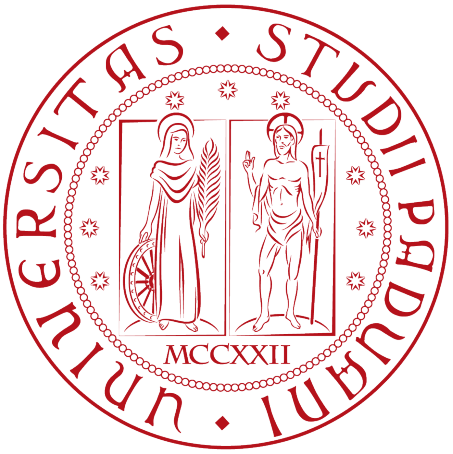
Università degli Studi di Padova

Dipartimento di Matematica "Tullio Levi-Civita"

Corso di Laurea in Informatica



Integrazione di processi PowerAutomate all'interno di applicazioni aziendali

Tesi di laurea

Relatore

Prof. Tullio Vardanega

Laureando

Silvio Nardo

Anno Accademico 2023-2024

Sommario

Il presente documento descrive il lavoro svolto dal laureando Silvio Nardo durante il periodo di stage presso l'azienda Wintech S.p.A. di Padova.
Esso è diviso in quattro capitoli: "Contesto aziendale" descrive l'azienda ospitante, con particolare attenzione ai servizi e prodotti offerti e alle metodologie lavorative; "Progetto di stage" narra il rapporto che l'azienda ha con gli stage universitari, descrivendo i diversi progetti proposti con particolare dettaglio allo stage da me svolto; In "Svolgimento stage" sono contenute le informazioni relative alle attività da me svolte durante il percorso di stage con spiegazione dei risultati raggiunti; "Verifica retrospettiva" infine analizza le conoscenze acquisite durante questo percorso e il loro rapporto con quelle fornite dall'università nel corso di laurea da me frequentato.
Lo stage si è svolto in conclusione del percorso di studi della laurea triennale in Informatica ed ha avuto la durata di circa trecentoventi ore.
L'obiettivo dello stage è stato compiere un'analisi al fine di valutare l'applicabilità delle pratiche DevOps a progetti aziendali realizzati con gli strumenti Power Automate e Power Apps.
Le soluzioni individuare durante le attività di ricerca sono state integrate ai processi aziendali mediante fasi di sviluppo collaborativo e individuale.

Convenzioni tipografiche

Gli acronimi e i termini di uso non comune menzionati vengono definiti nel glossario, situato alla fine del presente documento.
I termini in lingua straniera sono evidenziati con il carattere *corsivo*.

Indice

1 Contesto aziendale **1**
- 1.1 Wintech S.p.A. . . . 1
- 1.2 Servizi e prodotti . . . 1
  - 1.2.1 Servizi . . . 1
  - 1.2.2 Prodotti . . . 3
- 1.3 Processi aziendali . . . 6
  - 1.3.1 Metodologia *Agile* . . . 6
  - 1.3.2 Unità operative . . . 8
  - 1.3.3 Documenti e Certificazioni . . . 9
- 1.4 Tecnologie utilizzate . . . 9
- 1.5 Innovazione aziendale . . . 11

2 Progetto di *stage* **12**
- 2.1 Visione aziendale . . . 12
  - 2.1.1 Rapporto dell'azienda con gli *stage* . . . 12
  - 2.1.2 Progetti proposti . . . 12
- 2.2 *Stage* da me svolto . . . 14
  - 2.2.1 Obiettivi progettuali . . . 14
  - 2.2.2 Vincoli progettuali . . . 15
- 2.3 Obiettivi personali . . . 17

3 Svolgimento *stage* **19**
- 3.1 Analisi . . . 19
  - 3.1.1 Requisiti . . . 19
  - 3.1.2 Ambiente di lavoro . . . 20
  - 3.1.3 Tecnologie oggetto di *stage* . . . 21
  - 3.1.4 Tecnologie di supporto . . . 25
  - 3.1.5 DevOps . . . 27
  - 3.1.6 Analisi delle applicazioni aziendali . . . 28
  - 3.1.7 Angular . . . 30
- 3.2 Progettazione . . . 30
  - 3.2.1 *Scope* di Power Automate e Power Apps . . . 30
  - 3.2.2 Individuazione delle soluzioni tecnologiche . . . 31
  - 3.2.3 Soluzioni individuate per le fasi DevOps . . . 32
  - 3.2.4 Progettazione delle applicazioni aziendali . . . 36
- 3.3 Codifica e documentazione . . . 36
  - 3.3.1 PoC iniziali . . . 36
  - 3.3.2 Sviluppo DevOps . . . 37

3.3.3 Attività sulle applicazioni aziendali 41
3.3.4 Attività con Angular 42
3.4 Risultati raggiunti 43
3.4.1 Risultati qualitativi 43
3.4.2 Risultati qualitativi 44
3.4.3 Risultati quantitativi 44

Glossario **45**

Bibliografia **47**

Elenco delle figure

1.1 Schema del rapporto tra le connessioni LAN e WAN. 2
1.2 Soluzioni per l'automazione di processi. 3
1.3 Struttura dei moduli di Profis. 5
1.4 Organizzazione framework Scrum. 7
1.5 Rapporto tra le unità operative. 8
1.6 Esempio di un piano di Planner 10

2.1 Ciclo e fasi di DevOps. 13
2.2 Interfaccia grafica di esempio dell'IDE Visual Studio Code. 16
2.3 Flusso di Build, Delivery e Deployment in ambito DevOps. 17

3.1 Moduli formativi per Power Automate. 22
3.2 Esempio di un flusso Power Automate istantaneo. 23
3.3 Elemento Tree View per la gestione gerarchica dei componenti. 24
3.4 Esempio interfaccia di Jenkins. 26
3.5 Interfaccia grafica di Report di visita. 29
3.6 Funzionalità Version history nei flussi Power Automate. 31
3.7 Funzionalità Verifica App nelle applicazioni Power Apps. 33
3.8 Funzionalità Test nei flussi Power Automate. 34
3.9 Vista parziale dell'interfacccia atta a monitorare un flusso. 36
3.10 Flusso per testare condizioni, approvazioni ed esecuzione di *script*. . . 37
3.11 Configurazione dei permessi e selezione dei repository connessi ad una GitHub App. 39
3.12 Flusso di esempio per ricevere e risondere a chiamate HTTP. 40
3.13 Parte dello *script* responsabile per l'ottenimento del *token* di autenticazione. 40
3.14 Parte dello *script* responsabile per l'analisi dell'esito dei *test* su flussi Power Automate. 40
3.15 Grafico dell'esito dei *test* generato da Jenkins. 43

Elenco delle tabelle

3.1 Tabella dei requisiti progettuali. 19
3.2 Documenti che ho prodotto durante lo *stage*. 44
3.3 Software che ho prodotto durante lo *stage*. 44

Capitolo 1

Contesto aziendale

1.1 Wintech S.p.A.

Winning Technologies, in sigla Wintech S.p.A., venne fondata nel 1987 dall'attuale amministratore delegato Massimo Gallotta.
Essa si occupa di *system integration*, ovvero il processo di combinazione di diversi componenti *software* e infrastrutture in un sistema unico e coeso, al fine di garantire che le parti lavorino insieme in modo efficiente e sinergico.
Questo processo include la connessione di sistemi esistenti con nuove tecnologie per migliorare le funzionalità, la condivisione dei dati e il coordinamento delle operazioni.
Tale processo avviene per Wintech nell'ambito delle tecnologie dell'informazione e della comunicazione (o ICT), ovvero l'insieme delle tecnologie che permettono l'elaborazione, la gestione, la trasmissione e la comunicazione delle informazioni.
Queste ultime comprendono soprattutto l'ambito IT il quale si riferisce all'uso di tecnologie, dispositivi e sistemi per creare, memorizzare, elaborare, scambiare e proteggere informazioni e dati.

L'azienda, la quale conta più di 90 dipendenti, ha sede principale situata a Padova ma, grazie al suo sviluppo, ha acquisito filiali a Milano e a Bassano del Grappa.
Wintech ha stretto svariate *partnership* con aziende *leader* di mercato come IBM, Microsoft e HP grazie anche alla propria affidabilità e solidità finanziaria.
Inoltre, possiede diverse società partecipate, tra cui Sistemi S.p.A., con le quali è attiva una forte collaborazione.
I clienti di Wintech comprendono grandi imprese (tra cui BMW, Helvetia e Lindt), Piccole e Medie Imprese, professionisti, aziende assicurative e imprese finanziarie.

1.2 Servizi e prodotti

1.2.1 Servizi

I servizi offerti da Wintech comprendono diverse aree tecnologiche:

- **Infrastrutture IT**: vengono offerte soluzioni per modernizzare e gestire le infrastrutture aziendali sfruttando tecnologie basate su cloud, ovvero utilizzando risorse IT (come *server*, *storage*, *database*, *software* e reti) attraverso *Internet*,

anziché dipendere da infrastrutture fisiche locali.
Inoltre vengono realizzate progettazioni di connessioni di rete LAN e WAN. Esse sono delle reti di comunicazione informatica che presentano alcune differenze: le *Local Area Network* (LAN) permettono di connettere localmente dispositivi, come *computer*, stampanti e *server*, all'interno di un'area limitata, ad esempio un edificio, un ufficio o una scuola. Le LAN consentono la condivisione di risorse e la comunicazione tra dispositivi con velocità elevate e basse latenze; le *Wide Area Network* (WAN), a differenza, sono reti di comunicazione che collegano dispositivi o reti locali (LAN) su una vasta area geografica, come città, nazioni o continenti. Esse utilizzano infrastrutture pubbliche o private per trasmettere dati su lunghe distanze, consentendo la condivisione di informazioni tra utenti e sistemi remoti.

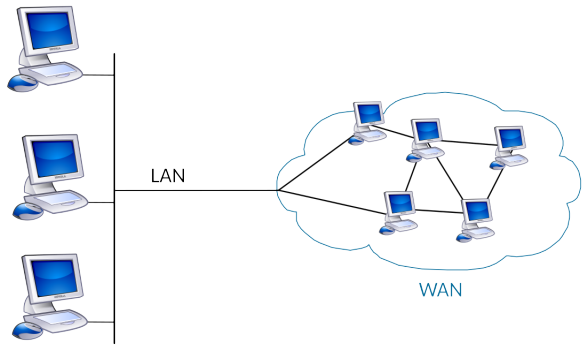


Figura 1.1: Schema del rapporto tra le connessioni LAN e WAN.
Fonte: https://en.wikipedia.org/wiki/Wide_area_network.

Ulteriori servizi offerti sono la gestione di *backup* sicuri e supporto IT, conforme agli standard definiti nell'*Information Technology Infrastructure Library*. Quest'ultimo, denominato anche ITIL, rappresenta l'insieme delle linee guida per la gestione dei servizi IT al fine di migliorarne l'erogazione, il supporto e la qualità, mantenendo un allineamento con gli obbiettivi aziendali.

- ***Cyber* sicurezza**: per garantire la continuità aziendale viene fornita protezione contro i rischi informatici mediante difesa delle reti e dei dispositivi, segmentazione della rete, servizi gestiti e soluzioni di *disaster recovery*. Quest'ultimo rappresesnta il processo e insieme di strategie volte a ripristinare sistemi, dati e infrastrutture IT critiche dopo un evento catastrofico, come guasti *hardware*, attacchi informatici, disastri naturali o errori umani.

- **Automazione di processi di *Business***: vengono forniti strumenti per automatizzare i processi aziendali come flussi approvativi, gestione documentale e metodologie DevOps.
  Queste ultime non rappresentano solo un insieme di pratiche, ma una vera e propria cultura metodologica. Essa unisce sviluppo *software* (Dev) e operazioni IT (Ops) per migliorare la collaborazione, l'efficienza e la velocità nella creazione,

distribuzione e gestione delle applicazioni, mantenendo alta la qualità e la stabilità dei servizi.

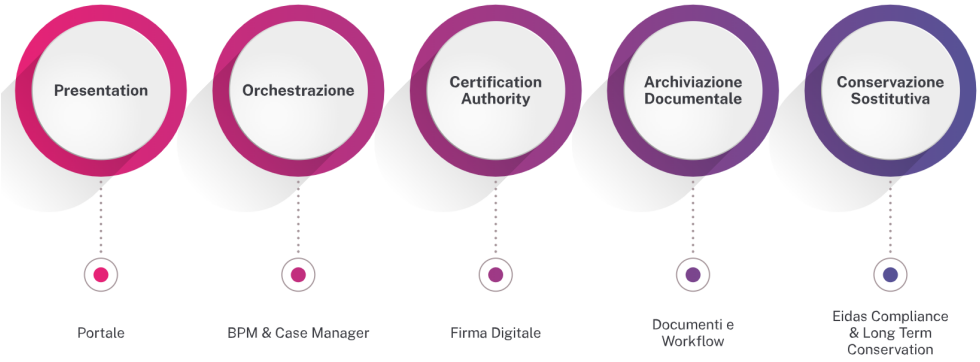


Figura 1.2: Soluzioni per l'automazione di processi.
Fonte: https://www.wintech.it/business-automation/.

- **Soluzioni ERP**: le *Enterprise Resource Planning* (ERP) sono sistemi *software* integrati utilizzati per gestire e ottimizzare i processi aziendali fondamentali come contabilità, gestione delle risorse umane, produzione, vendite, sistemi di gestione dei clienti (*Customer Relationship Management*) e *marketing*.
  Gli ERP centralizzano i dati in un'unica piattaforma, migliorando la condivisione delle informazioni, l'efficienza operativa e il processo decisionale per la produzione industriale e l'implementazione di soluzioni per industria 4.0/5.0.

Viene offerta consulenza al cliente al fine di fornire supporto tecnico e comprendere le sue necessità.
Successivamente viene identificata una soluzione adeguata, la quale può essere sviluppata partendo da un prodotto aziendale già consolidato apportando le opportune personalizzazioni, oppure realizzando un prodotto nuovo.
In seguito allo sviluppo di un prodotto, vengono forniti servizi di formazione, assistenza e manutenzione dello stesso.

1.2.2 Prodotti

I principali prodotti offerti dall'azienda sono:

Spring

Supporta le PMI gestendo le informazioni relative alle attività di diverse aree aziendali come:

- Amministrativa
- Vendite
- Acquisti
- Logistica e magazzino

- Analisi e *reporting*
- Gestione documentale

Studio

Offre servizi legati all'organizzazione e alla gestione dello studio professionale come:

- Calendari condivisi
- Gestione efficiente delle pratiche
- Emissione di parcelle
- Fatture elettroniche

Job

Fornisce servizi legati alla gestione del personale come:

- Gestione delle presenze
- Amministrazione delle Risorse Umane
- Elaborazioni dei cedolini
- Analisi sui costi del personale

Sportello

Propone servizi legati alla gestione dei clienti come:

- Gestione della fatturazione online
- Gestione degli incassi e dei pagamenti
- Condivisione, validazione e conservazione digitale dei documenti

eSolver

Mette a disposizione servizi legati alla gestione dei processi per aziende di produzione, di servizi, di commercio all'ingrosso e al dettaglio. Le principali funzionalità comprendono:

- Amministrazione e finanza
- Acquisti e rapporti di fornitura
- Vendite e attività commerciale
- Logistica
- Archiviazione e conservazione documentale
- Gestione risorse produttive
- Produzione manifatturiera
- Controllo di gestione
- *Reporting* e analisi

Profis

Fornisce servizi legati alle aree di attività dello studio del commercialista come:

- Area fiscale dei bilanci
- Contabilità e digitalizzazione dei processi di fatturazione

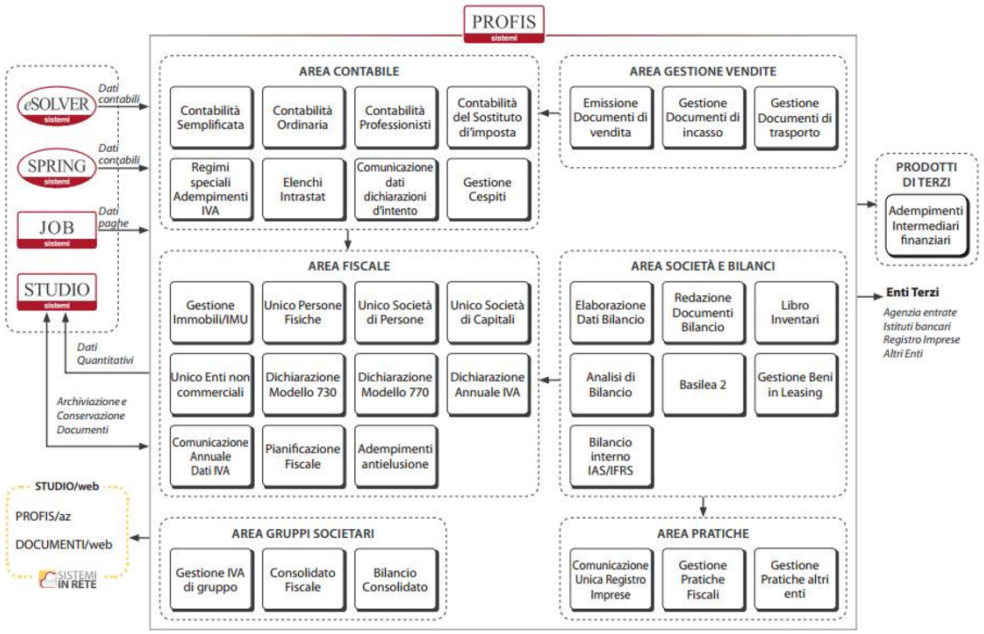


Figura 1.3: Struttura dei moduli di Profis.

WOW

"*World Of Wintech*" (WOW) viene considerato non solo un utile prodotto per i clienti, ma soprattutto un potente strumento gestionale aziendale utilizzato da tutti i dipendenti.
Essa è una soluzione basata interamente sul cloud, altamente personalizzabile ed integrabile con i sistemi gestionali e applicativi già in possesso da un cliente.
WOW permette di utilizzare svariati moduli gestionali in un'unica interfaccia utente e di gestire i documenti grazie alla piattaforma FileNet - IBM, permettendo operazioni di collaborazione in tempo reale.
Quest'ultima è una piattaforma di back-end, ovvero parte di un'applicazione o di un sistema informatico, generalmente non visibile all'utente, che gestisce la logica, il *database*, la sicurezza e la gestione dei dati, ed è responsabile della comunicazione con l'interfaccia utente.

I principali moduli che lo compongono sono:

- **Home**: organizzazione delle risorse aziendali interne.
- **Opportunità**: tracciamento delle trattative commerciali.

- **Marketing**: gestione delle azioni di *marketing*.
- **Campagne commerciali**: assegnazione delle attività e gestione campagne commerciali.
- **Anagrafiche**: profilazione clienti e contatti aziendali.
- **Commesse**: gestione informazioni anagrafiche di commessa al fine di categorizzare i documenti archiviati.
- **Add-In O365**: connessione con Outlook365 per l'archiviazione dei messaggi di posta elettronica.
- **Registro accessi**: gestione degli accessi in azienda da parte di ospiti e collaboratori.
- **Documenti**: archiviazione dei documenti mediante la piattaforma IBM e l'integrazione con Microsoft365.

WOW si integra con il prodotto eSolver al fine di gestire tutte le principali aree di interesse aziendale.

1.3 Processi aziendali

1.3.1 Metodologia *Agile*

Grazie all'utilizzo di WOW e alle politiche aziendali, è molto forte la collaborazione e la condivisione di informazioni tra i dipendenti. Il tutto è reso maggiormente dinamico ed efficiente grazie all'adozione della metodologia *Agile*.
Essa è un approccio alla gestione progettuale che promuove la collaborazione e il dialogo tra membri del *team* e con i clienti al fine di condividere costantemente *feedback* e rispondere prontamente ai cambiamenti.
I quattro valori che descrivono la filosofia *Agile* sono:

- Gli individui e le interazioni più che i processi e gli strumenti
- Il *software* funzionante più che la documentazione esaustiva
- La collaborazione col cliente più che la negoziazione del contratto
- Rispondere al cambiamento più che seguire un piano

In Wintech viene adottato il *framework Agile* "*Scrum*" per la gestione dei progetti.
Esso prevede una divisione temporale dell'avanzamento dei lavori organizzata per "*Sprint*", i quali in azienda hanno una durata variabile di circa un mese ciascuno.
All'inizio di ogni *Sprint* esso viene pianificato definendo tutte le attività da svolgere.
Quotidianamente è previsto un breve incontro tra i membri del *team* in modo da confrontarsi sullo stato di avanzamento dei lavori e su eventuali problemi riscontrati.
Al termine dello *Sprint* viene fatta una revisione per analizzare i risultati ottenuti e avviene un'analisi retrospettiva al fine di far emergere gli aspetti positivi e quelli

migliorabili riguardo il suo svolgimento, la collaborazione del *team* e l'utilizzo degli strumenti forniti.

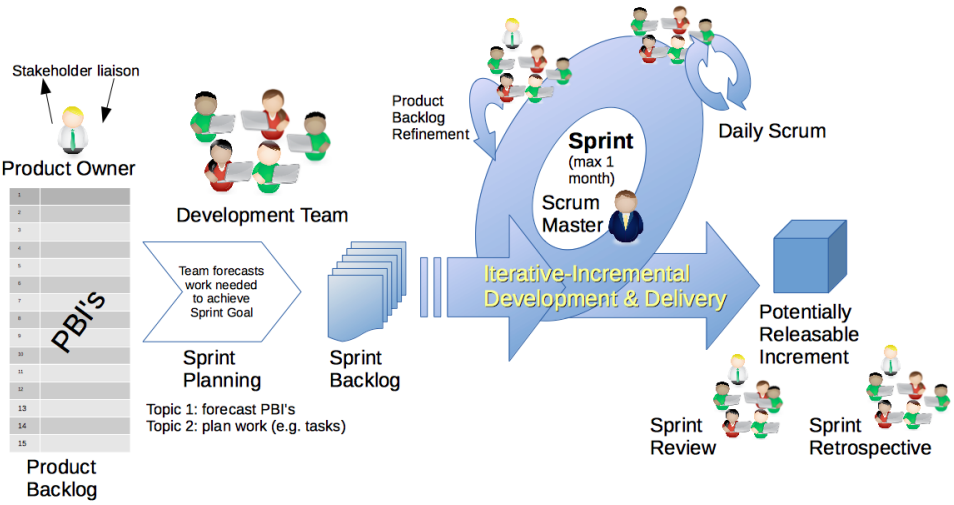


Figura 1.4: Organizzazione framework Scrum.

Fonte: https://commons.wikimedia.org/wiki/File:Scrum_Framework.png.

Come mostrato nell'immagine, tale processo è iterativo, il che implica che sono ammesse modifiche che possono talvolta compromettere quanto già realizzato. Tuttavia, ciò consente di riallineare il prodotto alle richieste del cliente, grazie ai suoi *feedback*, evitando di divergere dalla sua visione e garantendo un risultato conforme con le aspettative.
Il *framework* adottato prevede tre figure fondamentali:

Product Owner

Colui che si focalizza sulle esigenze aziendali, dei clienti e di mercato. Egli definisce le attività da compiere e la loro priorità. In azienda tale ruolo è responsabilità del *project manager*, ovvero il responsabile della pianificazione, esecuzione e supervisione di un progetto. Egli gestisce le risorse, coordina il *team*, monitora i progressi, risolve i problemi e comunica con clienti e fornitori per garantire il successo del progetto assicurandosi che gli obiettivi vengano raggiunti nei tempi, nel *budget* e con la qualità prevista.

Scrum Master

Colui che organizza le risorse per la pianificazione dello *Sprint* e coordina lo svolgimento delle procedure e cerimonie *Scrum* garantendo una corretta applicazione delle sue filosofie. In azienda tale ruolo è responsabilità del *project manager*.

Team di sviluppo

Il gruppo di lavoro, il quale gode di una buona autonomia nell'avanzamento e assegnazione dei lavori, è composto da una decina di membri con diverse competenze. Tra loro è presente una forte collaborazione e dialogo in modo da condividere le proprie conoscenze e progredire con lo stato dei lavori in modo efficace.

In Wintech i ruoli *Scrum* non sono rappresentati sempre dalle stesse figure bensì vengono ricoperti da individui differenti seguendo un preciso ordine e una rotazione specifica.
Inoltre, le cerimonie possono avvenire per sottogruppi in base alle circostanze e alle esigenze incontrate. Per esempio, può capitare che avvengano determinate cerimonie tra *project manager* e gli elementi del *team* più esperti mentre, in un secondo momento, avvenga un ulteriore incontro tra tali *senior* e il resto del *team*.
Durante questi incontri, a sviluppatori con poca esperienza o risorse esterne non viene necessariamente richiesto di contribuire fornendo la propria visione relativamente alla decisione dei *task* o alla personale analisi su dati e stime mostrate.

1.3.2 Unità operative

Anche denominate *Business unit* (o BU), rappresentano le strutture o divisioni all'interno di un'organizzazione e si occupano di specifiche attività operative per il raggiungimento degli obiettivi aziendali. Ogni unità operativa è generalmente responsabile di un'area funzionale specifica, come la produzione, la logistica, la vendita, o il servizio clienti.
Le principali BU in cui è strutturata Wintech sono:

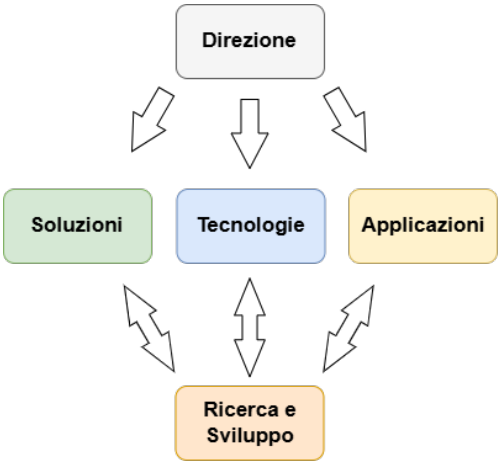


Figura 1.5: Rapporto tra le unità operative.

Applicazioni

Formata da un gruppo di consulenti applicativi i quali si occupano di consulenza, integrazione e assistenza sui *software* gestionali aziendali nei confronti dei clienti.

Soluzioni

Comprende programmatori con competenze trasversali in grado di definire, progettare e realizzare le soluzioni necessarie per soddisfare le richieste dei clienti.

Tecnologie

Al suo interno persone con competenza sistemistica monitorano e gestiscono i *server* aziendali al fine di garantire continuità operativa a tutti i clienti in cloud. Il personale si dedica all'assistenza dei clienti sia da remoto che in sede, proponendo componenti *hardware* adatte alle esigenze.

Ricerca e Sviluppo

Composta da un *team* di programmatori avente una maggiore libertà in termini di sperimentazione tecnologica e metodologica. Durante il corso dello *stage* ho fatto parte di questa unità operativa.

1.3.3 Documenti e Certificazioni

Durante il ciclo di vita di un progetto vengono generati un insieme di documenti atti a descriverlo e a normare le attività in esso svolte.
I documenti che descrivono le procedure e i metodi da adottare per garantire sicurezza e qualità al prodotto sono spesso comuni a più progetti, pertanto, essi vengono duplicati e modificati solo nei casi in cui le tecnologie o le circostanze rendono tali modifiche necessarie.
Tra questi documenti sono presenti i "Documenti di sviluppo sicuro" i quali normano le procedure legate al ciclo di vita del progetto e alle buone pratiche da adottare.
Differentemente, alcuni documenti devono venire prodotti più specificatamente per ogni progetto. Tra questi sono presenti: l'analisi dei requisiti, lo schema architetturale, le criticità, il manuale utente e di installazione, il verbale di collaudo e il "*Bill of Material*" contenente tutti i *software* di terze parti necessari all'utilizzo.
I processi aziendali definiti hanno reso possibile l'ottenimento delle certificazioni UNI CEI ISO/IEC 27001:2017 IAF 33 e UNI EN ISO 9001:2015 IAF 33,39,37 le quali attestano rispettivamente la conformità del sistema di gestione per la sicurezza delle informazioni e la qualità della gestione aziendale in termini di efficacia, efficienza e soddisfazione dei clienti.

1.4 Tecnologie utilizzate

All'interno dell'azienda, più specificatamente nelle unità operative Ricerca e Sviluppo e Soluzioni, vengono utilizzate le seguenti tecnologie:

- Per la pianificazione e il monitoraggio delle attività progettuali vengono utilizzate piattaforme dedicate come Taiga e Microsoft Planner, strumenti versatili che

permettono di organizzare in modo collaborativo ed efficiente i *task* e le attività ad alto livello definite casi d'uso o UC dalla sua traduzione inglese *use case*. Tale termine rappresenta la descrizione di una sequenza di azioni che il sistema esegue in risposta a una richiesta dell'utente. Sono incluse le varie possibilità di esito (ad esempio, scenari di successo o di errore), al fine di definire il suo comportamento in relazione ai requisiti prestabiliti.

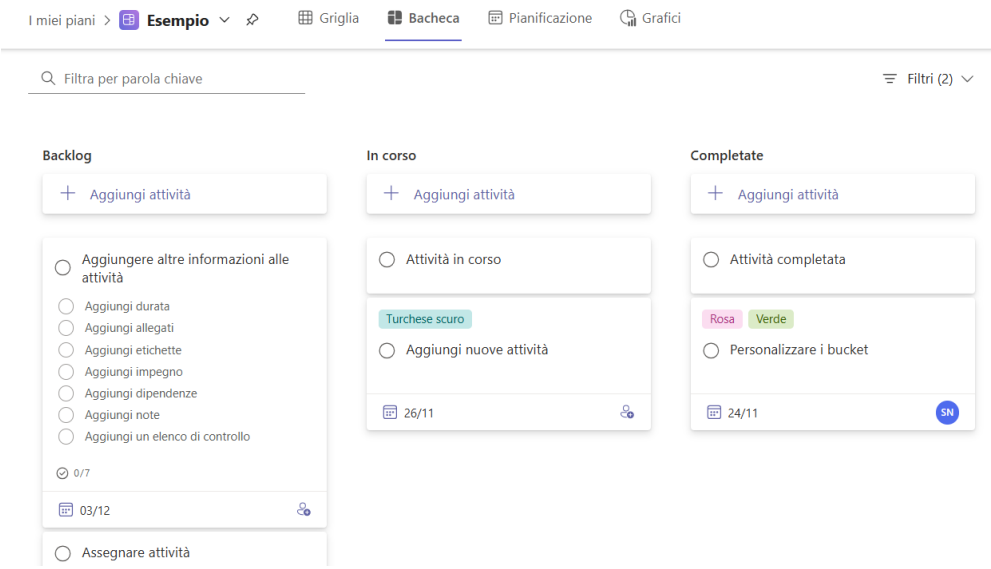


Figura 1.6: Esempio di un piano di Planner

- La collaborazione tra i membri del personale è agevolata da strumenti come Microsoft Teams e Outlook, i quali permettono di: scambiare ed organizzare messaggi ed *e-mail*, gestire eventi come riunioni e *meeting* in un apposito calendario sincronizzato ed effettuare videochiamate per agevolare l'interazione e la collaborazione tra colleghi.
- Gli sviluppatori possono scegliere liberamente gli ambienti di sviluppo più adatti alle loro esigenze, anche se i principali strumenti utilizzati sono Visual Studio Code e IntelliJ IDEA, che offrono supporto per molteplici linguaggi di programmazione e *framework*.
- I sistemi operativi prevalentemente utilizzati in azienda sono Windows 11 e Windows 10 in modo da garantire un ambiente compatibile con la maggior parte degli strumenti *software*.
- Il versionamento del codice sorgente è gestito tramite Git, con il supporto di piattaforme come GitHub e GitLab, che assicurano un controllo efficace delle versioni e agevolano la collaborazione.
- Per il controllo e l'automazione del ciclo di vita del *software* viene utilizzato Jenkins, uno strumento che permette di ottimizzare i processi di sviluppo e distribuzione tramite le pratiche di sviluppo *software* Continuous Integration(CI) e Continuous Deployment(CD).

La prima prevede che i membri di un *team* integrino frequentemente il loro lavoro svolto in un *repository* comune, generalmente più volte al giorno. Il suo obiettivo è ridurre i problemi di integrazione, migliorare la qualità del *software* e accellerare il ciclo di sviluppo, permettendo modifiche più rapide e sicure.
La seconda pratica invece prevede che ogni modifica al codice che supera i *test* automatici venga automaticamente distribuita in ambiente di produzione senza intervento manuale. L'automazione completa ottenuta con l'adozione di tale processo riduce i rischi introdotti dall'errore umano e migliora la velocità e l'affidabilità dei rilasci.

- La qualità del *software* è garantita attraverso strumenti come SonarQube per l'analisi statica del codice, JUnit per i *test* di unità ed integrazione, JMeter per i *test* di carico e *stress test*.
  Per migliorare il processo di *test* viene utilizzato, dove necessario, il *framework* Mockito per simulare il comportamento di specifici componenti o dipendenze esterne.
  Viene inoltre utilizzato lo strumento OWASP ZAP per testare il soddisfacimento di molteplici requisiti di sicurezza.

- Le tecnologie principali utilizzate per la creazione e gestione della documentazione aziendale comprendono l'insieme Microsoft Office 365 mentre, per lo sviluppo *software*, vengono utilizzati soprattutto il linguaggio Java e il *framework* Angular.

- Per la gestione e configurazione di un *server web* e di basi di dati, vengono utilizzate rispettivamente Apache Web server e SQL server.

1.5 Innovazione aziendale

L'azienda non sottovaluta l'importanza dell'innovazione in un ambiente lavorativo in costante aggiornamento come quello IT. Lo dimostra l'importanza che viene data all'unità operativa Ricerca e Sviluppo, alla quale ho partecipato approfondendo tecnologie nuove per l'azienda e valutando nuovi approcci organizzativi. Questi ultimi includono l'implementazione, per le nuove tecnologie adottate, di metodologie DevOps. Tali processi atti all'automazione vengono già applicati per tecnologie consolidate aumentando l'efficacia, l'efficienza e la qualità dei lavori.
Wintech riconosce il valore aggiunto che i giovani possono portare e, da diversi anni, offre la possibilità di svolgere *stage* universitari per formare e inserire nel mondo del lavoro studenti prossimi alla laurea.
Spesso, per questi ultimi, si concretizza l'opportunità di proseguire lavorativamente i rapporti con l'azienda, pertanto, il *team* di sviluppo è caratterizzato da un'età media relativamente giovane, pur mantenendo al suo interno figure con consolidata esperienza. Inoltre, al fine di accrescere le potenzialità delle proprie risorse, l'azienda fornisce corsi di formazione mirati per il personale.

Capitolo 2

Progetto di *stage*

2.1 Visione aziendale

2.1.1 Rapporto dell'azienda con gli *stage*

Wintech identifica gli *stage* universitari come strumenti utili alla formazione professionale dei laureandi al fine di renderli risorse produttive non solo per il periodo di *stage* definito, ma soprattutto per il periodo successivo. È infatti consuetudine per l'azienda decidere di proseguire i rapporti lavorativi con gli studenti, pertanto, è nel suo interesse rendersi disponibile fornendo approfondimenti e attività formative in ambito tecnologico e riguardo i processi aziendali.
Per proporre la propria disponibilità e avere un primo contatto con gli studenti, Wintech partecipa da diversi anni all'evento dedicato STAGE-IT promosso da Confindustria Veneto Est in collaborazione con i Dipartimenti di Scienze Statistiche, Matematica e Ingegneria dell'Informazione dell'Università di Padova, al fine di favorire l'incontro tra aziende associate e gli studenti per esporre i progetti ICT proposti.
Successivamente, organizza colloqui conoscitivi al fine di spiegare la propria visione aziendale e i propri progetti di *stage* agli studenti.
Essi non sono pensati per essere esclusivamente didattici ma forniscono un tangibile valore all'azienda in quanto si basano sulle tecnologie e metodologie da loro realmente utilizzate o che prevedono di integrare.
Lo stagista non è pertanto visto solo come uno studente ma viene integrato nel *team* con il quale lavora a stretto contatto acquisendo nuove competenze sia collaborativamente, mediante meeting e sviluppo cooperativo, che autonomamente, mediante fasi di ricerca e sviluppo individuali.

2.1.2 Progetti proposti

I progetti di *stage* proposti dall'azienda, contestualmente al periodo della nostra collaborazione, derivano dal valore aggiunto che l'automazione dei processi aziendali può portare in ambiti di pianificazione progettuale, per ambienti di sviluppo consolidati e per tecnologie oggetto di ricerca.
Tale processo permette di realizzare efficientemente prodotti con maggiore qualità riducendo sensibilmente i tempi e i costi dei lavori.
Inoltre, automatizzare attività ben definite riduce sensibilmente la possibilità di intro-

duzione manuale di errori e ne permette una rapida individuazione.
Durante la durata dello *stage*, noi stagisti abbiamo collaborato più volte trovando dei punti di incontro tra i propri progetti e rimanendo aggiornati riguardo ai rispettivi risultati ottenuti tramite appositi incontri.
I tre progetti proposti in questione sono:

Applicativi di DevOps in ambito Sistemi e Office365

Lo scopo dello *stage* da me svolto è stato quello di compiere un'analisi per verificare l'applicabilità delle pratiche di DevOps su progetti basati sull'utilizzo di tecnologie Office365, con lo scopo di integrarle nell'infrastruttura aziendale in maniera rapida e *low-code*, ovvero con limitata necessità e presenza di codice scritto dallo sviluppatore.
Più specificatamente sono stati utilizzati i programmi Power Automate e Power Apps: il primo permette di realizzare flussi di attività connesse con gli applicativi Microsoft, mentre il secondo agevola lo sviluppo di applicazioni aziendali.
Le soluzioni individuate durante la ricerca vengono introdotte nel contesto aziendale tramite fasi di sviluppo sia implementando tali risultati a reali applicazioni aziendali, realizzate collaborando con il *team*, sia producendo *Proof of Concept* (PoC), ovvero elementi dimostrativi sviluppati al fine di provare la fattibilità di un prodotto mediante l'utilizzo delle tecnologie e degli strumenti definiti. Esso rappresenta quindi una versione di prova prototipale e non ha lo scopo di divenire il prodotto finale ma viene realizzato a supporto dell'analisi progettuale.

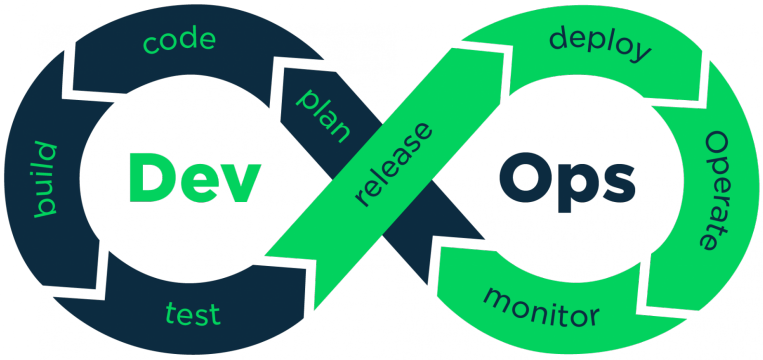


Figura 2.1: Ciclo e fasi di DevOps.

Integrazione sistemi di pianificazione di progetto

Lo scopo dello *stage* è stado compiere un'analisi per verificare l'utilizzo degli strumenti di pianificazione aziendale, nello specifico Planner e Taiga, e lo sviluppo di una soluzione atta ad automatizzare la comunicazione e sincronizzazione di tali strumenti.
Per raggiungere tale obiettivo è stato possibile utilizzare le *Application Programming Interface* (API): in italiano "Interfacce di Programmazione dell'Applicazione". Esse rappresentano un insieme di procedure che permettono la comunicazione e lo scambio di dati tra diversi componenti *software*. L'utilizzo di API permette di semplificare l'integrazione tra prodotti e servizi informatici rendendo disponibili funzionalità esterne senza doverne conoscere l'effettiva implementazione.

Applicativi di DevOps in ambito Sistemi

Lo scopo dello *stage* è stato compiere un'analisi per verificare l'applicabilità delle pratiche di DevOps all'ambiente di sviluppo Sistemi, con lo scopo di integrarle nell'infrastruttura aziendale in ambiti ben definiti.
Le soluzioni individuate durante la ricerca vengono introdotte nel contesto aziendale tramite fasi di sviluppo.

2.2 *Stage* da me svolto

2.2.1 Obiettivi progettuali

Gli obiettivi attesi del mio *stage* esprimono la necessità di uno studio approfondito riguardo all'ambiente Office365 e più specificatamente delle tecnologie Power Automate e Power Apps.
Tale studio deve essere accompagnato da fasi di sviluppo al fine di integrare e testare le soluzioni individuate durante la ricerca.
Ogni conclusione raggiunta deve essere opportunamente documentata in modo da definire se e come le metodologie DevOps aziendali siano applicabili all'ambiente Office365. Al *team* di sviluppo e al *tutor* aziendale dovranno essere esposte delle presentazioni relative a quanto appreso.
Tali obiettivi si considerano raggiunti se, al termine dello *stage*, i *software* e i documenti da me realizzati, contenenti i risultati delle mie ricerche e le indicazioni per l'applicazione delle soluzioni individuate, sono effettivamente introdotti e utilizzati nell'ambiente di lavoro, con la conferma del *tutor* aziendale circa il raggiungimento della qualità attesa.
Rientra tra gli obiettivi comprendere l'importanza e l'impatto che l'automazione possiede quando applicata correttamente ai processi lavorativi e comprendere le potenzialità delle metodologie DevOps e delle tecnologie in oggetto. Questo obiettivo è soddisfatto con l'approvazione aziendale riguardo alla corretta comprensione e applicazione delle pratiche di DevOps ai processi aziendali.
Sono previsti tra gli obiettivi di *stage* anche delle fasi di sviluppo in collaborazione con il *team* dell'unità operativa Ricerca e sviluppo, al fine di comprendere e migliorare prodotti aziendali realizzati utilizzando le tecnologie in oggetto. I documenti di sviluppo sicuro relativi ai progetti realizzati precedentemente allo *stage* devono essere adattati in caso di modifiche al *software* o alle metodologie applicate.
La scelta di assegnare all'unità operativa Ricerca e sviluppo lo studio e l'utilizzo delle tecnologie Power Automate e Power apps deriva dal bisogno di identificare un nuovo metodo che permetta di risparmiare tempo nelle fasi di sviluppo senza rinunciare alla qualità dei prodotti.
Esse, avendo una natura *low-code* ed essendo pensate per fornire nativamente un'agevole connessione con gli altri servizi Microsoft, permettono di applicare le personalizzazioni richieste da un cliente in maniera più rapida e sicura rispetto allo sviluppo delle stesse soluzioni tramite linguaggi di programmazione e altri strumenti consolidati.
È quindi mio obiettivo fondamentale verificare l'idoneità di tali strumenti al soddisfacimento del bisogno descritto.
L'ultimo obiettivo espresso dall'azienda è la realizzazione di una presentazione finale collaborativamente tra i tre stagisti al termine dei loro periodi di *stage* e la sua esposizione al presidente di Wintech e ai responsabili.

2.2.2 Vincoli progettuali

Vincoli temporali

Lo *stage* ha avuto inizio il giorno 11 settembre ed è terminato il giorno 20 novembre. Esso ha avuto luogo interamente in presenza nella sede di Padova, nella quale mi sono recato dalle ore 9:00 alle ore 13:00 e dalle ore 14:00 alle ore 18:00, per un totale di 320 ore lavorative.
Tale periodo è stato suddiviso dall'azienda in otto fasi pianificate ben definite, come dichiarato nel documento "Progetto Formativo", fornitomi dal *tutor* aziendale, generato all'inizio dello stage.
Queste indicazioni sono state intese come delle linee guida per indirizzare l'avanzamento dello stage e non come dei vincoli ferrei, pertanto, ho avuto la libertà di autogestire le mie attività dando priorità alle necessità e alle direttive fornitemi con l'avanzamento dei lavori.
Esse sono:

Fase 1: dal 11/09/2024 al 13/09/2024 (24h)

Durante i primi giorni lo studente, insieme agli *stakeholder*, prende confidenza con l'ambito di riferimento per lo *stage* e visiona come il ciclo di vita del *software* e le metodologie DevOps siano state applicate in azienda con le pratiche di sviluppo sicuro. Inoltre vengono analizzate le funzionalità di versionamento con Git e Git Server.

Fase 2: dal 16/09/2024 al 20/09/2024 (40h)

Con l'obiettivo di determinare, insieme ai *developer* dell'ambito di riferimento, la possibilità di versionare agevolmente con gli strumenti a disposizione, lo studente utilizza quanto appreso con Git per versionare un ambiente di prova e dimostra le funzionalità di salvataggio, modifica, assegnazione di autori e date, gestione dei *branch*, *merge* e *tag*.
Dopo aver ottenuto un riscontro, lo studente compila con i *developer* il documento di DevOps di sviluppo sicuro in modo da normare quanto appreso e deciso.

Fase 3: dal 07/10/2024 al 11/10/2024 (40h)

L'esperienza ottenuta dallo studente fino a questa fase viene fornita ai *team* tramite la realizzazione di documentazione e presentazione al fine di condividere i *feedback* su quanto appreso.

Fase 4: dal 14/10/2024 al 18/10/2024 (40h)

È obiettivo dello studente valutare la possibilità di utilizzare IDE non proprietari nell'ambito Office365. Tale acronimo indica il termine *Integrated Development Environment* (ambiente di sviluppo integrato), ovvero un'applicazione che permette di sviluppare codice sorgente in una moltitudine di diversi linguaggi di programmazione e può fornire servizi come *testing*, analisi statica del codice, *debugging*, compilazione e integrazione con sistemi di versionamento.

In seguito viene valutata la possibilità, per le tecnologie oggetto di *stage*, di utilizzare lo strumento di analisi statica SonarLint negli IDE o altri strumenti similari utilizzando come guida i documenti di sviluppo sicuro.
Ogni considerazione viene riportata nella relativa documentazione e viene eseguita una presentazione agli *stakeholder*.

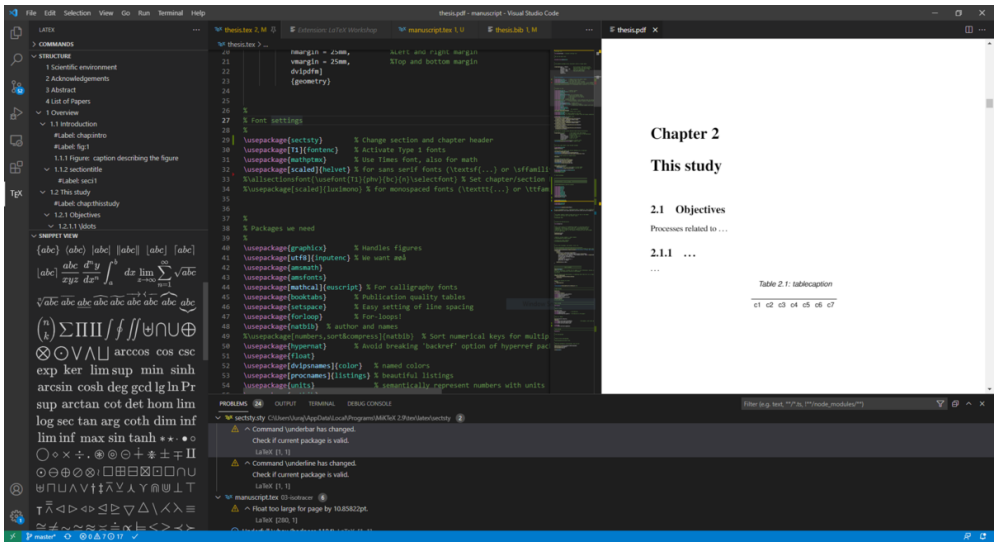


Figura 2.2: Interfaccia grafica di esempio dell'IDE Visual Studio Code.

Fonte:
https://commons.wikimedia.org/wiki/File:VsCode_LaTex_Workshop.png.

Fase 5: dal 21/10/2024 al 25/10/2024 (40h)

Lo studente discute con il *team* su come viene utilizzato Jenkins in azienda e come è stato analizzato e normato nei documenti di sviluppo sicuro. Al fine di prendere confidenza con tale strumento, viene realizzato un *job* che compili un modulo di esempio.
Viene determinato se sia stato possibile compilare, nell'ambito oggetto di *stage*, con l'*automation server* Jenkins come descritto nei documenti di sviluppo sicuro.

Fase 6: dal 28/10/2024 al 31/10/2024 e dal 04/11/2024 al 08/11/2024 (72h)

Viene approfondito l'argomento dei *test* automatici tramite lo strumento Jenkins e, a tale fine, lo studente inserisce dei *test* di esempio verificando di poterli eseguire e ne visiona i *report*.
Similmente viene verificata la possibilità di effettuare *deploy* tramite Jenkins.

Fase 7: dal 11/11/2024 al 15/11/2024 (40h)

Lo studente fornisce i suoi *feedback* finali su quanto appreso tramite l'*automation server* e sponsorizza la sua esperienza ai *team* realizzando documentazione e presentazione.

Fase 8: dal 18/11/2024 al 20/11/2024 (24h)

L'esperienza fatta dallo studente viene analizzata al fine di discutere i vantaggi nell'uso di un sistema di *build* e *deploy* automatizzato. Ogni esperienza fatta durate lo *stage* è corredata da documentazione, versionamento di configurazioni o *software* se utilizzati. Viene eseguita una presentazione finale.

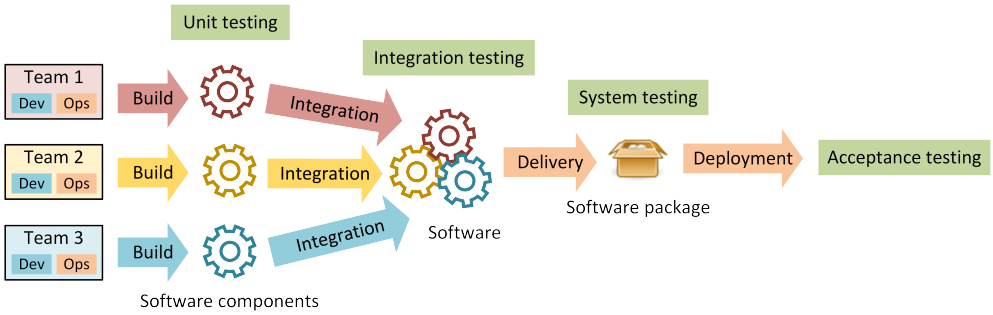


Figura 2.3: Flusso di Build, Delivery e Deployment in ambito DevOps.
Fonte: https://commons.wikimedia.org/wiki/File:DevOps_from_Integration_to_Deployment.png.

Tali fasi hanno rispettato quanto inizialmente pianificato con la differenza che è stata data maggiore importanza all'applicazione delle fasi di DevOps in ambienti Office365 e allo sviluppo con le tecnologie Power Automate e Power Apps.
Inizialmente era incluso fortemente l'ambiente proprietario Sistemi nelle fasi del progetto poiché era prevista una maggiore collaborazione con lo stage Applicativi di DevOps in ambito Sistemi.
Questi riallineamenti sono avvenuti in maniera naturale e coerentemente con le necessità e indicazioni fornitemi dal *tutor* aziendale.

Interazione con il tutor aziendale

Durante l'avanzamento dei lavori, le attività da me svolte sono sempre state dichiarate tramite lo strumento Planner, inoltre tutti i documenti da me realizzati sono stati condivisi in un ambiente comune al fine di mantenere chiarezza con il *team* e il *tutor* aziendale.
Con quest'ultimo ho avuto uno stretto contatto durante il periodo di *stage* in modo da favorire l'interazione e garantire il raggiungimento degli obiettivi prefissati.
Ad ogni significativo risultato ottenuto ho realizzato la relativa documentazione e una presentazione successivamente esposta al *tutor* e al *team* di sviluppo.

2.3 Obiettivi personali

Durante l'evento STAGE-IT avvenuto il giorno 8 aprile 2024, ho potuto dialogare con diverse aziende relativamente ai loro progetti di *stage* offerti. Sono seguitati colloqui nelle rispettive sedi e infine ho convenuto che l'azienda che mi suscitava maggiore

interesse fosse Wintech.
Questo è conseguenza dell'organizzazione aziendale in linea con i processi studiati durante il corso di studi, l'attenzione che è stata posta fin da subito agli stagisti e all'innovazione e grazie ai progetti di *stage* relativi a tematiche di mio interesse.
È per tali motivazioni che i miei obiettivi personali sono stati:

- Introduzione ad un contesto lavorativo, legato al percorso di studi da me frequentato, che comprendesse i principali processi aziendali studiati: il soddisfacimento è decretato dal completamento dello *stage* con soddisfazione del *tutor* aziendale e dallo studio dei processi aziendali.
- Collaborazione con un *team* di sviluppo professionale utilizzando gli strumenti adeguati: il soddisfacimento è decretato dalla corretta adozione delle tecnologie di collaborazione e sviluppo aziendali e dall'ottenimento di almeno un avanzamento effettivo realizzato lavorando con il *team*.
- Sviluppo, analisi ed integrazione con il contesto aziendale di tecnologie per me nuove. Creazione della conseguente adeguata documentazione: il soddisfacimento è decretato dalla conferma del *tutor* aziendale e del *team* che i documenti da me realizzati e le soluzioni da me sviluppate, utilizzando le tecnologie oggetto di *stage*, siano qualitativamente sufficienti per il contesto lavorativo.
- Realizzazione di flussi di automazione in grado di migliorare la produttività e la qualità dei processi aziendali e di aggiungere importanti funzionalità ad applicazioni e prodotti reali: il soddisfacimento è decretato dall'effettiva applicazione in azienda dei flussi di automazione da me individuati e definiti.
- Gestione delle risorse e del tempo fornitomi per organizzare e realizzare in autonomia i *task* definiti in fase di pianificazione: il soddisfacimento è decretato dal completamento di tutte le attività e gli obiettivi dello *stage* nei tempi prestabiliti.

È stato nel mio interesse entrare in un contesto lavorativo al fine di crescere professionalmente e acquisire maggiori opportunità in vista del periodo successivo al conseguimento della laurea.

Capitolo 3

Svolgimento *stage*

In questo capitolo vengono descritte tutte le attività da me svolte durante lo *stage*, divise nelle sezioni "Analisi", "Progettazione" e "Sviluppo software e documentale", in modo da fornire una panoramica chiara e strutturata del lavoro svolto, evidenziando il processo seguito e le competenze acquisite in ciascuna attività.

3.1 Analisi

3.1.1 Requisiti

I requisiti del mio *stage*, derivanti in parte dalle dichiarazioni aziendali presenti nel documento "Progetto Formativo" generato all'inizio del suo svolgimento, e in parte conseguenza dell'analisi dei requisiti avvenuta in corrispondenza con la presenza di specifiche necessità progettuali, sono divisi in categorie:

O - requisiti obbligatori, vincolanti in quanto obiettivi primari richiesti dall'azienda.

D - requisiti desiderabili, non strettamente necessari ma dal riconoscibile valore aggiunto.

F - requisiti facoltativi / opzionali, rappresentanti un valore aggiunto non strettamente competitivo.

Tabella 3.1: Tabella dei requisiti progettuali.

Codice	Descrizione
O1	Comprensione e mappatura delle funzionalità possibili tramite l'adozione dell'applicazione Microsoft Power Automate.
O1.1	Analisi approfondita riguardo ai flussi Power Automate ed individuazione delle caratteristiche, positive e negative, della sua adozione in azienda. Redazione di un documento dedicato.
O1.2	Sviluppo di almeno un flusso Power Automate.
O1.3	Produzione ed esposizione al *tutor* aziendale e al *team* di sviluppo di una presentazione riguardo ai risultati della mia analisi sull'adozione di Power Automate in azienda.

Codice	Descrizione
O2	Comprensione ed utilizzo della tecnologia Microsoft Power Apps e relative fasi di sviluppo collaborativo.
O2.1	Comprensione di Power Apps mediante lo svolgimento di attività di affiancamento con il *team* di sviluppo fino al raggiungimento della comprensione dei metodi lavorativi e dei prodotti aziendali sviluppati con tale strumento.
O2.2	Identificazione e utilizzo di un metodo adatto allo sviluppo collaborativo di applicazioni Power Apps.
O2.3	Ottenimento di avanzamenti nello stato dei lavori di applicazioni aziendali realizzate con Power Apps.
O3	Individuazione ed implementazione di un efficace sistema di versionamento per progetti realizzati utilizzando tecnologie Power Automate e Power Apps
O4	Comprensione delle metodologie DevOps e realizzazione della documentazione relativa alla sua applicabilità su tecnologie Power Automate e Power Apps.
D1	Collaborazione con gli altri stagisti universitari e individuazione di soluzioni collaborative relative ai bisogni progettuali comuni.
D2	Realizzazione di PoC a supporto delle soluzioni individuate durante le fasi di ricerca.
D2.1	Realizzazione di PoC riguardo allo sviluppo di flussi Power Automate approvativi.
D2.2	Realizzazione di PoC riguardo all'integrazione tra flussi Power Automate ed elementi esterni tramite le chiamate HTTP.
D2.3	Realizzazione di PoC riguardo al processo DevOps di *build* su progetti realizzati con tecnologie Power Automate/Power Apps.
D2.4	Realizzazione di PoC riguardo al processo DevOps di *test* su progetti realizzati con tecnologie Power Automate/Power Apps.
D3	Esecuzione di *meeting* e scambio di documenti con gli altri stagisti universitari al fine di comprendere l'applicabilità delle fasi di DevOps in ambito Sistemi
F1	Esplorazione della tecnologia Angular mediante realizzazione di un progetto di *test*
F2	Integrazione di un progetto Angular con le fasi di DevOps mediante un *server* Jenkins.
F3	Realizzazione ed esposizione di una presentazione finale, in collaborazione con gli altri stagisti, riguardo tutto il lavoro fatto durante lo *stage*.

3.1.2 Ambiente di lavoro

Durante i primi giorni dello *stage*, il *tutor* aziendale mi ha introdotto all'ambiente di lavoro e alle tecnologie di comunicazione e collaborazione utilizzate.
Conseguentemente ho analizzato e utilizzato il sistema di messaggistica basato su Microsoft Teams e Outlook e ho compreso la struttura di condivisione dei dati, utilizzata in azienda, basata sullo strumento Microsoft SharePoint. Esso è una piattaforma *software* in grado di organizzare dati sottoforma di *file* e strutture tabellari chiamate "Liste", al fine di gestire il materiale condiviso dall'azienda e dai singoli *team* tramite un sistema di accessi e autorizzazioni.

Tramite questi strumenti ho studiato i documenti aziendali a me forniti in modo da comprendere i principali processi produttivi di Wintech.
Tali documenti comprendono i "Documenti di sviluppo sicuro" e includono:

- Agile e SCRUM: descrizione delle metodologie Agile e SCRUM, spiegazione dei ruoli necessari e delle cerimonie previste.
- Presentazione sviluppo sicuro: presentazione PowerPoint che descrive i processi aziendali atti a migliorare la qualità dei prodotti realizzati automatizzando fasi ripetitive e rispettando criteri di sicurezza.
- Modelli di sviluppo sicuro: elenco dettagliato dei documenti di sviluppo sicuro i quali descrivono come applicare automazioni processuali (per esempio i processi di *build* e *deploy*) in modo sicuro e normato.
- Politiche di sviluppo sicuro: strategie e normative aziendali definite al fine di garantire sicurezza e qualità nei processi e nel ciclo di vita del *software*.
- Piani di progetto degli altri stagisti: piani formativi degli altri due stagisti che nel mio stesso periodo hanno effettuato lo *stage* universitario in Wintech.

Relativamente a quest'ultimo punto, nei primi giorni ho approfondito il lavoro svolto dagli altri stagisti tramite appositi *meeting* nei quali mi hanno descritto i risultati ottenuti fino a quel momento. Essi, avendo iniziato lo svolgimento del progetto circa una settimana prima, mi hanno esposto, mediante apposite presentazioni PowerPoint, le proprie ricerche riguardanti l'utilizzo dello strumento Git e dello studio avvenuto riguardo la possibilità di integrare tra loro gli strumenti Planner e Taiga.

3.1.3 Tecnologie oggetto di *stage*

Questa sottosezione descirve i metodi di apprendimento e le nozioni apprese durante la fase di analisi delle tecnologie su cui si basa la ricerca del mio progetto di *stage*.
Dopo aver compreso le tecnologie e i principali processi aziendali, ho partecipato ad un *meeting* con il *tutor* aziendale al fine di discutere il mio progetto di *stage*.
Gli obiettivi scaturiti da tale incontro sono stati:

- Autoapprendimento dello strumento Power Automate.
- Realizzazione di un PoC che testasse la possibilità di realizzare un flusso approvativo Power Automate.
- Testare le funzionalità disponibili con le licenze di utilizzo *standard*.

Power Automate

Ho pertanto studiato approfonditamente tali tecnologie con l'ausilio delle numerose guide e *tutorial* offerti da Microsoft. Essi sono direttamente accessibili dalla *home* di Power Automate e Power Apps e sono divisi in moduli testuali corredati da immagini, dalla durata e argomenti specifici.
Lo studio di tali moduli mi ha permesso di comprendere il funzionamento e le *features* principali di Power Automate. Esso è formato da una pagina *web* che offre controllo sui flussi di automazione creati: è possibile modificarli e visualizzarne i dettagli, le esecuzioni e le statistiche. Inoltre, si possono creare dei nuovi flussi partendo da altri progetti pubblici, o a noi condivisi, e modelli offerti da Microsoft da adattare alle

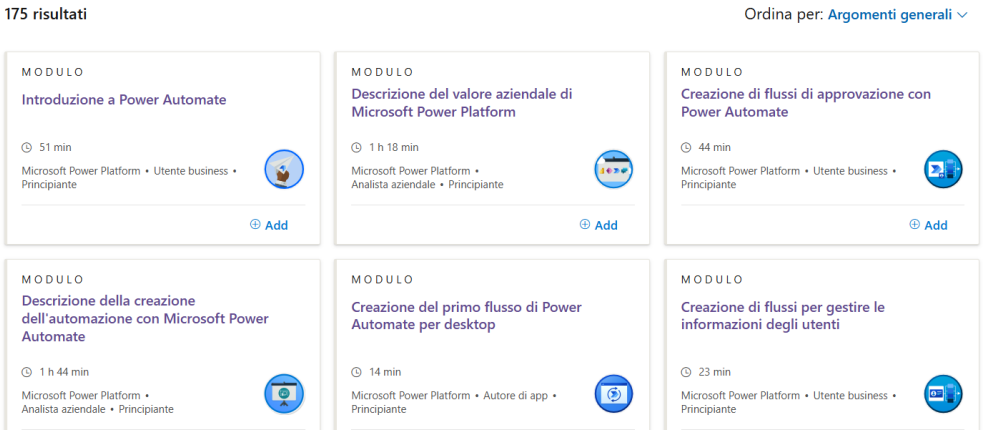


Figura 3.1: Moduli formativi per Power Automate.
Fonte: https://learn.microsoft.com/it-it/training/browse/?products=power-automate.

proprie esigenze.
La modifica di un flusso non avviene mediante la scrittura di codice tramite un linguaggio di programmazione, bensì tramite una composizione "a blocchi" personalizzabili collegati tra loro ciascuno avente proprietà e attributi definiti.
Essi sono selezionabili da una lista di blocchi relativi ciascuno a una funzionalità specifica di un servizio Microsoft. Tali blocchi possono essere "Trigger" o "Azioni" ed entrambi sono necessari per la creazione e il funzionamento di un flusso. In ogni flusso è presente uno e un solo Trigger il quale rappresenta il suo punto di partenza nonché la condizione che scaturisce la sua esecuzione.
Esistono tre tipologie principali di Trigger le quali determinano la tipologia stessa di ogni flusso:

- Automatico: per esempio "SharePoint - Quando viene creato un elemento".
- Istantaneo: per esempio "Attiva manualmente un flusso".
- Pianificato: per esempio "Ricorrenza", il quale attiva il flusso periodicamente.

Ad ogni Trigger possono essere collegate, in serie o in parallelo, una moltitudine di Azioni, ciascuna responsabile di uno specifico compito, per esempio sono presenti le azioni "Inizializza variabile", "Avvia e attendi un'approvazione", "Teams - Crea una chat" e "Outlook - Invia un messaggio di posta elettronica".
Sono inoltre presenti azioni dedicate alla gestione logica dei flussi come "Condizione" e "Do until". La prima rappresenta la struttura di controllo rappresentata nei classici linguaggi di programmazione con "if", responsabile della ramificazione dell'esecuzione del flusso in base a una condizione specifica. La seconda rappresenta la struttura di controllo rappresentata nei classici linguaggi di programmazione con "Do while", responsabile della ripetizione condizionata di un insieme di azioni garantendone sempre la prima esecuzione.
Successivamente ho individuato, durante un meeting con il *tutor* aziendale, la necessità di integrare i flussi Power Automate con il *software* gestionale WOW e gli altri prodotti aziendali, al fine di poter integrare le funzionalità desiderate con libertà mantenendo

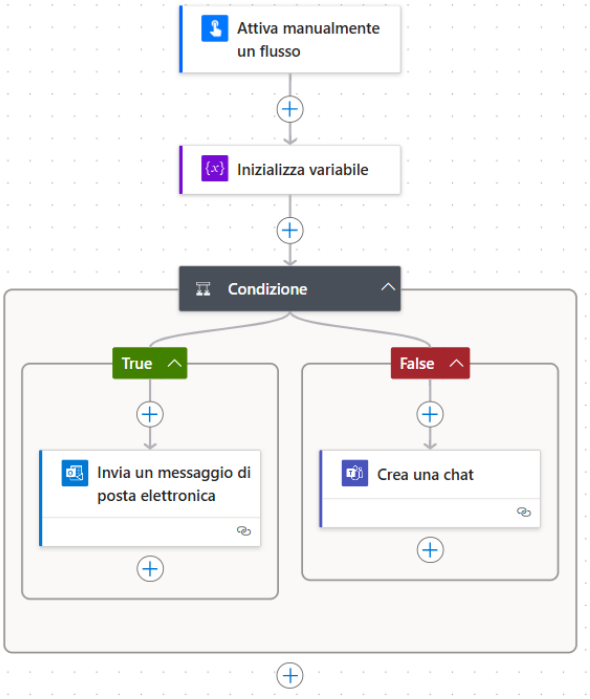


Figura 3.2: Esempio di un flusso Power Automate istantaneo.

coordinate le diverse parti del prodotto.
Sono emersi quindi due requisiti: il primo, utile per apprendere le tecnologie in oggetto, è relativo alla creazione di un flusso che automatizzi un processo approvativo (requisito D2.1).
Il secondo (requisito D2.2), più concreto e integrabile con i prodotti aziendali, si focalizza sull'applicazione ai flussi Power Automate di chiamate HTTP(Hypertext Transfer Protocol): in italiano "protocollo di trasferimento ipertestuale", è un protocollo di rete, ovvero un insieme di regole formalmente descritte che definiscono le modalità di comunicazione tra due o più apparecchiature elettroniche, usato come principale sistema per la trasmissione di informazioni sul *web*.

Power Apps

I flussi Power Automate portano grande vantaggio in termini di efficienza dei processi aziendali e possono aumentare il numero di funzionalità di un'applicazione.
Per questo motivo il loro utilizzo assume maggiore significato quando associato alla realizzazione di applicazioni Power Apps, le quali sono pensate per integrare nativamente i flussi creati.
Il metodo di apprendimento con cui ho analizzato e studiato Power Apps deriva non solo da attività individuali tramite i moduli didattici offerti da Microsoft nell'apposito sito *web*, simili a quelli per Power Automate, ma soprattutto grazie alla collaborazione con un membro del *team* di sviluppo che ho affiancato.
Egli è il responsabile in azienda di tutti i loro prodotti realizzati utilizzando le tecnologie oggetto del mio *stage*, e assieme le abbiamo attentamente analizzate e modificate al fine di migliorarle ed applicarci le soluzioni da me individuate relativamente alle pratiche

DevOps richieste.
Power Apps offre la possibilità di creare nuove applicazioni e visionare quelle già create, o a noi condivise, con la possibilità di apportare modifiche.
È inoltre possibile visualizzare informazioni e statistiche relative a tutte le applicazioni in nostro possesso.
Il loro sviluppo non avviene esclusivamente mediante codice di programmazione bensì dall'utilizzo di componenti, *standard* o *custom*, che vengono manualmente posizionati sull'interfaccia, creando in questo modo una struttura gerarchica ad albero chiamata "Tree View".

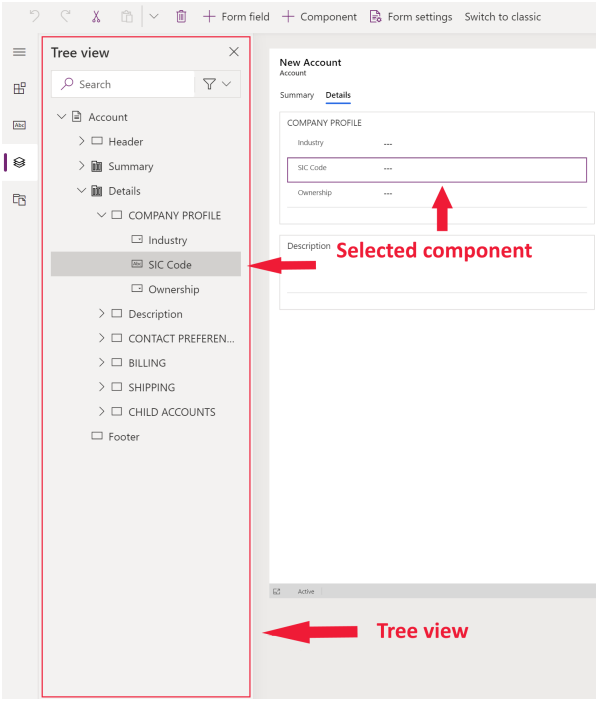


Figura 3.3: Elemento Tree View per la gestione gerarchica dei componenti.
Fonte: https://learn.microsoft.com/en-us/power-apps/maker/model-driven-apps/using-tree-view-on-form.

Al fine di modificare i parametri che lo compongono, ciascun componente può essere personalizzato tramite un'apposita finestra dell'interfaccia grafica.
Questo non comprende solo le peculiarità grafiche ma soprattutto il loro comportamento a seguito di azioni specifiche: per esempio è possibile creare un componente "Pulsante" e personalizzarne il campo "OnSelect" al fine di eseguire conseguentemente un flusso Power Automate e inviarne i dati di *output* ad un altro componente.
Tali istruzioni vengono definite utilizzando Microsoft Power Fx, il quale rappresenta un linguaggio di programmazione dichiarativo, fortemente tipizzato e con uso limitato di codice, usato in Microsoft Power Platform.
Quest'ultima piattaforma include un insieme di strumenti e tecnologie, compresi Power Automate e Power Apps, pensati per offrire la possibilità di sviluppare prodotti *software* con limitata necessità di scrivere, e conoscere, codice tramite linguaggi di programmazione.

3.1.4 Tecnologie di supporto

Jenkins

Al fine di applicare le automazioni necessarie per adottare al *software* le metodologie DevOps, come richiesto dagli obiettivi di *stage*, ho individuato lo strumento Jenkins in quanto esso rispecchia perfettamente le mie necessità ed è inoltre già conosciuto e applicato dal resto del *team* di sviluppo, rappresentando quindi una tecnologia consolidata che non necessita di ulteriore formazione all'interno dell'azienda.
Ho appreso le conoscenze necessarie per studiare e utilizzare tale strumento in totale autonomia mediante la consultazione delle guide e del numeroso materiale presente *online* comprese le pagine *web* ufficiali di Jenkins con gli annessi video *tutorial*.
Esso è uno strumento scritto in linguaggio di programmazione Java che viene eseguito all'interno di un *web server* e offre la possibilità di creare, gestire, eseguire e monitorare dei progetti Jenkins chiamati "Job", potendone visionare e registrare i *report* di *output*. Essi sono unità di lavoro configurabili che rappresentano un'attività o un progetto specifico che Jenkins esegue. Inoltre possono essere di diverso tipo ma tutti sono basati sull'esecuzione di uno *script* o una *pipeline* di automazione.
Una *pipeline* Jenkins è una sequenza di fasi chiamate "Stage", eseguite in serie o in parallelo, responsabili di specifiche operazioni definite dall'utente, per esempio *build*, *test*, *deploy*.
Esse possono essere definite tramite il linguaggio di programmazione Groovy oppure tramite l'interfaccia grafica di Jenkins.
Questo strumento permette dunque di fornire servizi di integrazione continua ed è indicato per l'applicazione di alcune fasi di DevOps come Build e Test, il caricamento automatico dei *file* sul *repository* di versionamento desiderato e l'autenticazione automatica ai servizi Microsoft tramite comandi Power Platform CLI.
Quest'ultima è un'interfaccia a riga di comando che offre la possibiltà di eseguire un insieme di comandi specifici atti all'autenticazione e alla gestione degli ambienti e delle Microsoft Solutions.

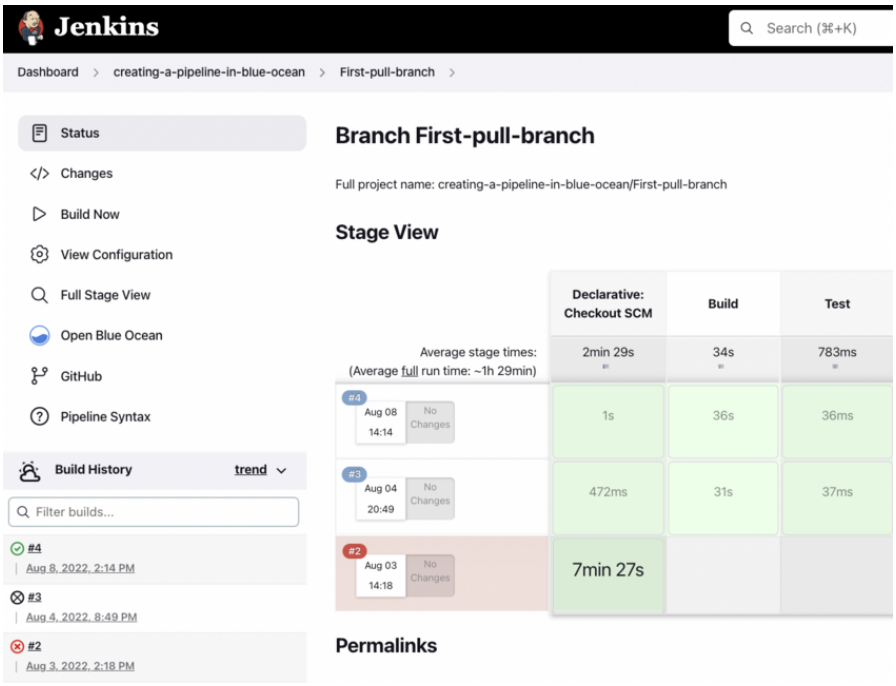


Figura 3.4: Esempio interfaccia di Jenkins.
Fonte: https://commons.wikimedia.org/wiki/File:Updated-jenkins-view.png.

Microsoft Solutions

Come ulteriore oggetto di analisi e studio, ho affrontato l'utilizzo e la comprensione delle Microsoft Solutions. Esse fanno parte della *suite* Microsoft Power Platform e sono degli strumenti che servono a contenere tutti i flussi Power Automate e applicazioni Power Apps corrispondenti a uno stesso progetto, in modo da poterli sviluppare e distribuire in maniera organizzata e centralizzata.
Esistono due tipi di Soluzioni:

- Gestite: destinate prevalentemente per gli ambienti di produzione e *test*, non permettono la modifica degli elementi al suo interno. Per generare una Soluzione gestita è necessario esportare una soluzione non gestita.
- Non gestite: destinate prevalentemente per gli ambienti di sviluppo, permettono la modifica degli elementi al suo interno e la sua esportazione.

A differenza delle tecnologie precedentemente descritte, le Microsoft Solution non erano materia già affrontata in azienda, per cui il loro apprendimento è avvenuto in maniera totalmente autonoma mediante la documentazione presente nei siti *web* Microsoft dedicati.
Le Soluzioni preservano i loro dati, e quelli degli elementi in esse contenuti, nel sistema di archiviazione Microsoft Dataverse che funge da *database* centralizzato per applicazioni sviluppate con la Power Platform.
Tale adozione permette di usufruire di funzionalità altrimenti assenti. Le conseguenze del loro utilizzo e le motivazioni che mi hanno portato ad utilizzare questa tecnolo-

gia sono materia di discussione presente nella sezione Adozione delle Microsoft Solutions.

3.1.5 DevOps

Prima di poter progettare una soluzione applicativa al fine di applicare le fasi di DevOps a progetti realizzati con Power Automate e Power Apps, ho analizzato e compreso quali fossero le implicazioni di ogni *step*.
Ho approfondito tale pratica soprattutto studiando gli approfondimenti presenti sul sito *web* di Atlassian, nota compagnia di *software enterprise*, e dalla teoria sul ciclo di vita del *software* e sulle metodologie di automazione dei processi aziendali appresa durante il percorso di studi universitario.
Ho inoltre studiato i documenti di sviluppo sicuro aziendali, nei quali sono presenti delle linee guida su come adottare il DevOps in Wintech.
Le sue fasi sono:

Plan

Essendo la fase iniziale del ciclo, ha lo scopo di definire i requisiti, pianificare il ciclo di vita del progetto e identificarne le metodologie di lavoro. Adottando una metodologa iterativa, è consono che la pianificazione e le conseguenti fasi successive, possano subire modifiche rispetto alle precedenti pianificazioni.

Code

Dopo aver compreso cosa e come soddisfare i requisiti progettuali, vengono sviluppate le soluzioni individuate seguendo le metodologie e le norme definite in fase Plan.
Essendo il progetto di *stage* basato su tecnologie non sviluppabili tramite strumenti tradizionali, come IDE non proprietari e linguaggi di programmazione consolidati, l'analisi della fase Code è stata necessaria per comprendere le funzionalità disponibili agli utenti al fine di sviluppare tali prodotti.
È compresa in questa fase anche la ricerca avvenuta al fine di individuare le metodologie e gli strumenti adatti a sviluppare progetti con le tecnologie in oggetto in maniera collaborativa e adottando un efficace metodo di versionamento.
Questo ha necessitato di uno studio sullo strumento Git, già ampiamente affrontato durante il percorso di studi universitari.
Infine, è compresa nella fase Code anche l'attività di analisi statica del codice, la quale supporta le fasi di sviluppo identificando e segnalando eventuali errori nel codice sorgente senza la necessità di eseguirlo.

Build

Fase relativa alla creazione dei pacchetti e dei *file* eseguibili costruiti partendo dal codice sorgente prodotto nella fase precedente.
Il risultato di questa operazione rappresenta il prodotto sul quale verranno effettuati i *test* e le verifiche necessarie.

Test

Ottenuto l'artefatto generato dalla fase Build, esso può essere testato mediante l'utilizzo di svariati strumenti appositi.
Essi comprendono diversi tipi di *test* dinamici:

- Test di unità: verifica che le singole unità di codice, per esempio funzioni, metodi o classi, funzionino come previsto.
- Test di integrazione: verifica che più parti di codice o servizi interagiscano tra loro correttamente.
- Test di carico: verifica il comportamento del sistema sotto carico normale ed elevato.
- Test di sicurezza: individua le eventuali vulnerabilità o minacce di sicurezza presenti nel sistema.

Durante le fasi analitiche del mio *stage*, ha avuto molta rilevanza lo studio sui metodi nativi ed esterni applicabili ai flussi Power Automate e alle applicazioni Power Apps al fine di applicare *test*.

Release

Quando il prodotto è stato creato e testato, esso è considerato pronto per essere reso disponibile ai clienti, e viene quindi rilasciato identificandone una versione specifica.

Deploy

Una versione rilasciata del prodotto viene distribuita nello specifico ambiente di produzione in cui gli utenti finali lo utilizzeranno, garantendone il corretto funzionamento.

Operate

La gestione e il funzionamento dell'applicazione in produzione viene garantita tramite attività come l'amministrazione e la manutenzione dei *server* e la risoluzione dei problemi.

Monitor

Al fine di garantire affidabilità e buone prestazioni del sistema distribuito, il suo comportamento viene osservato tramite strumenti che ne raccolgono e analizzano i dati. Vengono inoltre raccolti i *feedback* degli utenti finali e dei clienti al fine di migliorare la qualità del prodotto.

3.1.6 Analisi delle applicazioni aziendali

Il *focus* principale dello *stage* è compiere un'analisi sulle tecnologie al fine di raggiungere gli obiettivi riguardanti la ricerca sull'effettiva applicabilità delle pratiche di DevOps ai progetti realizzati con Power Automate e Power Apps. Con il *tutor* aziendale abbiamo però fin da subito definito che avrei alternato tali attività con fasi di sviluppo su applicazioni aziendali al fine di approfondire la conoscenza delle tecnologie in oggetto e di diventare operativo in modo più efficace.

Insieme al membro del *team* di sviluppo responsabile per la realizzazione delle applicazioni Power Apps e dei flussi Power Automate aziendali, ho affrontato problemi di sviluppo di vario genere in funzione delle necessità del momento, potendo analizzare diversi prodotti aziendali.
Le effettive attività di programmazione svolte su queste applicazioni sono presenti nella sezione Sviluppo di applicazioni aziendali.
I principali prodotti su cui abbiamo collaborativamente lavorato sono:

Report di visita

Principale applicazione su cui ho lavorato e con cui ho svolto attività di analisi al fine di comprendere le tecnologie utilizzate. Essa ha l'obiettivo di offrire la possibilità ad un dipendente dell'azienda, il quale dovesse trovarsi nella sede di un cliente, di registrare e condividere con i colleghi, in un ambiente condiviso, un *report* della visita al fine di tracciare l'esperienza con il cliente.
Essa offre quindi la possibilità di:

- Ricercare e selezionare una specifica azienda visualizzabile in un'apposita lista a schermo
- Visualizzare le sedi e le filiali dell'azienda selezionata
- Visualizzare le informazioni anagrafiche dei contatti dell'azienda selezionata
- Visualizzare i *report* di visita già creati come bozza
- Modificare eventuali bozze selezionate
- Creare un nuovo documento inserendo in un apposito *form* i dati relativi all'operatore, al cliente e alla visita specifica
- Salvare tale *report* come bozza o come documento definitivo
- Caricare automaticamente i dati generati nel *database* aziendale

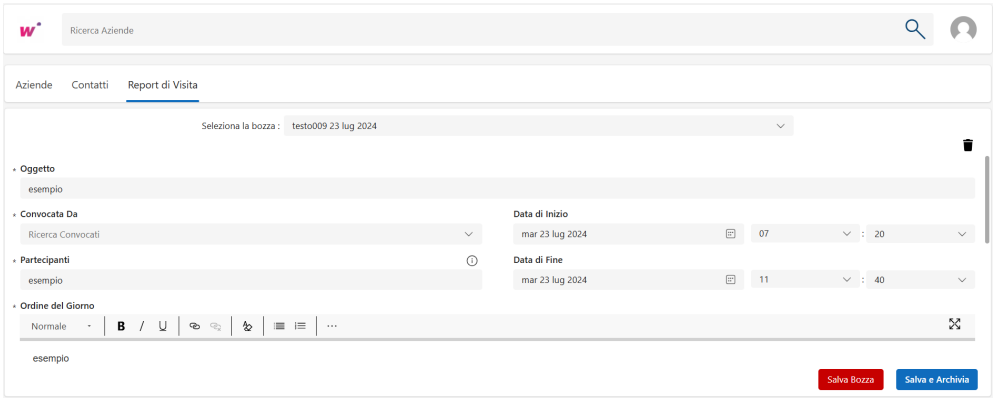


Figura 3.5: Interfaccia grafica di Report di visita.

Registro accessi

Applicazione realizzata al fine di gestire agevolmente i *check-in* e *check-out* lavorativi del personale.
Le funzionalità presenti sono:

- Identificazione del dipendente
- Selezione della sede, compresa la possibilità di selezionare lo *smart working* o lavoro fuorisede
- Selezione del *check-in*
- Visualizzazione di errore in caso di *check-in* in mancanza del precedente *check-out*
- Selezione del *check-out*
- Visualizzazione di errore in caso di *check-out* in mancanza del precedente *check-in*
- Visualizzazione di tutti i propri *check-in* e *check-out*

Non conformità

Nel momento in cui ho svolto lo *stage*, questa applicazione era nelle sue prime fasi di sviluppo e le sue funzionalità non erano state ancora definite pienamente.
I suoi obiettivi sono quelli di creare e gestire un flusso approvativo a più *step*, dove per ogni fase vengono notificati i corrispondenti incaricati, ai quali viene richiesta la compilazione di *form* relativi alla non conformità di un prodotto e alle conseguenti azioni da intraprendere.

3.1.7 Angular

Gli ultimi requisiti emersi durante lo svolgimento dello *stage* sono quelli relativi allo studio e all'applicazione di alcune fasi di DevOps a progetti realizzati con lo strumento Angular.
Esso è un *framework* per la realizzazione di applicazioni *web* tramite il linguaggio di programmazione TypeScript e rappresenta una delle principali tecnologie usate dai *team* di sviluppo di Wintech.
È per questo motivo che, nel momento in cui ho terminato le attività previste per il mio *stage* con alcuni giorni di anticipo, mi è stato affidato il compito di studiare e provare ad utilizzare le stesse loro tecnologie e metodologie di lavoro al fine di comprenderle al meglio ed integrarmi maggiormente nell'ambiente lavorativo.
Tramite il sito *web* ufficiale di Angular, ho seguito le guide fornite in modo da apprenderne le basi e poter realizzare quanto richiesto soddisfacendo i requisiti assegnati.

3.2 Progettazione

3.2.1 *Scope* di Power Automate e Power Apps

A seguito dell'analisi avvenuta sulle tecnologie oggetto di *stage*, ho identificato il compito che Power Automate e Power Apps devono avere all'interno di un progetto. Esse sono infatti tecnologie pensate per sviluppare in maniera rapida soluzioni

relativamente semplici. Nel momento in cui ci fosse la necessità di applicare logiche molto complesse o l'integrazione con servizi esterni dalla *suite* Microsoft, tali strumenti non sono più ideali in quanto si incomberebbe in numerosi vincoli tecnologici. Questo è un fattore importante da considerare nelle fasi di progettazione di un progetto.

3.2.2 Individuazione delle soluzioni tecnologiche

In questa sottosezione vengono esplicitate le motivazioni che mi hanno portato a identificare determinate soluzioni al fine di soddisfare i requisiti compresi in fase di analisi.

Adozione delle Microsoft Solutions

L'adozione delle Soluzioni, essendo esse basate sul sistema di archiviazione Dataverse, comporta la possibilità aggiuntiva di poter visionare la "Version history" dei flussi Power Automate. Tale funzionalità è in grado di ripristinare specifiche versioni precedenti dei flussi, agevolando le fasi di sviluppo.

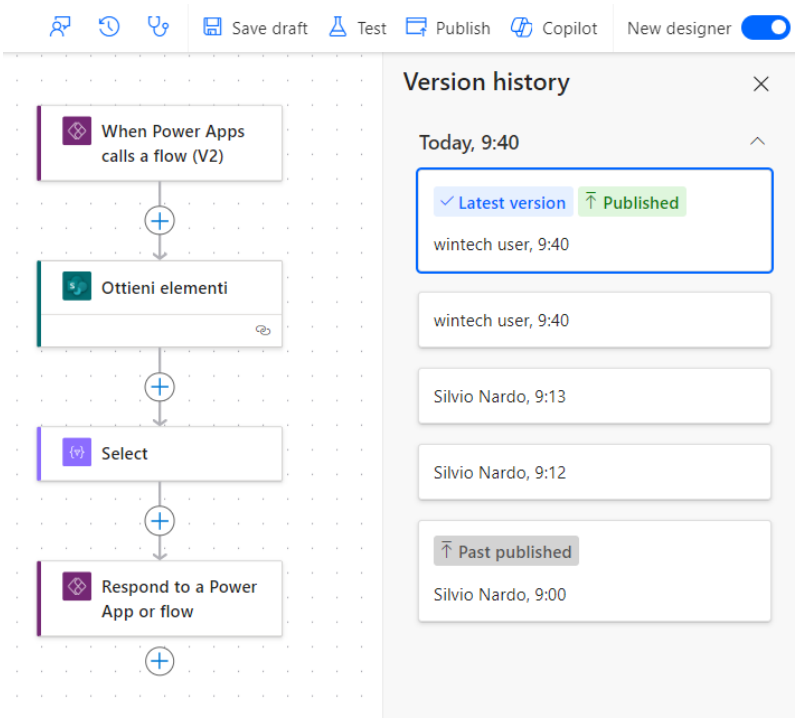


Figura 3.6: Funzionalità Version history nei flussi Power Automate.

Inoltre, essendo i componenti di una Soluzione contenuti in un unico pacchetto esportabile, la gestione dei progetti così organizzati risulta più agevole, soprattutto nelle fasi Build, Code e Deploy.

Job Jenkins

Riguardo alla scelta del tipo di Job da adottare in Jenkins, ho individuato il "Multi-branch Pipeline Job". Esso permette di collegare un *repository* Git e ha la possibilità di gestire automaticamente una *pipeline* indipendente, contenuta in un apposito *file* chiamato "Jenkinsfile", per ogni *branch*. Tale Job ha inoltre la possibilità di eseguire automaticamente i Jenkinsfile corrispondenti solo ai *branch* in cui è avvenuta una modifica, rendendo questo strumento ideale per la gestione di progetti Git.

3.2.3 Soluzioni individuate per le fasi DevOps

Plan

Come strumenti identificati per la pianificazione delle attività ho utilizzato Planner e Taiga. Essi permettono di categorizzare tutte le attività da assegnare al *team* di sviluppo tramite *tag* e contenitori specifici, i quali definiscono lo stato di avanzamento dei lavori.
Nello specifico, Planner viene utilizzato solo dal responsabile di progetto e serve a tenere traccia dei casi d'uso, mentre Taiga si occupa di tracciare i *task* più specifici assegnati ai membri del *team*, il quale si occupa di organizzarli.

Code

Il versionamento di tutte le parti dei progetti realizzati con Power Apps e Power Automate è avvenuto in un singolo *repository* Git in modo da garantire una migliore organizzazione dei dati e facilità di ripristino di specifiche versioni.
Per il lavoro collaborativo ho individuato la *feature* che integra il collegamento ad un *repository* Git con Power Apps e permette di fare *commit* e *push* automaticamente.
Il *repository* è diviso nei *branch* "Main", per le versioni funzionanti e stabili del prodotto, "Develop", per il prodotto ancora in fase di sviluppo, e diversi *feature branch* utilizzati dai singoli sviluppatori per apportare modifiche limitate a singole funzioni.
Per applicare a tali progetti l'analisi statica del codice, ho individuato come soluzione l'utilizzo degli strumenti nativi offerti da Microsoft:

- Verifica flusso: permette l'esecuzione di un controllo sul codice e la visualizzazione di errori sul flusso come i campi obbligatori mancanti, gli *input* non validi o i problemi legati alle licenze.
- Verifica app: permette l'esecuzione di un controllo sul codice e la visualizzazione di errori sull'applicazione come formule Fx non valide, problemi di accessibilità o problemi legati alle origini dati.
- Verifica soluzione: più dettagliata delle precedenti, permette l'esecuzione di un controllo sul codice e la visualizzazione di errori sui componenti della Soluzione.

Non ho utilizzato gli strumenti di analisi statica esterni poiché, a seguito di *test* esplorativi e attività di ricerca in merito, è emersa l'eccessiva difficoltà nell'integrazione di qualsiasi strumento non offerto da Microsoft, ai *file* generati dai flussi Power Apps e alle applicazioni Power Automate.

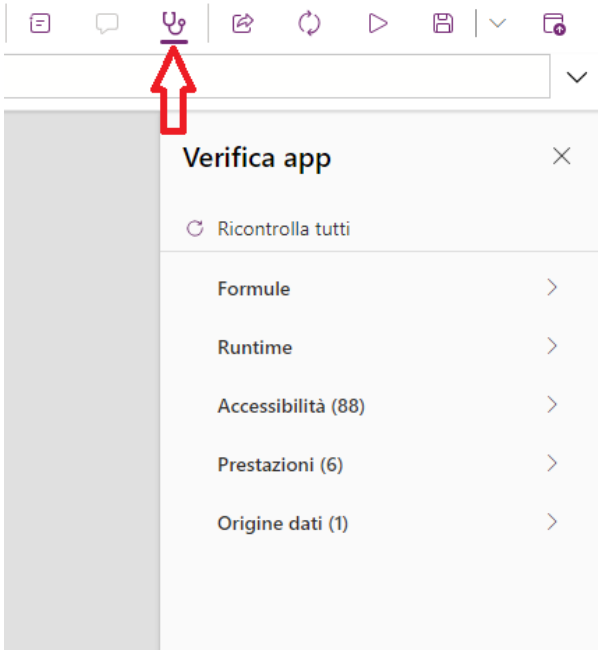


Figura 3.7: Funzionalità Verifica App nelle applicazioni Power Apps.

Build

I *file* generati da Power Automate e Power Apps al fine di rappresentare i flussi e le applicazioni create nelle rispettive interfacce di sviluppo, sono rappresentati da un pacchetto compresso ZIP contenente i *file* generati automaticamente da tali programmi. Essi sono principalmente di tipo JSON, e .msapp e possiedono una struttura di difficile comprensione, pertanto non sono pensati per uno sviluppo manuale diretto.
Le Microsoft Solution sono rappresentate dai *file* corrispondenti a quelli dei flussi e delle applicazioni in esse contenuti e nessuno di questi componenti possiede un *file* eseguibile.
Come metodo per l'esecuzione della fase Build automatica, ho individuato i comandi forniti dalla Power Platform CLI, automatizzati da un Job Jenkins, al fine di esportare i pacchetti relativi alle Soluzioni desiderate.

Test

A causa della natura dei pacchetti di dati generati con la fase Build è emerso, similmente all'analisi statica del codice, la necessità di utilizzare principalmente gli strumenti nativi forniti da Microsoft per eseguire *test* dinamici.
Per i flussi Power Automate è disponibile la funzione "Test" presente nell'interfaccia grafica di sviluppo.
Essa permette l'esecuzione del flusso e la visione dei suoi risultati compresi gli *input* e *output* di ogni azione e il loro tempo di esecuzione. Inoltre, in caso di errore, sono fornite le relative informazioni.

Figura 3.8: Funzionalità Test nei flussi Power Automate.

Per eseguire *test* esclusivamente su una specifica parte di un flusso Power Automate, simulando il comportamento delle azioni ad essa connesse, è disponibile la funzione "Static Result".
Essa permette un maggiore controllo sull'esecuzione del flusso durante i *test* e l'azzeramento dei tempi di esecuzione di azioni che non rappresentano il suo *focus*.
ho inoltre individuato la possibilità di utilizzare i Trigger e le Azioni Power Automate legate all'utilizzo delle chiamate HTTP, al fine di estrapolare i dati di *output* di un flusso ed elaborarli in uno *script* esterno realizzato con l'ambiente Node.js, il quale sfrutta il linguaggio JavaScript. Maggiori informazioni su quanto sviluppato per questa soluzione sono presenti nella parte Test della sottosezione Sviluppo DevOps.

Per quanto concerne i *test* su applicazioni Power Apps, sono presenti diversi strumenti nativi per verificare la loro corretta esecuzione.
"Test studio" è uno strumento che permette di eseguire l'applicazione e registrare tutte le azioni intraprese dall'utente (o definite manualmente), al fine di generare uno *script* di *test* che possa ripetere tali azioni e fornire un riscontro.
Attualmente l'utilizzo di Test studio non è conveniente a causa del numero considerevole di limitazioni: il suo funzionamento è limitato solo a componenti "classici", ovvero una lista di componenti presenti su Power Apps prima dell'arrivo dei componenti "moderni".
Esclude inoltre l'uso di componenti *custom* e non è compatibile con l'integrazione nativa tra Power Apps e Git.

Per questi motivi la metodologia identificata per permettere il *testing* ad applicazioni Power Apps è la funzionalità "Visualizza l'anteprima dell'app", con la quale è possibile eseguirla e verificarne il corretto funzionamento.
Nel caso in cui si verificassero degli errori durante tale operazione, i loro dettagli possono essere visualizzati.

Release

La strategia identificata per definire i rilasci è l'assegnazione di specifici *tag* alfanumerici ai *commit* nel *repository* Git.
Tale codice è composto dalla lettera "v" seguita da tre cifre separate da un carattere ".", per esempio "v1.0.0".
Il criterio di avanzamento delle cifre è:

- Avanzamento della prima cifra: sono avvenute modifiche architetturali che alterano significativamente le funzionalità previste ad alto livello.
  Esempio: sostituzione di una tecnologia fondamentale per il funzionamento di un programma o cambiamento radicale dei casi d'uso individuati.

- Avanzamento della seconda cifra: sono avvenute modifiche che alterano le funzionalità previste ad alto livello o ne aggiungono di nuove.
  Esempio: modifiche che soddisfano i requisiti di un caso d'uso non ancora risolto, aggiunta di classi in uno *script* che alterano il comportamento precedente del programma, aggiunta di una nuova scheda di un'applicazione.

- Avanzamento della terza cifra: sono avvenute modifiche che non alterano significativamente le funzionalità previste ad alto livello.
  Esempio: *Bug fix*, sostituzione di un componente di un'applicazione al fine di ottenere il medesimo risultato precedentemente atteso.

Deploy

Power Apps, Power Automate e le Micorsoft Solutions permettono non solo di esportare i pacchetti relativi ai propri progetti, ma anche di importarne di altri.
In tale funzione ho individuato il metodo per distribuire i prodotti negli ambienti di produzione.

Operate

L'operatività delle soluzioni implementate viene monitorata tramite il *feedback* fornito dal cliente.
Nel caso in cui si verificasse un problema con tale prodotto, viene fornita immediato supporto al fine di soddisfare nuovamente le attese.
Nel caso in cui il problema riscontrato fosse molto problematico, il sistema di versionamento definito permette il ripristino del prodotto ad una precedente versione stabile.

Monitor

Il monitoraggio dei *report* Jenkins, delle esecuzioni dei flussi Power Automate e delle esecuzioni delle applicazioni Power Apps, sono nativamente forniti dai rispettivi servizi. Tramite le interfacce disponibili è possibile vedere le informazioni sufficienti ad identificare eventuali problemi e "colli di bottiglia".

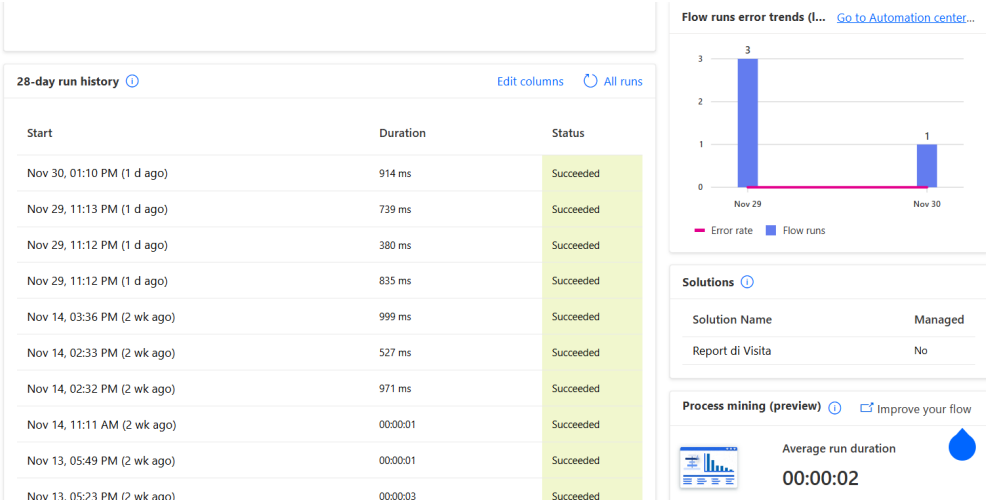


Figura 3.9: Vista parziale dell'interfacccia atta a monitorare un flusso.

3.2.4 Progettazione delle applicazioni aziendali

Le attività di progettazione che ho svolto, relative alle applicazioni aziendali, sono state limitate alla scelta delle strategie da esplorare per risolvere i problemi emersi. Questo perché tali applicazioni erano già presenti da tempo al momento del mio arrivo in azienda. Eccezione è fatta per l'applicazione "Non conformità" che, a differenza delle altre, era nelle sue prime fasi di sviluppo.
Al fine di comprenderne gli obiettivi ho preso parte a dei *meeting* dedicati con il tutor aziendale e la parte del *team* di sviluppo coinvolta.
Ho in seguito contribuito a progettare l'aspetto grafico e le soluzioni atte a soddisfarne i requisiti, ma il mio periodo di *stage* è terminato prima che io potessi iniziare il loro sviluppo.

3.3 Codifica e documentazione

In questa sezione sono presenti tutte le attività da me svolte al fine di sviluppare e implementare le soluzioni individuate in fase di progettazione.

3.3.1 PoC iniziali

Le prime fasi dello *stage* sono state autonome e a supporto della comprensione ed esplorazione delle tecnologie a me richieste. Avendo iniziato le mie attività con Power Automate, ho realizzato un primo flusso al fine di testare l'utilizzo di: Trigger, Azioni, collegamento con i servizi Microsoft OneDrive ed Excel, Azioni condizionali, cicli Do until, inizializzazione e modifica di variabili.
Tale flusso viene eseguito automaticamente quando viene creato un *file* dentro ad un *path* specifico su OneDrive e di conseguenza invia un'approvazione notificata sia tramite Teams che Outlook.
Dopo aver ricevuto la risposta, il flusso agisce di conseguenza spostando il *file* iniziale in un *path* specifico in funzione dell'esito dell'approvazione. Viene in aggiunta eseguito un controllo sui *file* già presenti nel percorso di destinazione e, nel caso fosse già presente

un documento omonimo, il documento da spostare viene rinominato.
Questo controllo avviene ciclicamente incrementando un valore posto nel nome del *file* fino al verificarsi dell'effettiva possibilità di eseguire lo spostamento.
Infine viene eseguito uno *script* Excel il quale scrive in una tabella di *log* le informazioni relative allo spostamento.

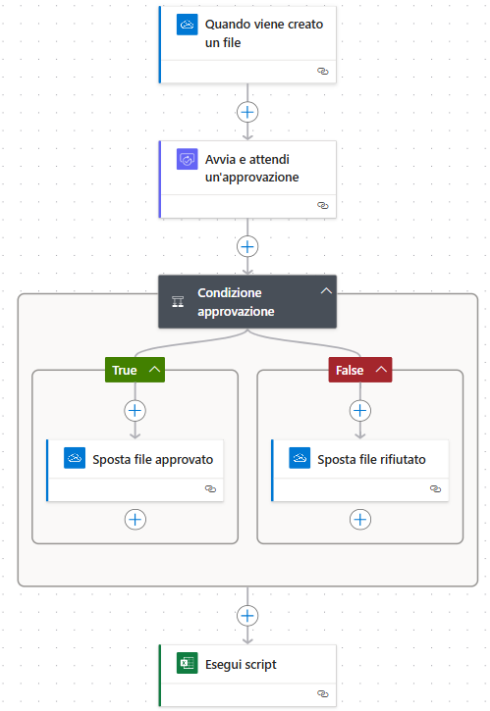


Figura 3.10: Flusso per testare condizioni, approvazioni ed esecuzione di *script*.

A fini illustrativi l'immagine mostra una versione semplificata del flusso descritto. Essa esclude il controllo della presenza di un *file* omonimo e quindi non include il ciclo Do until e l'inizializzazione e modifica di variabili.
Successivamente, ho prodotto una presentazione PowerPoint al fine di esporre al tutor aziendale le funzionalità, i lati positivi e i lati negativi delle tecnologie Power Automate e Power Apps, in funzione di quanto appreso durante la prima settimana di *stage*.

3.3.2 Sviluppo DevOps

Plan

Le attività pratiche legate alla fase di pianificazione sono la configurazione degli ambienti Planner e Taiga e la loro condivisione e utilizzo con i membri del *team* coinvolti.
Inoltre, in collaborazione con lo stagista responsabile del progetto Integrazione sistemi di pianificazione di progetto, abbiamo esplorato la possibilità di integrare i flussi Power Automate al fine di sostituire le logiche, realizzate tramite complessi *scirpt*, responsabili per l'interazione e la sincronizzazione tra le due piattaforme di pianificazione.
Per farlo abbiamo inoltre esplorato l'utilizzo dei *webhooks*, ovvero degli strumenti *web* che permettono di configurare l'invio automatico di chiamate HTTP di ritorno. Esse vengono eseguite da parte di un'applicazione *web* al fine di permettere dell'utilizzatore

di questo strumento di ricevere dati automaticamente al verificarsi di uno specifico evento.
A causa delle limitazioni incontrate nell'utilizzo di servizi di *webhooks*, e della fine dello svolgimento dello *stage* della persona con cui collaboravo, tali studi non hanno portato a soluzioni effettivamente utilizzate.

Code

Le attività al fine di predisporre la fase Code, per la sua applicazione a progetti Power Automate e Power Apps, comprendono principalmente la realizzazione e la documentazione del sistema di versionamento individuato in fase di progettazione e la documentazione e l'applicazione degli strumenti per l'analisi statica del codice identificati.
Per il versionamento ho adattato, prima in un *branch* di prova e poi nei restanti, il *repository* Git dell'applicazione aziendale Report di visita in modo da rispecchiare le norme definite. Esse riguardano le regole da seguire per versionare il progetto e la struttura dei percorsi dei *file*, divisi in base alla loro funzione.
Ho definito tali norme con la collaborazione di parte del *team* di sviluppo e dello stagista universitario responsabile del progetto Applicativi di DevOps in ambito Sistemi.
Per fare in modo di contenere nel *repository* i *file* relativi ai flussi Power Automate e le applicazioni Power Apps, ho predisposto una cartella apposita al fine di contenere il pacchetto ZIP relativo alla Microsoft Solution del progetto.
Tale pacchetto nel *repository* assume valore in ottica di *backup* e ripristino, in caso sia necessaria l'esecuzione di una specifica versione del progetto.
Per fare in modo che Power Apps riesca ad utilizzare la funzione integrata che gli permette di interfaccairsi con Git direttamente dalla sua interfaccia grafica, è necessario che nel *repository* sia presente tale applicazione in un formato non compresso. Abbiamo pertanto deciso di mantenere, per quanto ridondante, una copia dell'applicazione al fine di agevolare le fasi di sviluppo.
Per l'analisi statica del codice ho esplorato altri strumenti esterni come l'applicazione di strumenti di analisi statica consolidati, ad esempio SonarQube, applicati tramite IDE esterni ai dati generati da Power Automate e Power Apps.
Inoltre ho constatato la possibilità di utilizzare gli strumenti diagnostici nativi di tali tecnologie anche tramite i comandi della Power Platform CLI, in modo da automatizzare la raccolta dei risultati.
Tuttavia, questa funzione risulta superflua, poiché i risultati sono più facilmente accessibili tramite l'interfaccia grafica durante le fasi di sviluppo. Inoltre, perde di utilità se, per verificare lo stato del codice, è necessario eseguire uno *script* e analizzare un complesso *file* JSON.
Ho redatto e condiviso i documenti "Norme di progetto" e "Analisi statica" con il *team* di sviluppo e con il tutor aziendale.
Infine ho prodotto una presentazione PowerPoint contenente le principali informazioni presenti nei documenti citati e l'ho esposta al *team* di sviluppo.

Build

Al fine di collegare il Job Jenkins con il *repository*, senza dover utilizzare *token* di accesso GitHub personali, ho utilizzato la funzione che permette di generare "GitHub Apps". Esse sono applicazioni integrate in GitHub che permettono il collegamento tra i *repository* desiderati con altri specifici strumenti e servizi, definendone con precisione il livello di accesso e autorizzazioni.

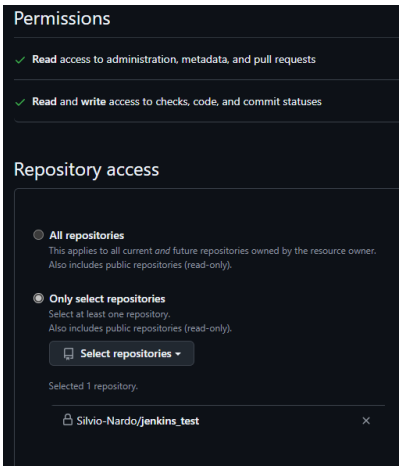


Figura 3.11: Configurazione dei permessi e selezione dei repository connessi ad una GitHub App.

Per automatizzare l'esportazione delle Microsoft Solutions e il loro caricamento nel *repository*, ho adottato un Multibranch Pipeline Job Jenkins. Il relativo Jenkinsfile da me prodotto esegue, tramite i propri Stage, le seguenti operazioni:

- Seleziona il corretto *branch* del *repository* ed esegue un comando *pull*
- Cerca modifiche nella cartella relativa all'applicazione Power Apps
- Esegue l'autenticazione Micorsoft tramite gli appositi comandi forniti dalla Power Platform CLI
- Esporta la Microsoft Solutions
- Controlla l'eventuale presenza di una omonima Solution già presente nel *repository* e, se presente, la elimina
- Carica il pacchetto relativo alla nuova versione della Solution nel *repository*

Ho redatto il documento "Guida Jenkins", il quale comprende tutte le nozioni e le norme relative all'adozione di Jenkins, al fine di applicare la fase Build ai progetti Power Automate e Power Apps e l'ho condiviso con il *team* di sviluppo e con il tutor aziendale.

Test

L'applicazione di *test* automatici per i flussi Power Automate è avvenuta, oltre all'utilizzo degli strumenti nativi già discussi, tramite lo sviluppo di soluzioni esterne sfruttando le chiamate HTTP. Esse permettono la comunicazione e lo scambio di dati tra flussi Power Automate e altri flussi o applicazioni: esiste infatti la possibilità di richiamare lo specifico Trigger "Alla ricezione di una richiesta HTTP", il quale genera un personale URL (Uniform Resource Locator), ovvero una sequenza di caratteri che identifica univocamente l'indirizzo di una risorsa su una rete di *computer*.
In seguito è possibile utilizzare la corrispondente azione "Response" al fine di rispondere al chiamante con l'*output* della richiesta.

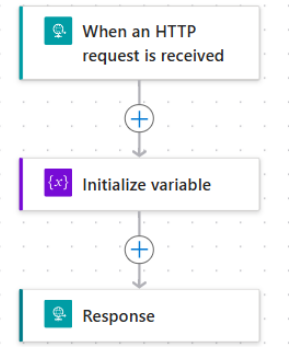


Figura 3.12: Flusso di esempio per ricevere e risondere a chiamate HTTP.

L'invio delle chiamate e l'analisi dei dati di *output* inviati dal flusso oggetto di *test*, sono avvenute mediante uno *script* Node.js con l'ausilio della libreria "Axios", la quale offre funzionalità per l'invio e la ricezione asincrona delle chiamate HTTP.
Per poter inviare una chiamata al flusso desiderato tramite *script*, è necessario ottenere, tramite un'ulteriore chiamata HTTP, un *token* Microsoft di autenticazione.

```
const url = `https://login.microsoftonline.com/${tenantId}/oauth2/v2.0/token`;
const params = new URLSearchParams();
params.append('grant_type', 'client_credentials');
params.append('client_id', clientId);
params.append('client_secret', clientSecret);
params.append('scope', 'https://service.flow.microsoft.com//.default');
const response = await axios.post(url, params);
return response.data.access_token;
```

Figura 3.13: Parte dello *script* responsabile per l'ottenimento del *token* di autenticazione.

In seguito è possibile, aggiungendo l'*output* ottenuto dalla precedente chiamata, inviare la richiesta di esecuzione al flusso da testare.
Grazie all'Azione del flusso "Response", lo *script* può proseguire alla ricezione della risposta contenente l'*output* di Power Automate, in modo da analizzarla e determinarne l'esito.

```
// Analizza la risposta del flusso
if (response.status === 200) {
    console.log('Risultati:', response.data);
    // Logica per determinare il successo del test
    if (response.data.testInt === 1) {
        console.log('Il test del flusso è riuscito.');
    } else {
        console.log('Il test del flusso è fallito');
    }
} else {
    // il flusso ha mandato un codice di errore
    console.log('Errore nel test del flusso:', response.status);
}
```

Figura 3.14: Parte dello *script* responsabile per l'analisi dell'esito dei *test* su flussi Power Automate.

Nell'immagine è mostrata una porzione dello *script* nel quale viene, a fini dimostrativi,

controllato se il risultato della chiamata è stata un successo, definito con codice 200, e se l'*output* del flusso è uguale ad un intero di valore 1. In funzione dell'esito del controllo e dello stato di successo della chiamata, viene mostrato a schermo un relativo messaggio.
Ai fini di rendere automatico questo processo, ho incluso l'esecuzione dello *script* illustrato all'interno del nel Jenkinsfile relativo al Job Jenkins dell'applicazione aziendale Report di visita. Tale pratica garantisce la qualità e la funzionalità attesa dai flussi testati, però è da considerare che la natura dei prodotti realizzati con Power Automate richiede che essi vengano modificati, al fine di essere predisposti alla ricezione e alla risposta delle chiamate HTTP di *test*. Inoltre è da tenere in considerazione che tali funzioni necessitano della licenza *premium*.

Per le applicazioni Power Apps ho esplorato la possibilità di estrarre, tramite la funzione "Scarica suite", un *file* contenente le azioni registrate dallo strumento Test Studio ma, in funzione delle conclusioni esplicitate precedentemente riguardo tale strumento, non ho integrato tale strategia nel sistema di *testing*.

Ho prodotto il documento "Test dinamici", contenente tutte le informazioni relative ai metodi individuati per effettuare tali *test* su Power Automate e su Power Apps, sia con strumenti nativi che esterni. Ho aggiornato il documento "Guida Jenkins" al fine di contenerne anche le nozioni riguardanti quanto descritto.
Ho poi condiviso tali documenti con il *team* di sviluppo e con il tutor aziendale.
Inoltre ho prodotto, ed esposto al *team* di sviluppo, una presentazione PowerPoint contenente le principali informazioni apprese e le soluzioni individuate e sviluppate relativamente alle fasi Build e Test.

Deploy

Al fine di comprendere e testare l'applicazione della fase Deploy ho esplorato, insieme a parte del *team* di sviluppo, le funzionalità legate all'importazione delle Microsoft Solutions, gestite e non gestite, all'interno di diversi ambienti di produzione.
Al fine di agevolare il collegamento agli *account* e ai servizi Microsoft corrispondenti al nuovo ambiente, ho utilizzato i "riferimenti alle connessioni" all'interno della relativa Solution. Essi sono entità che rappresentano il collegamento tra un'applicazione, un servizio o un flusso e una risorsa esterna, e offrono la possibilità di essere cambiati in maniera dinamica durante la fase di importazione in un ambiente.

L'applicazione delle fasi di DevOps Release, Operate e Monitor, a seguito delle attività di analisi e progettazione, non hanno avuto necessità di attività di sviluppo aggiuntive.

3.3.3 Attività sulle applicazioni aziendali

Report di visita

Le attività di sviluppo dell'applicazione aziendale Report di visita hanno affiancato le attività di ricerca del mio *stage* per la maggior parte della sua durata.
Esse, oltre all'integrazione delle pratiche DevOps descritte precedentemente, si sono focalizzate sulle strategie di *retrieve* dei dati tramite appositi flussi Power Automate collegati all'applicazione.
Ho usato SharePoint come strumento di archiviazione principale, ma ho testato anche la possibilità *premium* di leggere dati da SQL Server.

Le principali difficoltà che ho affrontato sono relative ai limiti sul numero di *record* ottenibili per ogni richiesta a SharePoint:

- Essendo per Power Apps una funzione non delegabile, ovvero che non può delegare l'elaborazione dei dati al *server*, è presente il limite di 500 *record* ottenibili di *default* estendibile fino a 2000 modificando le impostazioni. Inoltre eventuali filtri sulla ricerca vengono applicati solo dopo aver ricevuto quel numero di dati, pertanto, il risultato ottenuto è un insieme limitato del risultato atteso.

- Eseguendo l'analoga funzione da un flusso Power Automate, si incorre in un ulteriore limite imposto da Sharepoint uguale a 5000 risultati ottenibili per chiamata.
  In merito ho esplorato soluzioni legate al precaricamento di tutti i dati, mediante chiamate multiple, al fine di applicare successivamente il filtraggio dei dati richiesti, e all'utilizzo di chiamate HTTP a servizi SharePoint. Ho successivamente scartato tali soluzioni a causa degli eccessivi tempi di esecuzione e ad altre limitazioni incontrate, come l'impossibilità per SarePoint di eseguire il comando di ricerca per sottostringa.

- Dopo aver ottenuto i dati desiderati, sfruttando particolari funzionalità di indicizzazione delle liste SharePoint, ho incontrato difficoltà legate all'aggiornamento grafico di alcuni componenti fondamentali dell'applicazione utilizzati per mostrare dinamicamente i contenuti ricavati.
  Ho risolto tale problema in collaborazione con il *team* di sviluppo mediante la realizzazione di un nuovo componente *custom*.

Al fine di fare *test* su liste SharePoint, popolate dinamicamente da un numero elevato di elementi, ho generato tabelle Excel in maniera automatica scrivendo appositamente degli "Office Scripts".
Le mie attività di sviluppo legate alle applicazioni Registro accessi e "Non conformità" sono legate ad aspetti grafici e all'individuazione di possibili risoluzioni di problemi legati alle singole funzioni Fx.

3.3.4 Attività con Angular

Al fine di comprendere il funzionamento e lo sviluppo di progetti basati su Angular, ho realizzato una basilare applicazione *web* che mostrava nell'interfaccia grafica immagini e testo.
Essa è stata il punto di partenza per lo sviluppo di una *suite* di *test* eseguiti con l'apposito strumento "Karma". Ho poi automatizzato tale esecuzione mediante un Job Jenkins di tipo *pipeline*, il quale:

- Seleziona e utilizza la versione di Node.js predeterminata
- Preleva il progetto dal *repository* Git
- Esegue i *test* di unità definiti
- Compila i *file* e genera il relativo pacchetto composto da elementi HTML e JavaScript
- Mostra a schermo il risultato dei *test* in un apposito grafico

- Salva gli artefatti creati in Jenkins, vincolando la presenza dei salvataggi ai soli tre più recenti
- Pulisce il *workspace* relativo al Job e alle compilazioni avvenute

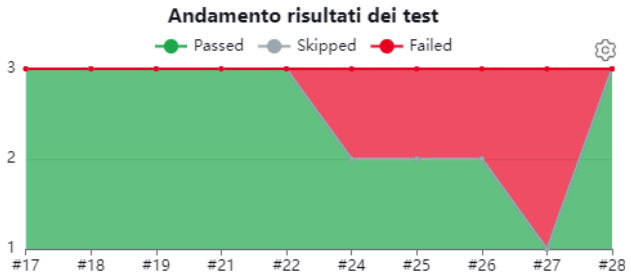


Figura 3.15: Grafico dell'esito dei *test* generato da Jenkins.

Inoltre ho applicato quanto appreso per produrre un ulteriore *pipeline* Job al fine di replicare le fasi di DevOps anche su progetti aziendali realizzati con Angular.
Infine ho generato il documento "Angular in Jenkins" contenente quanto affrontato in merito e l'ho esposto al *team* di sviluppo e al tutor aziendale.

3.4 Risultati raggiunti

3.4.1 Risultati qualitativi

Lo studio delle tecnologie Power Automate e Power Apps ha fatto emergere la loro natura: essi sono *software* utili nei casi in cui si debba realizzare un'applicazione semplice rapidamente, ma è da considerare che sono strumenti ancora relativamente nuovi, quindi privi di una documentazione esaustiva e aventi *features* soggette a frequenti e importanti modifiche.
Offrono la possibilità di connettersi agevolmente con i servizi Microsoft ma presentano allo stesso tempo un alto numero di limitazioni, le quali non offrono la libertà di sviluppo ottenibile con i consolidati strumenti e linguaggi di programmazione.
Lo studio da me svolto durante lo *stage* ha fatto emergere l'effettiva possibilità di applicare le metodologie DevOps a progetti realizzati con Power Auotmate e Power Apps.

Tramite periodi di sviluppo collaborativo, ho dimostrato la possibilità di applicare strategie di collaborazione e condivisione delle risorse tra membri del *team* di sviluppo.
Ho definito le norme di versionamento che il *team* deve seguire per organizzare efficacemente tutte le parti che compongono un progetto Power Automate/Power Apps, testando la possibilità di effettuare il ripristino di specifiche versioni rilasciate.
Ho applicato metodologie di analisi statica del codice e *testing* dinamico, sia mediante strumenti nativi che esterni, al fine di verificare il corretto funzionamento e garantire la qualità dei prodotti realizzati.
Ho dimostrato la possibilità di gestire il ciclo di vita del *software* prodotto, relativamente alle tecnologie in oggetto, comprese le fasi di distribuzione dei prodotti e il loro

monitoraggio.
Ogni nozione da me acquisita, ogni norma stipulata e ogni soluzione implementata è stata da me documentata e resa disponibile al personale attraverso gli strumenti di condivisione aziendale.

3.4.2 Risultati qualitativi

Tabella 3.2: Documenti che ho prodotto durante lo *stage*.

Documenti prodotti
"Analisi statica del codice"
"Norme di versionamento"
"Test dinamici"
"Guida Jenkins"
"Angular in Jenkins"
Presentazione sulle funzionalità, i lati positivi e i lati negativi di Power Automate e Power Apps
Presentazione relativa alle limitazioni di tali tecnologie e le rispettive soluzioni individuate
Presentazione relativa al versionamento e all'analisi statica del codice
Presentazione relativa ai processi Build e ai *test* dinamici
Presentazione finale realizzata in collaborazione con gli altri stagisti universitari riguardo a quanto fatto durante i nostri *stage*

3.4.3 Risultati quantitativi

Tabella 3.3: Software che ho prodotto durante lo *stage*.

Software prodotti
PoC sui flussi approvativi
PoC sull'integrazione di chiamate HTTP con i flussi
Jenkinsfile per progetti Power Automate/Power Apps
Script Excel per la creazione automatica di tabelle
Script Node.js per il *test* di flussi tramite chiamate HTTP
Sviluppo di funzioni Fx sul prodotto aziendale Report di visita
Flussi per il *retreive* dei dati da liste SharePoint
Progetto di esempio Angular
Jenkinsfile relativo al progetto di esempio Angular

Glossario

Back-end Gli sviluppatori *back-end* si occupano di scrivere il codice che permette di gestire le operazioni come la gestione degli utenti, l'elaborazione dei dati e la gestione delle richieste provenienti dall'interfaccia grafica. Alcuni esempi di linguaggi di programmazione utilizzati per la scrittura della logica lato *server* sono Ruby, Java e Python. 5

Cloud I servizi *cloud* permettono alle aziende e agli utenti di archiviare e accedere ai dati e alle applicazioni da qualsiasi luogo, con vantaggi in termini di scalabilità, flessibilità, costi ridotti e aggiornamenti automatici. 1, 5, 9, 46

Continuous Deployment (CD) L'adozione di questo approccio consente di rilasciare nuove versioni del *software* in modo rapido e frequente, garantendo che le funzionalità siano disponibili per gli utenti finali in tempi brevi. Inoltre, il *team* di sviluppo non è più obbligato ad interrompere lo sviluppo per prepararsi ed effettuare i rilasci. Questi ultimi sono meno rischiosi poiché le modifiche apportate al prodotto sono tipicamente contenute ed è quindi più agevole identificare eventuali problemi. Infine il cliente ha la possibilità di fornire *feedback* costantemente potendo verificare ogni avanzamento. 10, 45

Continuous Integration (CI) Ogni integrazione viene verificata automaticamente attraverso l'esecuzione di *test* per rilevare rapidamente eventuali errori o conflitti nel codice. Il concetto della *Continuous Integration* è stato originariamente proposto come contromisura preventiva per il problema dell'"*integration hell*", ovvero le difficoltà dell'integrazione di porzioni di *software* sviluppati in modo indipendente su lunghi periodi di tempo e che di conseguenza potrebbero essere significativamente divergenti. 10, 45

DevOps Metodologia che enfatizza l'automazione, la condivisione di responsabilità e il miglioramento continuo, utilizzando strumenti e processi che supportano la Continuous Integration, il Continuous Deployment e il monitoraggio costante dei sistemi. 2, 11, 13–15, 17, 20, 24, 25, 27, 28, 30, 41, 43

HyperText Transfer Protocol (HTTP) Protocollo a livello applicativo, ovvero il livello più alto definito dal modello OSI (Open Systems Interconnection), il quale rappresenta uno *standard* architetturale per reti di calcolatori. Tale livello è responsabile della gestione delle comunicazioni tra applicazioni, fornendo i servizi necessari per lo scambio di dati strutturati e significativi tra *client* e *server*. 20, 23, 34, 37, 39–42, 44

Information Technology (IT) Sottocategoria dell'ICT, include infrastrutture *hardware*, *software*, reti e servizi correlati. Numerose industrie sono legate alla tecnologia dell'informazione, inclusi produttori di elettronica, semiconduttori, attrezzature per la telecomunicazione, commercio elettronico, web design e servizi informatici. L'infrastruttura IT può esistere localmente oppure in un ambiente cloud. 1, 2, 11

Piccole e Medie Imprese (PMI) Categoria di aziende che, in base a dimensioni e fatturato, rientrano nelle definizioni stabilite da enti nazionali o internazionali. Nell'Unione europea esse sono contraddistinte da un numero di dipendenti inferiore a 250 e un fatturato inferiore o uguale a 50 milioni di euro. 1, 3

Sistemi Sistemi S.p.A. è una società italiana partecipata con Wintech S.p.A. Essa possiede tecnologie ed ambienti di sviluppo dedicati al fine di creare soluzioni *software* gestionali e servizi per professionisti e imprese, soprattutto in ambiti relativi a studi professionali di commercialisti, consulenti del lavoro e avvocati, imprese e associazioni di categoria. 1, 14, 17, 20

Tecnologie dell'Informazione e della Comunicazione (ICT) Il termine include una vasta gamma di strumenti e risorse, come *computer*, *software*, reti di telecomunicazione, *internet* e dispositivi mobili, che consentono la creazione, l'archiviazione, la gestione e lo scambio di dati e contenuti. 1, 12, 46

Bibliografia

Siti web consultati

Atlassian, Scrum. URL: https://www.atlassian.com/it/agile/scrum.

Documentazione Jenkins. URL: https://www.jenkins.io/doc/.

Documentazione Power Apps. URL: https://learn.microsoft.com/en-us/power-apps/.

Documentazione Power Automate. URL: https://learn.microsoft.com/en-us/power-automate/.

Sistemi. URL: https://www.sistemi.com/chi-siamo/.

Wikipedia. URL: https://it.wikipedia.org/.

Wintech. URL: https://www.wintech.it/.