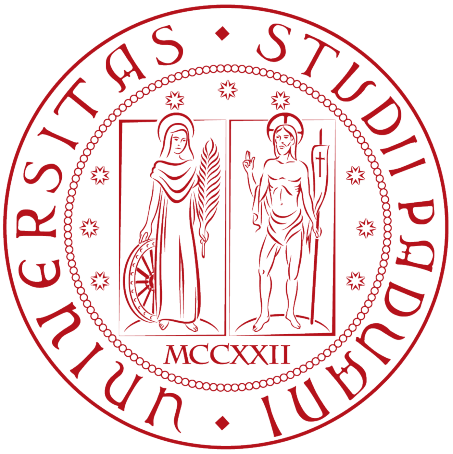
Università degli Studi di Padova

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

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Svolgimento stage

In questo capitolo vengono descritte tutte le attività da me svolte durante lo *stage*, divise nelle sezioni "Analisi", "Progettazione", "Programmazione" e "Verifica e validazione", in modo da fornire una panoramica chiara e strutturata del lavoro svolto, evidenziando il processo seguito e le competenze acquisite in ciascuna fase.
Esse non sono intese come completamente sequenziali bensì, fin dalla prima fase, sono presenti tutte le attività correlate in modo da coprire l'intero periodo di *stage*.

1.1 Analisi

In questa sezione sono presenti tutte le attività analitiche da me svolte e il suo scopo è descrivere le modalità con cui ho compreso i bisogni, i requisiti e le tecnologie del mio progetto di *stage*.
Essendo quest'ultimo focalizzato per la maggior parte su attività legate alla ricerca e all'esplorazione tecnologica, la fase di analisi rappresenta la parte più corposa delle mie attività.

1.1.1 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.

Essi sono:

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.
O1.2	Sviluppo collaborativo sui flussi 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.
O2	Comprensione ed utilizzo della tecnologia Microsoft Power Apps e relative fasi di sviluppo collaborativo.
O2.1	Comprensione di Power Apps affiancando il *team* di sviluppo fino al raggiungimento della comprensione dei metodi lavorativi e dei prodotti aziendali sviluppati con tale strumento.
O2.2	Identificazione e definizione 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 all'applicazione di analisi statica del codice su progetti realizzati con tecnologie Power Automate/Power Apps.
D2.5	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*.

Tabella 1.1: Tabella dei requisiti progettuali.

1.1.2 Ambiente di lavoro

Durante i primi giorni dello *stage* sono stato introdotto all'ambiente di lavoro e alle tecnologie di comunicazione e collaborazione.
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.

1.1.3 Tecnologie oggetto di stage

In questo sottocapitolo vengono descritti 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*.
I requisiti 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.

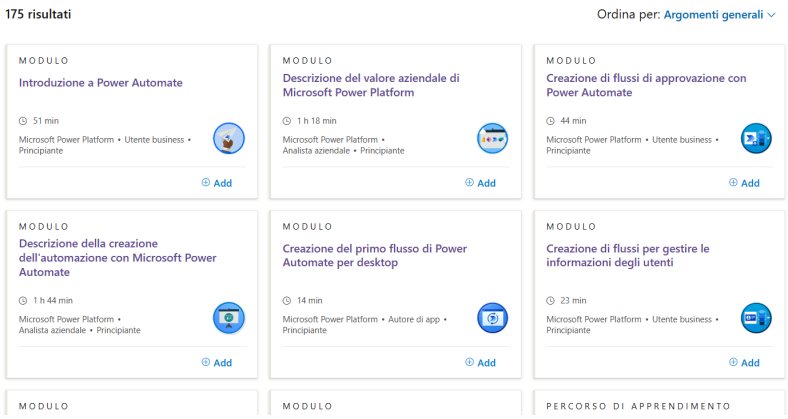


Figura 1.1: Moduli formativi per Power Automate.

Fonte: https://learn.microsoft.com/it-it/training/browse/?products=power-automate.

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. È possibile creare dei nuovi flussi partendo da altri progetti pubblici, o a noi condivisi, e modelli offerti da Microsoft da adattare alle 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.

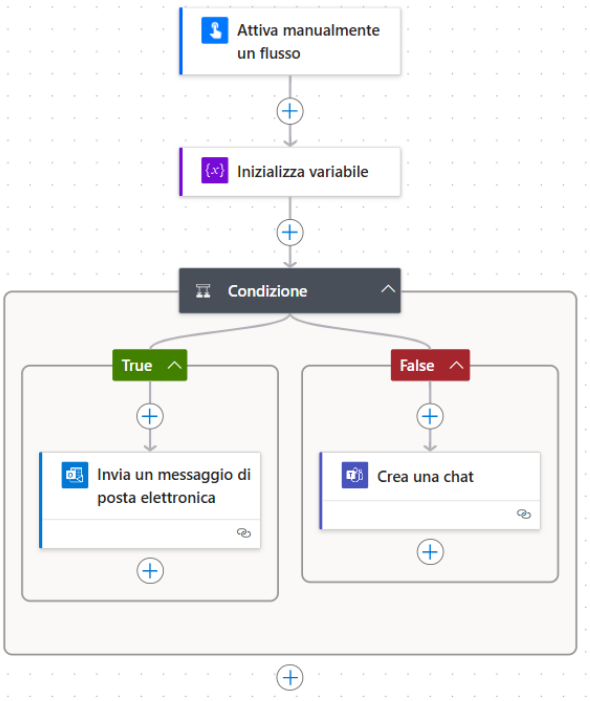


Figura 1.2: Esempio di un flusso Power Automate istantaneo.

Successivamente è emersa, da parte del 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 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.
Il secondo, più concreto e integrabile con i prodotti aziendali, è basato 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 d'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 le applicazioni 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.
Lo sviluppo di tali applicazioni 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".

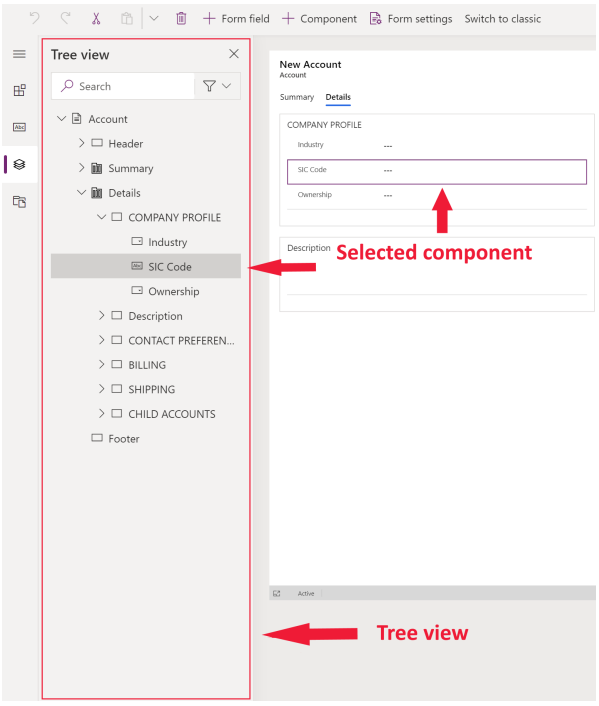


Figura 1.3: Elemento Tree View per la gestione gerarchica dei componenti.
Fonte: https://learn.microsoft.com/en-us/power-apps/maker/model-driven-apps/using-tree-view-on-form.

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.

1.1.4 Tecnologie strumentali

In questo sottocapitolo vengono descritti i metodi di apprendimento e le nozioni apprese durante la fase di analisi degli strumenti e delle tecnologie adottate al fine di perseguire gli obiettivi e soddisfare i requisiti di *stage*.

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. 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.

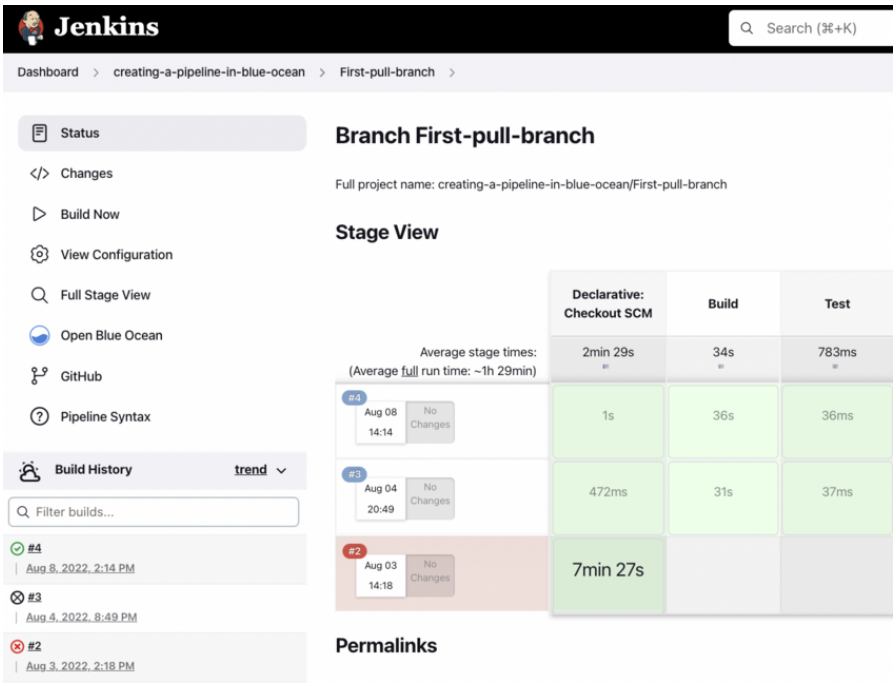


Figura 1.4: Esempio interfaccia di Jenkins.
Fonte: https://commons.wikimedia.org/wiki/File:Updated-jenkins-view.png.

1.1.5 Microsoft Solutions

Un'ulteriore oggetto di analisi e studio è stato 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 Micorsoft 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 tecnologia

sono materia di discussione presente nel capitolo 3.2 Progettazione.

1.1.6 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 tastato 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.
Esso 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 garantitala 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.

1.1.7 Applicazioni aziendali

Oltre all'analisi avvenuta 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, è stato fin da subito definito che avrei alternato tali attività con fasi di sviluppo su applicazioni aziendali.
Questo approccio mi ha permesso 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 al capitolo 3.3 Programmazione.
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
- Caricamento automatico dei dati generati nel *database* aziendale

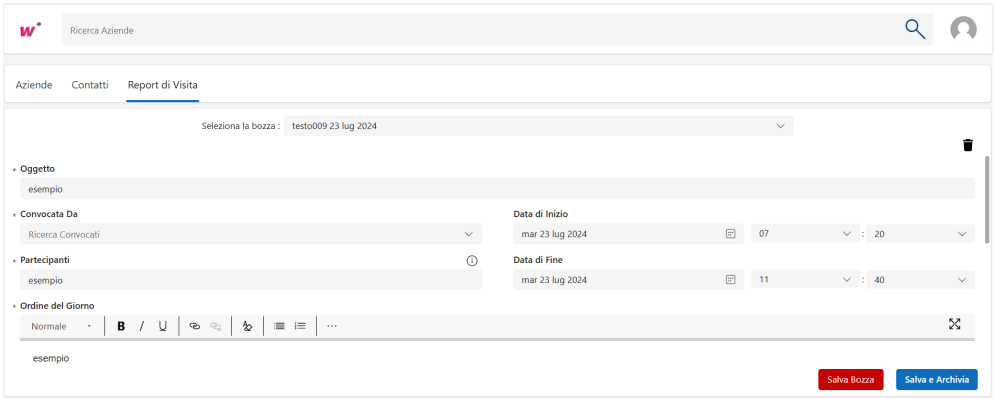


Figura 1.5: Interfaccia grafica di Report di visita.

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.

1.1.8 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.

1.2 Progettazione

In questa sezione sono presenti tutte le attività di progettazione da me svolte e il suo scopo è descrivere le modalità con cui ho individuato le soluzioni ai bisogni progettuali in modo da soddisfarne i requisiti.

1.2.1 Scope di Power Automate e Power Apps

A seguito dell'analisi avvenuta sulle tecnologie oggetto di *stage* è stato possibile identificare 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 integrazione con servizi esterni dalla *suite* Microsoft, tali strumenti non sono più ideali in quanto si incomberebbe in numerosi vincoli tecnologici e difficoltà di integrazione. Questo è un fattore importante da considerare nelle fasi di progettazione di un progetto.

1.2.2 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.

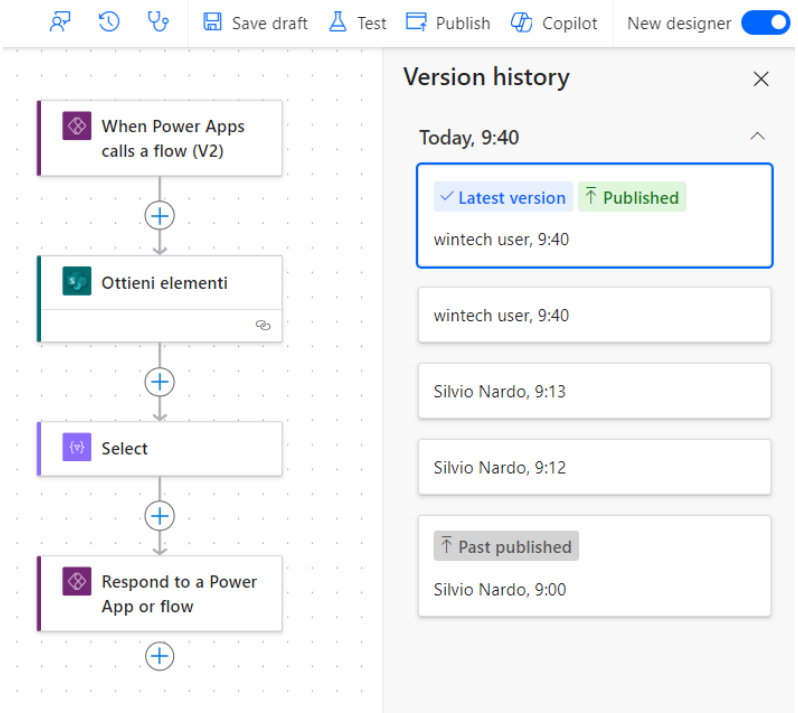


Figura 1.6: Funzionalità Version history nei flussi Power Automate.

Inoltre, essendo i componenti di una Soluzione contenuti in un unico pacchetto esporta-

bile, 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, è stata individuato il "Multibranch 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.

1.2.3 Soluzioni individuate per le fasi DevOps

Plan

Come strumenti identificati per la pianificazione delle attività sono stati utilizzati 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 è stata individuata 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.

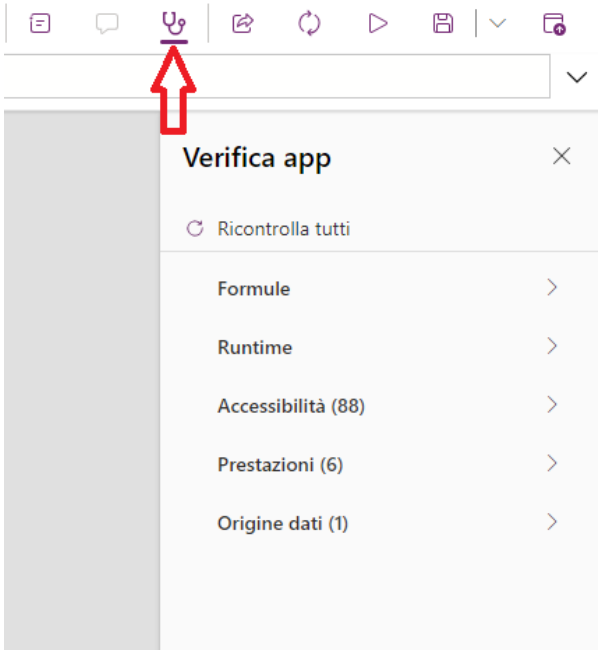


Figura 1.7: Funzionalità Verifica App nelle applicazioni Power Apps.

Gli strumenti di analisi statica esterni non sono stati utilizzati 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.

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 1.8: 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*.
È inoltre stata individuata 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 nel capitolo 3.3 Programmazione.

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 delle sue 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, 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 tali 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 verisonamento 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".

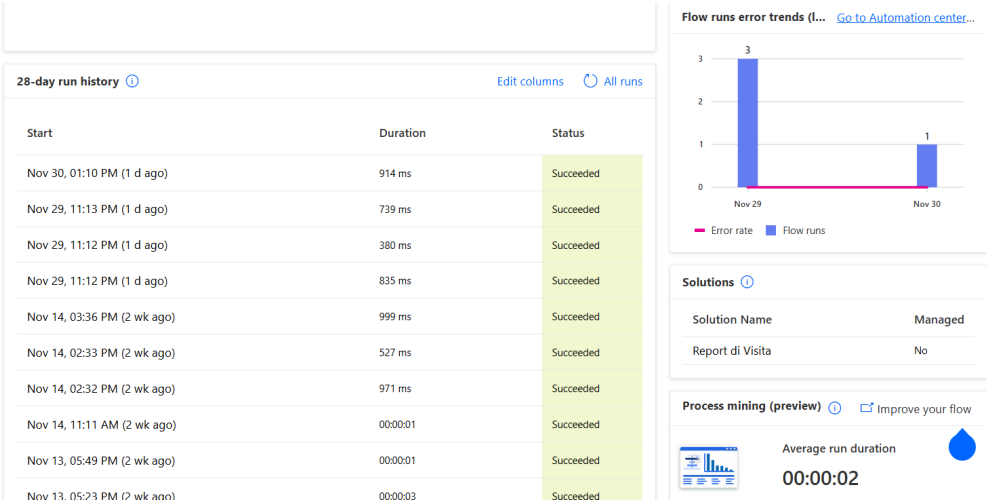


Figura 1.9: Vista parziale dell'interfacccia atta a monitorare un flusso.

Applicazioni aziendali

Le fasi di progettazione relative alle applicazioni aziendali su cui ho lavorato 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.

1.3 Sviluppo software e documentale

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

1.3.1 PoC iniziali

Le prime fasi di sviluppo dello *stage* da me frequentato 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 a che non si verifichi l'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.

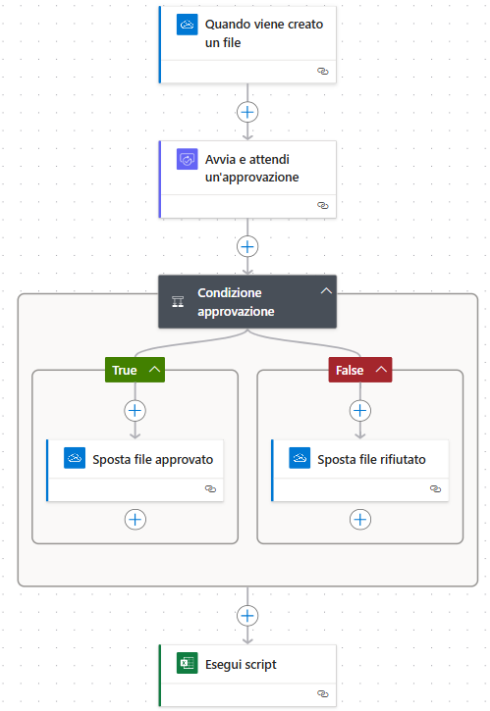


Figura 1.10: Flusso per testare condizioni, approvazioni ed esecuzione di *script*.

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*.

1.3.2 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 LINKDELPROGETTODIGIACOMO, 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 da parte di un'applicazione *web* al fine 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 da me svolte 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.
Tali norme sono state definite con la collaborazione di parte del *team* di sviluppo e dello stagista universitario responsabile del progetto LINKPROGETTOBROTTO.
Per fare in modo di contenere nel *repository* i *file* relativi ai flussi Power Automate e le applicazioni Power Apps, è stata predisposta 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, nel caso in cui si necessitasse 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. È pertanto stato 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.
Questa funzione risulta però superflua in quanto i risultati sono di più agevole fruizione con l'adozione di tali strumenti tramite l'interfaccia grafica durante le fasi di sviluppo, e perde di significato se, per verificare lo stato del proprio codice, è necessario eseguire uno *script* e studiare un complicato *file* JSON.
I documenti "Norme di progetto" e "Analisi statica" sono stati redatti e sono stati condivisi con il *team* di sviluppo e con il *tutor* aziendale.
Inoltre 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.

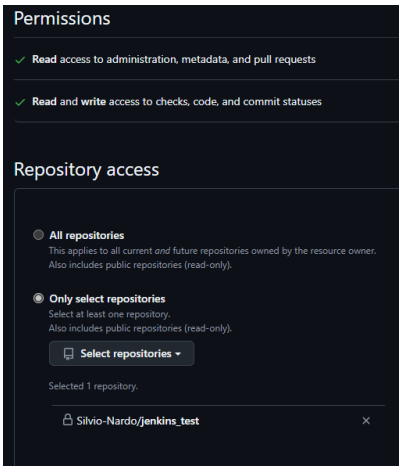


Figura 1.11: Configurazione dei permessi e selezione dei repository connessi ad una GitHub App.

Per automatizzare l'esportazione delle Microsoft Solutions e il loro caricamento nel *repository*, ho adottato un Multibranch Pipeline Job Jenkins. Il relativo Jenkinsfile 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 prodotto 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 a 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.

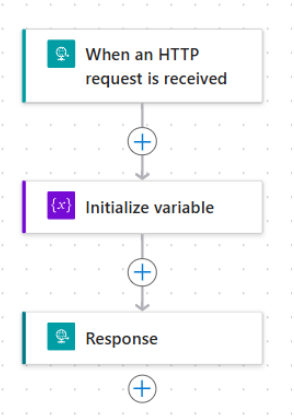


Figura 1.12: Flusso di esempio per ricevere e risondere a chiamate HTTP.

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. In seguito

```
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 1.13: Parte dello *script* responsabile per l'ottenimento del *token* di autenticazione.

è possibile, aggiungendo quanto ritornato 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 1.14: 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, l'esecuzione dello *script* illustrato è stato incluso 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, tale strategia non è stata integrata 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.
Tali documenti sono stati condivisi 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, sono state utilizzate, ed inserite all'interno della relativa Solution, i "riferimenti alle connessioni". 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.

1.3.3 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.
Gli strumento di archiviazione utilizzato principalmente è stato SharePoint ma è stata testata anche la possibilità *premium* di leggere dati da SQL Server.
Le principali difficoltà affrontate 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 sono state esplorate 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. Queste soluzioni sono state successivamente scartate 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, sono state incontrate difficoltà legate all'aggiornamento grafico di alcuni componenti fondamentali dell'applicazione utilizzati per mostrare dinamicamente i contenuti ricavati.
  Tale problema è stato risolto tramite lo sviluppo 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.

1.3.4 Sviluppo Angular

Al fine di comprendere il funzionamento e lo sviluppo di progetti 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". Tale esecuzione è poi stata automatizzata in 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

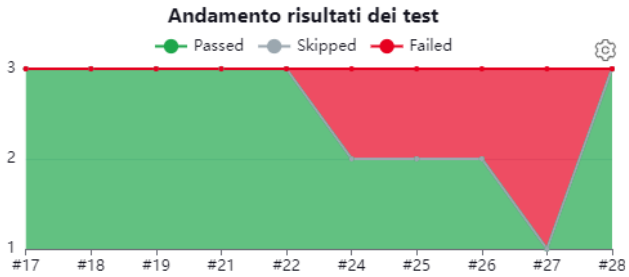


Figura 1.15: Grafico dell'esito dei *test* generato da Jenkins.

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.

1.4 Risultati raggiunti

1.4.1 Qualitativamente

Lo studio delle tecnologie Power Automate e Power Apps ha fatto emergere la loro natura: essi sono *software* utili da utilizzare 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 è stata dimostrata 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 documentata ed è stata resa disponibile al personale attraverso gli strumenti di condivisione aziendale.

1.4.2 Quantitativamente

I documenti che ho prodotto durante lo *stage* sono:

- "Analisi statica del codice"
- "Norme di versionamento"
- "Test dinamici"
- "Guida Jenkins"
- "Angular in Jenkins"
- Presentazione sulle funzionalità, i lati postitivi 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*

Il software prodotto è stato:

- 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

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. 27

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. 27

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, 6, 7, 9, 10, 12, 23, 25

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*. iv, 2, 5, 16, 19, 21–24, 26

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. 2

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/.