errore 400/errore HTTP 400) è un errore generico che viene ritornato a chi fa una richiesta ad un servizio ed esso indica che il server non può o non vuole elaborare la richiesta a causa di qualcosa che è percepito come un errore del client (ad esempio: la sintassi della richiesta malformata - potrebbero esserci delle incongruenze con i tipi dei dati passati -, framing del messaggio di richiesta non valido o routing della richiesta ingannevole).

9.2.4 Branch

Un branch (in italiano "ramo") in Git è proprio come un ramo di un albero; è una parte che compone il tutto (l'albero) di un progetto.

Ciascun branch è un ambiente isolato nel quale aggiungere, modificare o eliminare una parte di codice senza interferire con il codice principale del progetto (in genere, il ramo master dell'ambiente). L'attività che consente di creare una seconda linea di sviluppo nel progetto, senza interferire con il ramo principale, è detta branching.

9.2.5 Browser Web

Un browser web (o semplicemente browser) non è altro che un'applicazione installata su un dispositivo che permette l'acquisizione, la presentazione e la navigazione di risorse sul web. Tali risorse (come pagine web, immagini, testi e video) vengono messe a disposizione sul World Wide Web su una rete locale o sullo stesso dispositivo su cui il browser è in esecuzione.

9.3 C

9.3.1 Chiavi in mano

La terminologia "chiavi in mano" (o "turn-key") nell'ambito dello sviluppo software la si utilizza per indicare un prodotto realizzato internamente in azienda e già pronto per essere venduto o personalizzato secondo le esigenze del cliente. A differenza dei software su misura (o "tailored solution"), dove viene realizzato un progetto secondo delle linee guida specifiche date dall'utente o da una determinata azienda, una soluzione chiavi in mano è più economica, consente di avere maggior libertà (ad esempio, per la scelta dei linguaggi e delle tecnologie da utilizzare, oltre all'interfaccia grafica) e permette di mantenere la paternità del software in questione (ad esempio: posso vendere una licenza per usarlo).

9.3.2 Cloud Computing

Con il termine "cloud computing" si intende la distribuzione di risorse digitali tramite la rete internet a seconda della richiesta dell'utente. Anziché possedere, acquistare e mantenere delle strutture fisiche nelle quali conservare dati ed applicazioni, mediante il cloud computing è possibile accedere a servizi tecnologici, capacità di calcolo, database ed archiviazione a seconda delle necessità dell'utente. Tra i tanti vantaggi del cloud computing ci sono senz'altro:

- Performance, scegliento i migliori fornitori di servizi cloud, è possibile usufruire delle migliori infrastrutture cloud ed accedere ai dati contenuti al loro interno in maniera fluida e veloce;
- Costo, che può essere dilazionato, fisso o variabile a seconda delle risorse a cui l'utente accede;
- Accessibilità, perché è possibile accedere a delle risorse condivise da qualsiasi dispositivo in qualsiasi parte del mondo.

I principali fornitori che offrono il cloud computer sono Amazon con Amazon Web Services (AWS) e Microsoft con Microsoft Azure, Google con Google Cloud Platform (GCP), IBM con IBM Cloud (Kyndryl), Oracle con Oracle Cloud e Alibaba con Alibaba Cloud.

9.3.3 Continuous integration, Integrazione Continua

Si tratta di un metodo di sviluppo software con cui gli sviluppatori aggiungono regolarmente modifiche al codice in un repository[G]centralizzato; quindi, la creazione di build e test vengono eseguiti automaticamente. Gli obiettivi principali dell'integrazione continua sono individuare e risolvere i bug con maggior tempestività, migliorare la qualità del software e ridurre il tempo richiesto pe convalidare e pubblicare nuovi aggiornamenti.

9.3.4 Coni di visibilità

Coni di visibilità è un termine generico per indicare i ruoli e permessi che vengono dati ad un utente registrato all'interno di una piattaforma.

In altri termini, rappresenta le funzionalità che vengono date ad un utente e le operazioni che può compiere all'interno del proprio spazio di lavoro.

Per ICP, un utente che non ha associato il cono di visibilità relativo agli iter, non può visualizzare lo storico iter (l'icona che rimanda allo Storico Iter viene nascosta a coloro che non dispongono di questa funzionalità).

Un altro esempio potrebbero essere i siti web con abbonamenti a pagamento (servizi di streaming musicale, streaming video, e-commerce); anche qui, se un utente non ha effettuato l'upgrade ad un piano a pagamento, gli verrà associato il cono di visibilità che, al suo interno, avrà anche il ruolo relativo alla ricezione della pubblicità (vedi Spotify e Spotify Premium) o la spedizione a pagamento (vedi Amazon ed Amazon Prime).

9.4 D

9.4.1 DevOps

La terminologia "DevOps" è data dalla combinazione di due parole:

- *Dev*, abbreviazione di "Development" - in italiano "sviluppo",
- *Ops*, abbreviazione di "Operations" - in italiano "operazioni".

e descrive gli approcci da adottare per accelerare i processi che consentono ad un'idea (ad esempio: una nuova funzionalità software, una richiesta di miglioramento o un bug fix) di passare dallo sviluppo al deployment in ambiente di produzione, dove l'implementazione diventa subito fruibile all'utente.

9.4.2 Dependency Injection

Dependency injection (DI) è un design pattern della programmazione orientata agli oggetti il cui scopo è quello di semplificare lo sviluppo e migliorare la testabilità di software di grandi dimensioni.

Per utilizzare tale design pattern è sufficiente dichiarare le dipendenze di cui un componente necessita (dette anche interface contract). Quando il componente verrà istanziato, un iniettore si prenderà carico di risolvere le dipendenze (attuando dunque l'inversione del controllo). Se è la prima volta che si tenta di risolvere una dipendenza l'injector istanzierà il componente dipendente, lo salverà in un contenitore di istanze e lo restituirà. Se non è la prima volta, allora restituirà la copia salvata nel contenitore. Una volta risolte tutte le dipendenze, il controllo può tornare al componente applicativo.

Il pattern Dependency Injection coinvolge almeno tre elementi:

- una componente dipendente,
- la dichiarazione delle dipendenze del componente, definite come interface contract,
- un injector (chiamato anche provider o container) che crea, a richiesta, le istanze delle classi che implementano delle dependency interface.

9.5 I

9.5.1 IDE

Un ambiente di sviluppo integrato (o "IDE", acronimo di "Integrated Development Environment") è un software (per la maggior parte dei casi, da installare nella propria macchina, ma ne esistono anche di gratuiti online) che, in fase di coding, supporta i programmatori nello sviluppo e nel debugging del codice sorgente di un programma.

Un buon IDE aiuta anche ad identificare eventuali errori di sintassi del codice in fase di scrittura, dispone di un compilatore e/o un interprete, presenta un tool di building automatico ed offrire varie funzionalità di debugging. In genere, con un IDE è possibile sviluppare codice in diversi linguaggi di programmazione.

9.6 F

9.6.1 Fork

Nell'ambito del versionamento[G] e, in particolar modo, di Git, una fork[G] è una copia di un repository[G] o di un branch[G], che permette di sperimentare qualsiasi cambiamento, senza dover modificare il progetto originario.

A differenza del "clone", una fork[G] mantiene una connessione con il repo originario e, grazie a questo aspetto, è possibile contribuire con le proprie modifiche al branch o al repository[G] originale tramite una operazione di push.

9.6.2 Front-end

La terminologia "Front-end" (o "frontend") consiste nell'altra faccia della medaglia rispetto alla parte "backend[G]" e rappresenta ciò che un utente vede quando usa un programma o un sito web/web App. In altre parole, con il termine Frontend ci si riferisce all'interfaccia grafica, la quale permette di interagire con l'applicativo in questione con una grafica migliore e più intuitiva che una semplice riga di comando.

Mediante l'interazione dell'utente con la parte frontend di un programma o un'applicazione web vengono inviati dei dati lato backend, che può elaborare quanto ricevuto o, semplicemente, salvare tali dati in un database o cloud, e - a sua volta - ritornare dei dati da mostrare a schermo.

9.7 G

9.7.1 Git

Git è un software per il controllo di versione distribuito utilizzabile da interfaccia a riga di comando creato da Linus Trovalds, autore del kernel Linux, nel 2005.

Git (che nello slang britannico significa "idiota") nacque per essere un semplice strumento per facilitare lo sviluppo del kernel Linux ed è diventato uno degli strumenti di controllo versione più diffusi. Grazie alle sue eccellenti prestazioni e funzionalità, Git risulta essere il VCS gratuito ed open source più diffuso ed utilizzato.

9.7.2 Google Feedback Tool

Si tratta di uno strumento realizzato da Google per permettere agli utenti di inviare una segnalazione di un bug o dare un suggerimento per contribuire a migliorare i prodotti realizzati da Google.

9.8 H

9.8.1 Hooks

Gli hooks in Angular rappresentano un insieme di eventi che definiscono ogni singolo passsaggio che effettua un componente sviluppato con questo framework, dalla sua creazione alla sua distruzione.

Tali eventi possono essere sfruttati per effettuare operazioni in un preciso stato nel quale si trova un componente.

Alcuni degli hooks Angular più utilizzati sono:

- **Constructor**, è il metodo di base che, sebbene non faccia parte del ciclo di vita del componente, viene trattato come tale. Questo componente viene eseguito quando il componente viene istanziato;
- **OnChanges**, questo evento viene utilizzato in prima istanza (al primo evento del componente), ma anche quando i valori in input del componente vengono cambiati;
- **OnInit**, questo evento dichiara che Angular è pronto alla creazione del componente. Questa fase si verifica solo dopo il primo OnChanges e viene eseguito un'unica volta;
- **DoCheck**, questo evento viene utilizzato per valutare se sono state fatte delle modifiche ai componenti ed ai dati. A differenza dall'OnChanges, che si attiva ad ogni cambiamento di una proprietà di input a seguito di un'azione svolta da parte dell'utente, il DoCheck viene lanciato molto più spesso - ad esempio, anche allo spostamento del mouse -;
- **AfterContentInit**, questo evento si riferisce al completamento degli eventi legati al componente, nello specifico alla fine della creazione dell'albero dei componenti figli;
- **OnDestroy**, questa è l'ultimo evento che può essere innescato in un componente e si verifica prima che questo venga distrutto definitivamente (per questo motivo, viene eseguito un'unica volta). In genere, viene usato per ripulire il componente e, per esempio: interrompere i timer degli intervalli, lanciare l'unuscribe agli oggetti Observable ed a tutte le chiamate.

9.9 K

9.9.1 Kanban Board

Come fa intuire il nome stesso, una Kanban Board è una sorta di tavola/lavagna/cruscotto che permette di avere una visione generale (ma anche di dettaglio) del flusso di lavoro di un progetto.

Una Kanban Board offre numerose funzionalità e componenti, i quali permettono di realizzare e mantenere al meglio il workflow del progetto, di modo che le attività da svolgere ed il relativo stato di avanzamento siano sempre sotto controllo.

Ad esempio, la principale funzionalità della Kanban Board di Jira è quella di permettere all'utente ed al team di visualizzare i ticket aperti in un determinato sprint, oltre a mostrare il "Reporter" e l'assegnatario del ticket stesso, i tempi di consegna, la to-do list, gli allegati del ticket e molte altre informazioni utili. Inoltre, per ciascun ticket è possibile visualizzare e modificare lo stato di avanzamento.

9.10 M

9.10.1 Markup, Linguaggi di

Un linguaggio di markup (di marcatura o di formattazione) consiste in un insieme di istruzioni che descrivono i meccanismi di rappresentazione o impaginazione di un testo.

Un esempio popolare di linguaggio di Markup (di tipo descrittivo) è l'HTML, che consente di descrivere pagine web ed utilizza un insieme predefinito di tag per descrivere gli elementi presenti all'interno delle stesse (come: <head></head>, <body></body>). Un altro linguaggio di Markup (di tipo procedurale) è LaTeX, un programma di composizione tipografica che permette di realizzare documenti formali.

9.10.2 Merge

L'operazione di merge consiste nell'unire un'insieme diramazioni di branch[G](dette anche fork[G]) e relativi commit, per svolgere questa attività è necessario invocare il comando di git[G]merge. In ICP ho utilizzato molto spesso l'operazione di merge per unire le mie modifiche al branch[G]di sviluppo (develop), dove era presente una versione stabile della piattaforma.

9.10.3 Microservizio

I microservizi (o micro-servizi) sono un approccio per sviluppare e organizzare l'architettura dei software, i quali saranno composti di servizi indipendenti di piccole dimensioni e che comunicano tra loro tramite API ben definite.

Le architetture a microservizi permettono di scalare e sviluppare le applicazioni in modo più rapido e semplice, oltre a promuovere l'innovazione e accelerare il time-to-market (ossia, la fase temporale che intercorre tra l'ideazione di un prodotto e la sua effettiva commercializzazione) di nuove funzionalità.

9.10.4 Mobile Friendly

Un sito web o una Web App Mobile Friendly è un sito nel quale sono presenti dei contenuti che possono essere consultati anche su dispositivi con schermi di piccole dimensioni (dispositivi mobile come tablet e smartphone).

In genere, questa funzionalità la si ottiene grazie al design responsive; una tecnica di web design che permette di adattare automaticamente vari componenti presenti in un sito su schermi di varie dimensioni.

Ciò riduce al minimo la necessità, da parte dell'utente, di ridimensionare o scorrere i componenti presenti nella pagina, poiché si adattano in automatico alla dimensione dello schermo.

9.10.5 Mockup

Un mockup (o mock-up) non è altro che una realizzazione a scopo dimostrativo di un oggetto o sistema, senza alcuna funzionalità ausiliaria; un mockup può rappresentare la totalità o solo una parte dell'oggetto originale di riferimento.

Per quanto riguarda ICP, sono stati realizzati dei mockup in formato JSON contenenti un esempio di request e response per il microservizio in oggetto (il MS che ritorna la lista degli iter, dati dei filtri).

9.11 R

9.11.1 Repository

Un repository (letteralmente "deposito" o "ripostiglio") è un ambiente che contiene tutti i file e la documentazione relativa ad un determinato progetto. Al suo interno vengono archiviate le modifiche fatte al codice nel tempo. Il reposotory tiene traccia della cronologia del progetto e di chi ha apportato cambiamenti allo stesso (con data e descrizione se pervenuta), aspetto che permette di tornare ad uno stadio precedente di uno o più file nel caso uno sviluppatore abbia sbagliato qualcosa.

Esistono repository collaborativi, nei quali diversi membri del team contribuiscono in maniera diversa ed asincrona allo stesso progetto, e repository privati.

9.11.2 REST

REST, acronimo di "Representational State Transfer", è uno stile architetturale per sistemi distribuiti. L'espressione fu introdotta nella tesi di dottorato di Roy Fielding, uno dei principali autori delle specifiche dell'HyperText Transfer Protocol (HTTP), nel 2000.

L'utilità dell'architettura REST è relativa alla trasmissione di dati tramite HTTP, senza ulteriori livelli aggiuntivi e non prevedono il concetto di sessione.

Il funzionamento delle chiamate REST prevede una struttura URL ben definita, la quale identifica univocamente una o un insieme di risorse e l'utilizzo dei metodi HTTP specifici la gestione delle informazioni (ad esempio: mediante un metodo GET è possibile ottenere delle informazioni, mediante un metodo POST, PUT, PATCH e DELETE per la modifica).

9.12 S

9.12.1 Sistema di versionamento del codice, Sistemi di controllo versione

Un sistema di versionamento[G]del codice è un ambiente nel quale è possibiile visualizzare lo storico del codice e le modifiche apportate.

All'interno di un sistema di versionamento[G]del codice (come Git) è possibile gestire le varie versioni di uno stesso file o documento, poiché tiene traccia delle modifiche effettuate nel tempo.

Ad esempio, ogni volta che viene fatta un'operazione di "push" di un file TS nel repository[G]di ICP, viene creata una nuova versione del file. Mediante un sistema di versionamento[G]del codice è possibile andare a vedere le modifiche che erano state caricate prima della versione attuale e confrontare i cambiamenti.

9.12.2 Software as a Service

Acronimo di "SaaS", che letteralmente si traduce in "Software come un servizio" o, in alcuni casi, anche come "Pay Per Use" ("Paga per l'uso"), è un modello di servizio del software nato negli anni 2000 che consiste di un un produttore che mette a disposizione un programma, tramite terze parti o direttamente, con modalità telematiche come una Web App. Alcune caratteristiche di questo modello sono:

- Il software non viene installato localmente nella macchina dell'utilizzatore finale, ma viene reso disponibile tramite la rete (ossia "cloud computing");

- L'utilizzatore non paga il software, bensì il suo utilizzo;
- Permette alle aziende di esternalizzare alcune parti di un meccanismo o di un sistema informatico, che permette loro di ridurre i costi di funzionamento e manutenzione.

Tra i tanti vantaggi dei SaaS ci sono:

- Optare per una soluzione SaaS permette di risparmiare in termini economici e prestazionali (un computer collegato ad una rete internet ed un browser possono essere più che sufficienti per usufruire del servizio - come una Web App -);
- In genere, queste soluzioni sono molto flessibili e permettono di implementare software "on demand" (su richiesta del cliente);
- È possibile fruire di questa tipologia di software da qualsiasi dispositivo ed in qualunque parte del mondo, indipendentemente dal fatto che quest'ultimo sia collegato ad una rete privata virtuale (a meno di eventuali blocchi imposti); Senza contare il costo pari o inferiore rispetto all'acquisto di una licenza tradizionale per l'uso di altri programmi, in quanto i costi di processo (gestione e manutenzione) risultano quasi inalterati rispetto al costo dell'acquisizione di una licenza (capitale e costo operativo).

9.12.3 Software House

Una software house è un'azienda specializzata nella creazione di prodotti software. Ad esempio, Adobe è una software che produce l'omonima suite di programmi (Adobe Photoshop, Adobe Illustrator etc.).

9.12.4 Superset

Quando si dice che «TypeScript è un superset di JavaScript» si intende dire che il primo è un linguaggio è più completo rispetto al secondo, in quanto presenta più funzionalità e permette di scrivere il codice con un certo rigore evitando l'effetto "vanilla" (un esempio sono i tipi delle variabili; similmente ad altri linguaggi di programmazione come il Java o il C++, anche TypeScript è fortemente tipizzato e, di conseguenza, viene fatto "type-checking" ogni qualvolta venga fatta un'assegnazione o ritornato un determinato tipo da una funzione, se il match non va a buon fine si ha un errore e la compilazione non va a buon fine).

9.13 T

9.13.1 Tenant

Con tenant si intende una singola istanza del software e dell'infrastruttura di supporto che serve un singolo cliente. Mediante una singola "tenancy" un utente può vedere i propri dati e le informazioni di suo interesse senza doverli condividere con nessun altro.

Si tratta di una modalità sicura, affidabile e personalizzabile con cui gestire le utenze di un software.

9.13.2 Toggle

Un toggle in Angular è una sorta di interruttore che può essere attivato o disattivato a seguito di un click con il mouse da parte dell'utente. In genere, l'interfaccia grafica di un toggle è simile a quella di un bottone.

9.13.3 Tools

I tools sono degli strumenti (dall'inglese, tool significa "strumento", "utensile") digitali che permettono di compiere una determinata funzione.

In ambito informatico e digitale, un tool è un "programma o un insieme di programmi che hanno un compito preciso, utili per uno scopo preciso".

Si distinguono da altri software in quanto hanno un'unica funzionalità specifica, ad esempio: Keycloack può essere definito un tools in quanto può essere utilizzato per configurare l'autenticazione degli utenti in una piattaforma o applicazione.

9.13.4 Two-way data binding

Il "two-way data binding[G]" è un meccanismo automatico di sincronizzazione che consente di effettuare il binding (in italiano, legame) bidirezionale di un elemento dell'interfaccia con il modello dei dati e viceversa, dal modello dei dati verso l'elemento dell'interfaccia.

In altri termini, quando un utente modifica un interfaccia (ad esempio, inserendo un campo di input o selezionando un elemento da un menù a tendina), ottiene subito qualcosa (per esempio, un messaggio di saluto, oppure la lista dei risultati ottenuti dalla chiamata di un microservizio).

Questo meccanismo è diverso rispetto al semplice "data binding", in quanto la sincronizzazione tra modello e vista è doppio (va in entrambe le direzioni), mentre il data binding singolo, in genere, va dal modello alla vista (esempio: un sito statico che, una volta effettuato l'accesso, mostra dei dati statici con i quali non è possibile interagire).

9.14 U

9.14.1 UAT

UAT è l'acronimo di "User Acceptance Testing" (o Test di Accettazione) e consiste nella fase di test formale dell'intero processo di collaudo (quality assurance del software) quando il prodotto è pronto per la consegna al cliente.

Il test viene eseguito per scoprire se un sistema software soffisfa i criteri di accettazione e per fare in modo che l'acquirente dia un riscontro positivo o meno sul prodotto sviluppato.

9.15 V

9.15.1 Versionamento

Per versionamento[G]si intende il controllo delle versioni del software, questa azione consente di tracciare i cambiamenti occorsi ad un file o ad un insieme di file. Inoltre, permette di riportare i file o l'intero progetto ad uno stadio precedente, visualizzare le modifiche nel corso del tempo, sviluppare più linee di lavoro in parallelo e identificare gli autori delle modifiche.

9.16 W

9.16.1 Warehouse Management System

Con il termine Warehouse Management System (abbreviato WMS) si intende un sistema di gestione del magazzino, che supporta l'azienda in tutte le fasi di organizzazione, coordinamento e controllo dei movimenti e processi logistici. Un software WMS si basa su una architettura client-server o web-server e deve essere in grado di integrarsi sia con la parte amministrativa che con altri eventuali software impiegati nei sistemi di stoccaggio e di movimentazione automatici presenti all'interno del magazzino.

In altre parole, il Warehouse Management System serve ad ottimizzare l'attività di tutte le risorse presenti nel magazzino: merci, uomini e mezzi. Dal tracking del prodotto, che ne identifica la posizione (come ubicazione del magazzino, area o scalera nel caso delle forme di formaggio) per lo stoccaggio, alle funzioni di picking, passando dal controllo delle giacenze e l'automatizzazione della ricezione merci fino alla gestione delle spedizioni e al tracciamento dei corrieri.

Non esiste un WMS universale, ma ne esistono diversi che possono essere adattati a seconda delle specifiche esigenze del cliente.

9.17 Y

9.17.1 YAML

YAML (inizialmente acronimo di "Yet Another Markup Language", successivamente reinterpretato in: "YAML Ain't Markup Language") è un formato per la serializzazione dei dati disponibile per tutti i linguaggi di programmazione e che sfrutta elementi derivati da vari linguaggi di programmazione (tra cui: Perl, C e Python). Il vantaggio di YAML è la sua leggibilità: un utente che apre un file YAML può comprenderne il significato ed i dati contenuti in esso in autonomia e senza dover studiare la sintassi di un linguaggio di programmazione.

I file YAML vengono usati principalmente nei casi in cui i dati debbano essere letti da un essere umano o come file di configurazione ed è una valida alternativa a XML e JSON.

10 Sitografia

Angular: https://aulab.it/notizia/306/angular-al-microscopio

Angular: https://www.spindox.it/it/blog/angular-timeline/#gref

Bad Request: https://www.rfc-editor.org/rfc/rfc7231#section-6.5.1

Branch: https://vitolavecchia.altervista.org/differenza-tra-fork-e-branch-in-informatica/

Continuous Integration: https://aws.amazon.com/it/devops/continuous-integration/

Fork: https://devdev.it/git-differenza-fork-clone-1205/

GitLab: https://www.geekandjob.com/wiki/gitlab

Hooks: https://italiancoders.it/ciclo-di-vita-di-un-componente-angular/

Merge: https://www.atlassian.com/it/git/tutorials/using-branches/git-merge#:~:text=L'operazione%20di%20merge%20%C3%A8,integrarle%20in%20un%20unico%20branch

Microservizio: https://aws.amazon.com/it/microservices/

MVVM: https://www.vivido.it/blog/2013/06/25/mvvm-in-pratica-breve-introduzione-alla-teoria/ https://www.vivido.it/blog/2013/06/25/mvvm-in-pratica-breve-introduzione-alla-teoria/

Pegno: https://www.treccani.it/vocabolario/pegno

Pegno rotativo: https://www.sistemiamolitalia.it/faq/in-cosa-consiste-il-pegno-rotativo/

WMS: https://news.beta80group.it/wms-che-cos-e-e-perche-e-fondamentale-per-la-logistica

YAML: https://www.geekandjob.com/wiki/yaml

Repository: https://vitolavecchia.altervista.org

Scrum: https://www.scrumalliance.org/ScrumRedesignDEVSite/media/ScrumAllianceMedia/Files%20and%20PDFs/Why%20Scrum/Core%20Scrum%20Translations/Core-Scrum-Italian.pdf

Versionamento: https://docs.italia.it/italia/docs-italia/docs-italia-guide/it/bozza/appendice-1.html