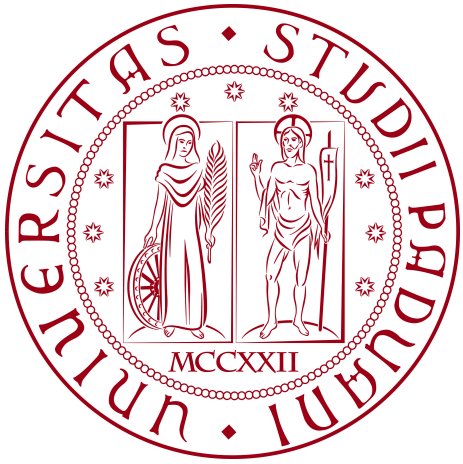
Università degli Studi di Padova

Dipartimento di Matematica "Tullio Levi-Civita"

Corso di Laurea in Informatica



Ottimizzazione della produzione in una sala stampa 3D: programmazione a vincoli e Google OR-Tools.

Tesi di laurea triennale

Relatrice

Prof.ssa Losiouk Eleonora

Laureando

Mazzotti Matteo

Matricola 2068245

Anno Accademico 2025-2026

Ringraziamenti

Desidero esprimere la mia gratitudine alla professoressa Losiouk Eleonoraper l'aiuto e il sostegno che mi ha dato durante la stesura dell'elaborato.

Vorrei anche ringraziare, con affetto, i miei genitori per il loro sostegno, il grande aiuto e la loro presenza in ogni momento durante gli anni di studio.

Desidero poi ringraziare i miei amici per i bellissimi anni trascorsi insieme e le mille avventure vissute.

Padova, Dicembre 2025 *Mazzotti Matteo*

Sommario

L'azienda Spazio Dev dispone di una sala stampa 3D la cui pianificazione manuale della produzione richiede un notevole tempo operativo. Il presente lavoro descrive lo sviluppo di un sistema automatizzato, basato sulla programmazione a vincoli e sull'utilizzo di Google OR-Tools (CP-SAT), in grado di proporre all'operatore una coda di stampa ottimizzata secondo le buone pratiche aziendali. Il sistema, integrato con il software gestionale esistente, costituisce una soluzione concreta al problema della pianificazione manuale, contribuendo a migliorare la produttività della sala stampa e a ottimizzare l'impiego delle risorse.

Indice

Elenco delle figure

Elenco delle tabelle

Elenco dei codici sorgenti

Acronimi e abbreviazioni

API Application Program Interface. 36

ASGI Asynchronous Server Gateway Interface. 26

JWT JSON Web Token. 25

ORM Object-Relational Mapping. 25

SDK Software Development Kit. 36

TSA Termine solo acronimo. 36

Glossario

API In informatica un'Application Programming Interface (API) è un insieme di regole, contratti e punti di accesso che un componente software espone per consentire ad altri componenti o servizi di interagire in modo controllato, nascondendo i dettagli implementativi e semplificando l'integrazione tra sistemi eterogenei. 3, 36

Backlog Elenco ordinato e dinamico di elementi di lavoro (item) che rappresentano valore da realizzare. In Scrum, il Product Backlog, di responsabilità del Product Owner, raccoglie e priorizza le esigenze del prodotto; lo Sprint Backlog è il piano dei Developer per raggiungere lo Sprint Goal durante lo Sprint. 7

Bugfix Modifica del codice, della configurazione o dei dati volta a rimuovere un malfunzionamento (bug) e a ripristinare il comportamento atteso del sistema senza introdurre cambiamenti funzionali non richiesti. 6

Bug In informatica, errore di funzionamento di un sistema o di un programma. 6

Casi d'uso Descrizioni strutturate di come gli attori interagiscono con il sistema per raggiungere obiettivi specifici, evidenziando flussi principali, varianti ed esigenze funzionali che guidano l'analisi dei requisiti. 15

Client Attore software o dispositivo che invia richieste a un servizio o *API* remoto utilizzando credenziali come `client_id` e `client_secret`; può rappresentare applicazioni, integrazioni di terze parti o componenti interne autorizzate a consumare l'interfaccia esposta. 27

Code style Insieme di regole e convenzioni formali che disciplinano formattazione, nomenclatura e organizzazione del codice sorgente, così da mantenerlo leggibile e uniforme all'interno di un team. 25

Daily Scrum Evento giornaliero, time-box di 15 minuti, in cui i Developer ispezionano i progressi verso lo Sprint Goal e adattano il piano per le prossime 24 ore; non è una riunione di status per gli stakeholder. 7

Deploy Processo di rilascio e messa in esercizio di un'applicazione su un ambiente target (test, staging, produzione), comprendendo packaging, distribuzione, configurazione e attivazione dei servizi necessari per renderla disponibile agli utenti. 26

Design Pattern Soluzione progettuale riutilizzabile che descrive come risolvere un problema ricorrente di design software, codificando ruoli, responsabilità e interazioni tra componenti per migliorare manutenibilità, flessibilità e comunicazione all'interno del team. 23, 24

Design pattern architetturale Schema di alto livello che definisce composizione e interazione dei componenti di un sistema software per soddisfare requisiti non funzionali come scalabilità, resilienza o sicurezza, fornendo linee guida per strutturare l'intera architettura. 29

Feature Unità coerente di comportamento di un sistema che produce un beneficio osservabile per l'utente. 5

Framework Insieme coerente di componenti software riutilizzabili che fornisce struttura, astrazioni e convenzioni per sviluppare una specifica classe di applicazioni, velocizzando lo sviluppo e favorendo la consistenza del codice. 24

Git branching model Un modello di branching Git è una strategia o un insieme di regole che definisce come i team devono creare, gestire e unire i branch in un repository Git, al fine di organizzare il flusso di sviluppo. 5

Issue Unità di lavoro o ticket che rappresenta una richiesta, un bug, un miglioramento o un'attività da tracciare e gestire in un sistema di gestione del lavoro come Jira o GitHub Issues. Ogni issue dovrebbe includere contesto, criteri di accettazione e stato per facilitare la collaborazione. 8

Multi tenant Modello architetturale in cui un'unica istanza dell'applicazione serve più tenant isolati (organizzazioni o gruppi di utenti) condividendo infrastruttura e codice ma mantenendo separati dati e configurazioni, così da ottimizzare i costi e semplificare il rilascio. 28

Open source Modello di distribuzione del software in cui il codice sorgente è reso pubblico con una licenza che consente a chiunque di studiarlo, modificarlo e ridistribuirlo, favorendo collaborazione e trasparenza. 26

Ottimizzazione combinatoria Area dell'ottimizzazione che studia problemi in cui si cerca la soluzione migliore tra un insieme finito ma molto ampio di combinazioni discrete, tipicamente soggetti a vincoli (es. scheduling, routing, assegnamento). 26

Pattern a layer Design pattern architetturale che suddivide un sistema in strati indipendenti con responsabilità specifiche (es. presentazione, logica, accesso ai dati), così da ridurre le dipendenze e facilitare riuso, manutenzione e test. 29

POC Il Proof Of Concept (POC) è l'allestimento di una demo prototipale del sistema o applicazione in sviluppo o in corso di valutazione. 3

Product Owner Figura chiave di Scrum responsabile di massimizzare il valore del prodotto e del lavoro del team. Definisce e mantiene il Product Backlog, ne ordina gli elementi in base al valore e agli obiettivi, chiarisce i requisiti e accetta l'incremento completato. 7

Scalabilità Capacità di un sistema di sostenere carichi crescenti (utenti, dati o richieste) mantenendo livelli di servizio accettabili, adattando risorse hard-

ware o struttura software tramite scaling verticale, orizzontale o elastico. 29

Scrum Master Servant leader del team Scrum. Promuove e supporta Scrum come definito nella Scrum Guide, facilita gli eventi Scrum, rimuove impedimenti, tutela il team e aiuta l'organizzazione ad adottare pratiche agili. 7

SDK A Software Development Kit (SDK) is a collection of development tools in one installable package, facilitating application creation by providing a compiler, debugger, and sometimes a software framework. SDKs are typically specific to a hardware platform and operating system combination. Many application developers use specific SDKs to enable advanced functionalities such as advertisements, push notifications, etc. 36

Server di deploy Computer dedicato all'esecuzione in produzione di un'applicazione software. Ospita il codice distribuito, espone le relative API e garantisce risorse hardware e configurazioni adeguate per gestire il carico operativo previsto.. 13

Sprint Time-box tipico del framework Scrum, della durata compresa fra uno e quattro settimane, durante il quale il team realizza un incremento potenzialmente rilasciabile seguendo uno Sprint Goal condiviso. Lo Sprint include pianificazione, sviluppo, revisione e retrospettiva. 8

Sprint Planning Evento che apre lo Sprint in cui lo Scrum Team definisce lo Sprint Goal, seleziona gli elementi del Product Backlog da includere nello Sprint e pianifica il lavoro necessario nel Sprint Backlog. 7

Sprint Retrospective Ultimo evento dello Sprint dedicato all'ispezione delle modalità di lavoro del team e all'individuazione di miglioramenti pratici da implementare nel prossimo Sprint. 7

Sprint Review Evento alla fine dello Sprint per ispezionare l'incremento e adattare il Product Backlog. Coinvolge stakeholder e team per raccogliere feedback, rivedere i risultati e allineare i prossimi passi. 7

Nome del termine Descrizione. 36

Version Control System Un sistema di versionamento (Version Control System o VCS) è uno strumento software che traccia e gestisce le modifiche apportate a un file o a un insieme di file nel tempo, permettendo di recuperare versioni precedenti e di collaborare con altri utenti. 5

Capitolo 1

Introduzione

1.1 L'azienda

Spazio Dev S.r.l.[1] è una software house situata a Tombolo (PD) fondata da due soci, ad oggi l'azienda conta circa 17 dipendenti e si occupa di offrire servizi legati al mondo dello sviluppo web e dell'ottimizzazione dei processi industriali. L'azienda si occupa, nello specifico, di sviluppare siti web, e-commerce, software su misura e di integrare algoritmi che sfruttano l'intelligenza artificiale per ottimizzare la produzione o monitorare lo stato dell'azienda in tempo reale.

Figura 1.1: Logo dell'azienda Spazio Dev

1.1.1 Mugalab

Mugalab[2] è la divisione di Spazio Dev S.r.l. dedicata alla gestione della sala stampa 3D presente nella sede aziendale. Attraverso la tecnica della stampa additiva, Mugalab si occupa di:

- progettare e prototipare componenti per il settore industriale;

[1] *Sito Spazio Dev.* URL: https://spaziodev.eu/.

[2] *Sito Mugalab.* URL: https://mugalab.com/.

- progettare e prototipare componenti per dispositivi elettronici;
- disegnare e produrre oggetti ornamentali.

Figura 1.2: Logo di Mugalab

1.2 Il progetto

1.2.1 L'idea

L'idea di questo percorso di stage curriculare nasce dalla necessità dell'azienda di gestire in maniera efficiente la produzione delle componenti stampate in 3D. Attualmente lo stabilimento dispone di 12 stampanti e la pianificazione della stampa degli ordini di produzione viene effettuata manualmente da un singolo operatore, il quale si accerta di ottimizzare la produzione seguendo alcune buone pratiche che permettono di risparmiare tempo e aumentarne l'efficienza. L'obiettivo principale dell'azienda è quindi quello di ideare un sistema automatizzato, il cui scopo è creare una coda di stampa ottimizzata a partire dagli ordini di produzione non ancora evasi completamente. Tale sistema dovrà integrarsi completamente con il sistema gestionale esistente, il quale viene utilizzato quotidianamente dal personale dell'azienda per gestire la coda di stampa e registrare gli ordini ricevuti dalle piattaforme utilizzate per la vendita delle componenti e degli oggetti ornamentali (Amazon, Shopify e vendita B2B). Questo progetto è destinato esclusivamente ad uso interno dell'azienda, assumendo quindi il ruolo

di *Proof Of Concept*$_G$ della fattibilità di un sistema di schedulazione automatizzato e vincolato. Il codice potrà quindi essere ulteriormente ottimizzato e ampliato in modo tale da migliorarne l'efficienza e aumentare la qualità del risultato prodotto, oltre ad essere adattato a scenari d'utilizzo differenti da quello proposto.

1.2.2 Motivazioni e problematiche attuali

La pianificazione manuale della coda di stampa genera ritardi e inefficienze operative, con ricadute dirette sui tempi di consegna. Una schedulazione basata su un algoritmo di ottimizzazione consente di generare piani fattibili in modo più rapido e meno incline a errori, migliorando l'utilizzo delle stampanti e riducendo i tempi morti.

1.3 Organizzazione del testo

Il secondo capitolo descrive ...

Il terzo capitolo approfondisce ...

Il quarto capitolo approfondisce ...

Il quinto capitolo approfondisce ...

Il sesto capitolo approfondisce ...

Nel settimo capitolo descrive ...

Durante la stesura del testo sono state adottate le seguenti convenzioni tipografiche:

- gli acronimi, le abbreviazioni e i termini ambigui o di uso non comune menzionati vengono definiti nel glossario, situato alla fine del presente documento;
- per la prima occorrenza dei termini riportati nel glossario viene utilizzata la seguente nomenclatura: *Application Program Interface*$_G$;

- i nomi degli oggetti, delle funzioni, delle tabelle, delle colonne, delle classi e delle rotte *API* sono evidenziati con il carattere `monospaziato`.
- i termini in lingua straniera o facenti parte del gergo tecnico sono evidenziati con il carattere *corsivo*.

Capitolo 2

Il contesto aziendale

2.1 Sviluppo software

Spazio Dev è un'azienda di piccole dimensioni, di recente fondazione e ancora in fase di crescita. Di conseguenza il progetto è stato sviluppato con budget limitato e da una singola persona. I processi di sviluppo aziendali, pertanto, sono ancora in rapida evoluzione.

2.1.1 Version Control System

L'azienda utilizza Git[1] come *Version Control System*$_G$ e Gitea[2] come piattaforma di hosting per i repository. Per quanto riguarda il *Git branching model*$_G$ viene adottato il Gitflow workflow[3], adattato nel seguente modo:

- il *branch* dedicato alle versioni stabili è denominato *main*;
- il *branch* dedicato alle versioni di sviluppo è denominato *develop*;
- i *branch* dedicati alle implementazioni di *feature*$_G$ vengono denominati con lo schema *feature/<nome_feature>* dove il parametro *<nome_feature>* è una breve descrizione della *feature* che si sta codificando;

[1] *Sito ufficiale Git.* URL: https://git-scm.com/.

[2] *Sito ufficiale Gitea.* URL: https://about.gitea.com/.

[3] *Spiegazione completa Gitflow workflow.* URL: https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow.

- i *branch* dedicati ai *bugfix*$_G$ vengono denominati con lo schema *fix/<nome_bug>* dove il parametro *<nome_bug>* è una breve descrizione del *bug*$_G$ che si sta risolvendo.
- i *branch* dedicati alla documentazione vengono denominati con lo schema *docs/<nome_documentazione>* dove il parametro *<nome_documentazione>* è una breve descrizione della modifica alla documentazione.

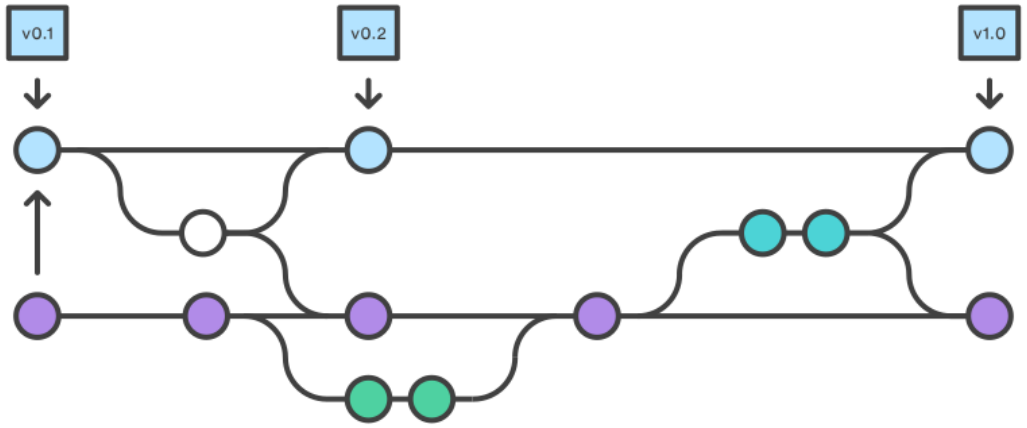


Figura 2.1: Funzionamento Gitflow Workflow

Viene inoltre adottato lo standard Conventional Commits 1.0.0[4] per la scrittura dei messaggi di commit, in particolare ogni commit possiede un messaggio così formato: *<tipo>: <descrizione>*, dove il parametro *<tipo>* è:

- *fix* in caso di commit contenente *bugfix*;
- *docs* in caso di commit di aggiornamento della documentazione;
- *feat* in caso di commit contenente *feature*.

Non sono state definite policy di protezione dei rami, requisiti di revisione o regole di merge, tali aspetti potranno tuttavia essere definiti in futuro dall'azienda.

[4] *Documentazione ufficiale Conventional Commits 1.0.0.* URL: https://www.conventionalcommits.org/en/v1.0.0/.

2.1.2 Modello di lavoro agile

L'azienda adotta il framework Scrum[5] come modello di lavoro agile[6], in particolare:

- il ruolo di *Product Owner*$_G$ viene ricoperto dai titolari dell'azienda i quali si occupano di creare e gestire il *backlog*$_G$ del prodotto, fornire indicazioni al team su quali feature implementare e decidono le scadenze dei rilasci del prodotto;
- il ruolo di *Scrum Master*$_G$ viene ricoperto dai titolari dell'azienda, i quali si occupano di pianificare *Sprint Planning*$_G$, *Daily Scrum*$_G$, *Sprint Review*$_G$ e *Sprint Retrospective*$_G$;
- il ruolo del team di sviluppo viene ricoperto dagli sviluppatori, i quali vengono divisi in gruppi e assegnati ai progetti in fase di sviluppo.

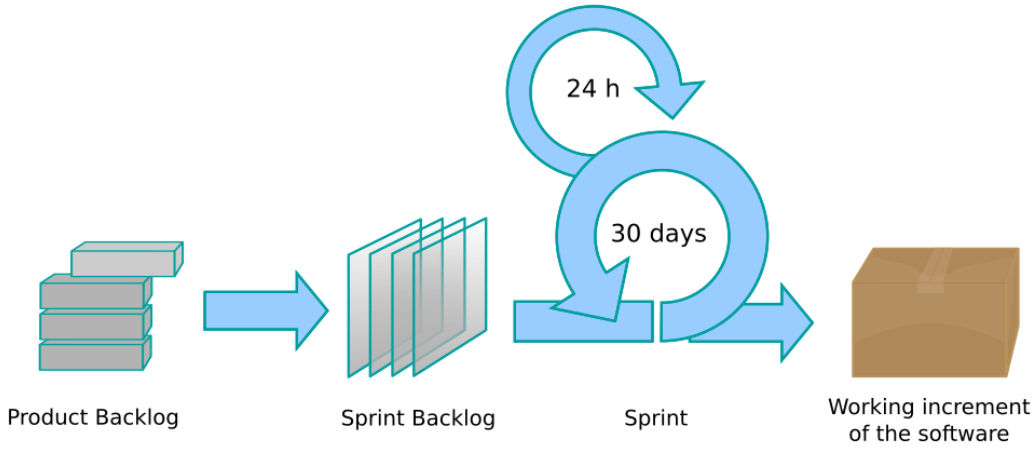


Figura 2.2: Framework Scrum

[5] *Framework Scrum*. URL: https://it.wikipedia.org/wiki/Scrum_(informatica).

[6] *Manifesto Agile*. URL: http://agilemanifesto.org/iso/it/.

2.1.3 Issue Tracking System

L'azienda utilizza Plane[7] come sistema di tracciamento delle attività. Le *issue*$_G$ sono organizzate su una board in stile Kanban composta da cinque colonne:

- *Backlog*: raccolta iniziale delle attività proposte, non ancora pianificate;
- *Todo*: attività prioritarie selezionate per l'esecuzione;
- *In Progress*: attività in lavorazione;
- *In Review*: attività concluse e in fase di verifica;
- *Done*: attività completate.

Ogni *issue* è inoltre associata a un livello di priorità tra i seguenti:

- *Critical*: attività richiedenti intervento immediato;
- *High*: richieste ad alto impatto o con scadenza molto ravvicinata;
- *Medium*: *bugfix* non critici che vengono inseriti nella pianificazione ordinaria degli *Sprint*$_G$;
- *Low*: miglioramenti minori non prioritari.

La combinazione di colonna Kanban e priorità rende evidente sia lo stato di avanzamento sia l'urgenza.

[7] *Sito ufficiale Plane*. URL: https://plane.so/.

Figura 2.3: Kanban board

2.1.4 Strumenti di comunicazione

Le comunicazioni interne avvengono interamente tramite Telegram[8]: ogni progetto dispone di un gruppo dedicato in cui titolari e sviluppatori condividono aggiornamenti, documentazione e decisioni operative in tempo reale. Questo canale unico permette di evitare dispersioni e di mantenere traccia delle richieste provenienti dai clienti.

Le riunioni operative vengono organizzate con cadenza variabile, in funzione delle necessità del progetto o delle scadenze concordate con i clienti. Quando emergono nuove priorità o si avvicina un rilascio, i titolari convocano incontri mirati per allineare il team sugli obiettivi, discutere eventuali impedimenti e definire le attività da pianificare nel successivo *sprint*.

2.2 Gestione della sala stampe

Spazio Dev possiede attualmente molteplici punti di ingresso degli ordini relativi alla stampa 3D dei componenti, in particolare:

[8] *Telegram*. Sito ufficiale Telegram. URL: https://telegram.org (visitato il giorno 23/10/2025).

- La vendita di vasi portafiori e complementi d'arredo viene effettuata online e sulla piattaforma e-commerce Amazon;
- Le produzioni personalizzate e B2B vengono gestite manualmente.

Per la gestione di tali ordini e della coda di stampa viene utilizzato un gestionale sviluppato dall'azienda stessa, denominato "Idrotech Manager".

2.2.1 Ottimizzazione della coda di stampa

Per gestire in maniera efficiente l'evasione degli ordini l'azienda applica diverse ottimizzazioni alla coda di stampa, in modo tale da sfruttare nel miglior modo possibile le capacità di produzione. Tali procedure riguardano in particolare gli orari di avvio della stampa, la gestione dei cambi di materiale e dei cambi di ugello.

2.2.1.1 Orari di avvio della stampa

La gestione degli orari di avvio della stampa risulta fondamentale per ottimizzare al meglio la produzione. Ciò è dovuto al fatto che ogni stampante, quando termina la produzione di un piatto, necessita di un intervento manuale da parte di un operatore per la rimozione del prodotto finito e l'avvio della stampa successiva. L'idea alla base è quindi quella di pianificare tutte le stampe brevi durante l'orario lavorativo (in modo che sia sempre possibile avviare una nuova produzione). Le stampe più lunghe vengono invece avviate poco prima della fine della giornata per sfruttare al meglio i periodi di tempo in cui non c'è personale presente all'interno dello stabilimento.

2.2.1.2 Cambi di materiale

La coda di stampa può essere ottimizzata riducendo al minimo i cambi di materiale (o colore) tra una stampa e quella successiva. Questo perché ogni qual volta viene richiesto di cambiare il colore del filamento o il materiale utilizzato la stampa viene interrotta e un operatore deve intervenire manualmente per la sostituzione del filamento. Per semplicità, l'azienda assume che il tempo necessario per cambiare un filamento sia di 5 minuti.

2.2.1.3 Cambi ugello

Ogni articolo possiede uno specifico ugello che viene utilizzato per cambiare lo spessore della stampa. Come per i cambi di materiale anche il cambio ugello richiede un intervento manuale da parte di un operatore. Per semplicità, l'azienda assume che il tempo necessario per cambiare un filamento sia di 15 minuti.

Capitolo 3

Descrizione dello stage

3.1 Introduzione al progetto

Come descritto nell'introduzione (1) il progetto codificato durante il periodo di stage curricolare è uno schedulatore automatico che ottimizza e riordina la coda di stampa a partire dagli ordini di produzione presenti all'interno del software gestionale Idrotech Manager. Tale schedulatore deve tenere conto delle ottimizzazioni descritte nella sezione dedicata (2.2), oltre a doversi integrare con il gestionale in utilizzo dal personale.

3.2 Analisi preventiva dei rischi

La prima fase dello stage curricolare è stata dedicata all'analisi preventiva dei rischi. Tale procedura ha lo scopo di delineare i possibili rischi a cui si può andare incontro durante lo sviluppo del progetto, oltre a trovare delle possibili soluzioni per mitigare tali problematiche.

1. Errata scelta delle tecnologie

Descrizione: una scelta errata delle tecnologie da utilizzare per la codifica del progetto potrebbe portare ad un risultato inutilizzabile o non abbastanza performante.

Soluzione: coinvolgimento del responsabile nella scelta e occupare il primo periodo per validare le idee proposte.

2. Sicurezza nell'integrazione del modulo

Descrizione: la comunicazione tra il modulo da sviluppare e il gestionale esistente deve rispettare degli standard minimi di sicurezza, in modo che non tutti possano accedere alle *API* esposte.

Soluzione: dedicare parte del tempo allo studio su come rendere più sicuro l'accesso alle rotte *API*.

3. Risultati di ordinamento insoddisfacenti

Descrizione: il modulo potrebbe restituire risultati non soddisfacenti, con conseguenti inefficienze nella produzione.

Soluzione: validare assieme al tutor interno le logiche implementate in maniera periodica, testare il modulo con dataset realistici.

4. Prestazioni dello schedulatore

Descrizione: il modulo potrebbe impiegare troppo tempo per ottimizzare la coda di stampa.

Soluzione: utilizzare un *server di deploy*$_G$ con buone prestazioni, velocizzare l'esecuzione diminuendo, ad esempio, il tempo disponibile per la ricerca di una soluzione ottima.

5. Rispetto delle scadenze

Descrizione: le tempistiche dello stage curricolare potrebbero non essere sufficienti per avere un risultato concreto e utilizzabile.

Soluzione: svolgere una pianificazione (3.3) accurata del periodo di tempo a disposizione e convalidare il raggiungimento degli obiettivi.

3.3 Pianificazione

Oltre all'analisi preventiva dei rischi (3.2) è stata svolta una pianificazione accurata di tutte le fasi del periodo di stage. Questo, oltre a mitigare il rischio "Rispetto delle scadenze" (5) serve a determinare i contenuti da revisionare al termine di ogni *sprint*.

Settimana	**Obiettivi**	**Ore**
1	Analisi dei requisiti e dei rischi del progetto, scelta e inizio studio delle tecnologie.	40
2	Lettura della documentazione, studio del gestionale e progettazione delle integrazioni.	40
3	Inizio della codifica del progetto.	40
4	Codifica del progetto.	40
5	Codifica del progetto.	40
6	Codifica del progetto.	40
7	Codifica del progetto, validazione logiche implementate.	40
8	Test dell'algoritmo con dataset realistici.	20

Tabella 3.1: Tabella riassuntiva della pianificazione del periodo di stage.

Capitolo 4

Analisi dei requisiti

4.1 Casi d'uso

Come descritto nella sezione Pianificazione la prima parte del periodo di stage è stata dedicata all'analisi dei requisiti del progetto. Prima di fare ciò sono stati definiti tutti i *casi d'uso*$_G$ del sistema.

4.1.1 Attori individuati

L'utente finale del sistema non utilizzerà direttamente il modulo di pianificazione, le sue funzionalità potranno essere messe a disposizione attraverso il software gestionale, per questo sono stati individuati due principali attori:

- operatore, ovvero l'utente che interagisce con il software gestionale;
- software gestionale, il quale interagisce con il modulo di pianificazione.

Trattandosi di un sistema con il compito di automatizzare un processo, il numero dei *casi d'uso* risulta limitato.

4.1.2 Tabella dei casi d'uso

Identificativo	**Descrizione**
UC1	*Attori*: Gestionale. *Scopo*: Ricevere una lista di ordini da pianificare dal software gestionale. *Pre-condizione*: Endpoint raggiungibile, autenticazione valida e payload validato correttamente. *Post-condizione*: Richiesta accettata e messa in coda per l'elaborazione.
UC1.1	*Attori*: Gestionale. *Scopo*: Calcolare un piano di stampa coerente con i vincoli. *Pre-condizione*: Esiste una richiesta di schedulazione accettata. *Post-condizione*: Piano di schedulazione prodotto e pronto all'invio.
UC1.2	*Attori*: Gestionale. *Scopo*: Notificare al gestionale l'esito della schedulazione. *Pre-condizione*: Risultato disponibile, webhook del gestionale configurato e raggiungibile. *Post-condizione*: Esito notificato e inviato al gestionale.

Continua...

Continua...

Identificativo	**Descrizione**
UC1.E	*Attori*: Gestionale. *Scopo*: Comunicare un errore occorso in UC1 o UC1.1 *Pre-condizione*: Si è verificata un'anomalia (mancata autorizzazione, validazione dei dati fallita o altro). *Post-condizione*: Errore tracciato e inviato al gestionale.
UC2	*Attori*: Operatore. *Scopo*: Consultare l'elenco dei risultati di schedulazione. *Pre-condizione*: Schedulazioni precedenti presenti. *Post-condizione*: Schedulazioni mostrate.
UC2.1	*Attori*: Operatore. *Scopo*: Visualizzare il piano su vista a calendario. *Pre-condizione*: Almeno una schedulazione presente nel database. *Post-condizione*: Calendario visualizzabile a schermo con gli slot pianificati.

Continua...

Continua...

Identificativo	**Descrizione**
UC2.2	*Attori*: Operatore. *Scopo*: Confermare l'accettazione della coda proposta. *Pre-condizione*: Sezione dei dettagli della schedulazione aperta. *Post-condizione*: Schedulazione marcata come accettata e importazione avvenuta nel calendario.
UC2.E	*Attori*: Operatore. *Scopo*: Informare l'utente di errori avvenuti durante l'importazione a calendario. *Pre-condizione*: Errore occorso in UC2.2. *Post-condizione*: Errore comunicato e stato invariato.
UC3	*Attori*: Operatore. *Scopo*: Richiedere l'avvio della schedulazione automatica. *Pre-condizione*: Operatore autenticato, sezione Ordini accessibile. *Post-condizione*: Richiesta inviata al modulo e accettata.

Continua...

Continua...

Identificativo	Descrizione
UC3.1	*Attori*: Operatore. *Scopo*: Selezionare uno o più ordini da includere nella schedulazione. *Pre-condizione*: Modale di selezione ordini aperto. *Post-condizione*: Ordini da pianificare selezionati e pronti all'invio.
UC3.2	*Attori*: Operatore. *Scopo*: Informare dell'avvenuta accettazione della richiesta da parte del modulo. *Pre-condizione*: Risposta positiva del modulo. *Post-condizione*: Notifica visualizzata, modale chiuso.
UC3.E	*Attori*: Operatore *Scopo*: Informare l'utente di errori incontrati durante l'avvio della schedulazione. *Pre-condizione*: Errore occorso in UC3. *Post-condizione*: Errore comunicato e stato invariato.

4.1.3 Diagrammi dei casi d'uso

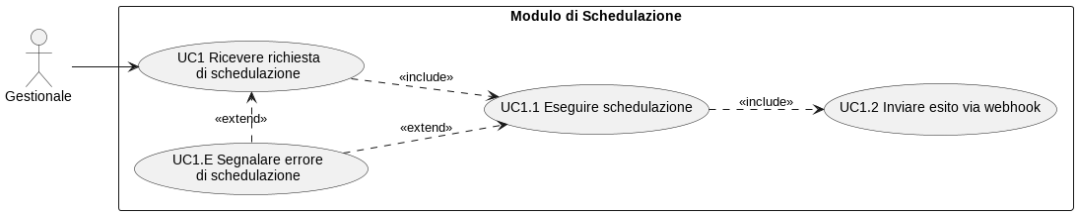


Figura 4.1: Diagramma dei casi d'uso UC1, UC1.1, UC1.2, UC1.E

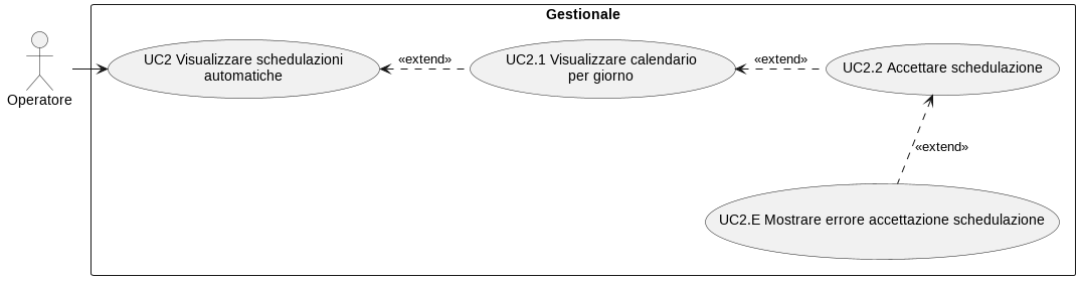


Figura 4.2: Diagramma dei casi d'uso UC2, UC2.1, UC2.2, UC2.E

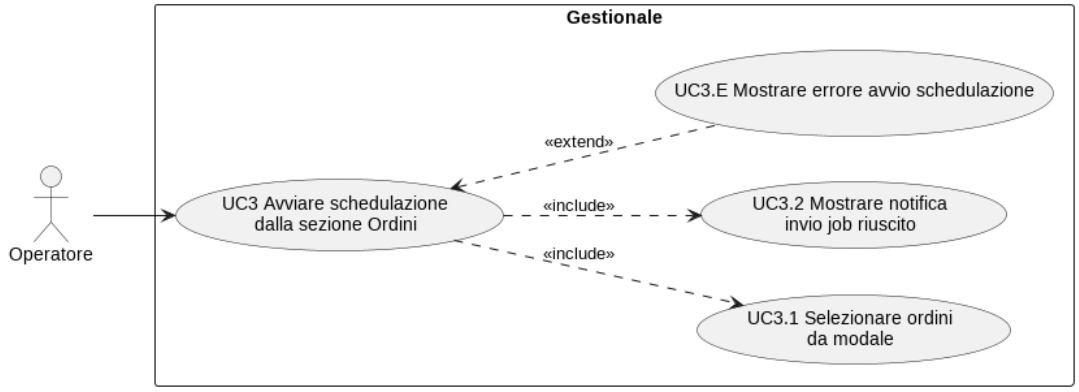


Figura 4.3: Diagramma dei casi d'uso UC3, UC3.1, UC3.2, UC3.E

4.2 Analisi dei requisiti

Una volta stilata la lista dei *casi d'uso* è stato possibile procedere con l'effettiva analisi dei requisiti del sistema. Tale operazione si è svolta attraverso un incontro con il tutor aziendale, durante il quale sono stati rilevati tutti i requisiti obbligatori, desiderabili e facoltativi.

4.2.1 Tracciamento dei requisiti

Ogni requisito viene identificato attraverso un codice identificativo di questo tipo:

R.x.y

dove:

- la lettera R identifica il requisito;
- la lettera x identifica la tipologia di tale requisito, ovvero:

  - O obbligatorio;
  - D desiderabile;
  - F facoltativo.

- la lettera y è un valore numerico progressivo a partire da 0.

4.3 Tabelle dei requisiti

4.3.1 Tabella dei requisiti obbligatori

Requisito	Descrizione
RO0	Il modulo deve produrre una coda di stampa dettagliata, evidenziando quale articolo stampare e quale stampante usare.
RO1	La schedulazione proposta deve considerare gli orari di lavoro aziendali.
RO2	La schedulazione proposta deve ottimizzare la stampa raggruppando i lavori con lo stesso materiale.
RO3	La schedulazione proposta deve ottimizzare la stampa pianificando i lavori corti durante la giornata lavorativa.
RO4	La schedulazione proposta deve ottimizzare la stampa pianificando i lavori lunghi al di fuori della giornata lavorativa.
RO5	La schedulazione deve essere avviabile dal software gestionale esistente.
RO6	La schedulazione deve essere visualizzabile dal software gestionale.
RO7	La schedulazione deve essere importabile nel calendario del software gestionale.
RO8	La schedulazione deve essere visibile in maniera chiara, in modo che l'utente possa trovare facilmente le informazioni utili.
RO9	La soluzione deve essere ammissibile e non devono esserci lavori sovrapposti.
RO10	Tutti i pezzi da stampare devono essere inseriti nella pianificazione, non sono ammesse stampe parziali.
RO11	Deve essere considerato un tempo di cambio piatto tra una stampa e l'altra.
RO12	Deve essere considerato un tempo di cambio filamento tra un lavoro di stampa e il suo successivo se usano materiali diversi.

Tabella 4.1: Tabella del tracciamento dei requisiti obbligatori.

4.3.2 Tabella dei requisiti desiderabili

Requisito	Descrizione
RD0	Deve essere disponibile un metodo di autenticazione per validare l'origine della richiesta delle schedulazioni.
RD1	Lo sviluppo deve avvenire utilizzando *design pattern*$_G$ comprovati e producendo codice mantenibile per facilitare le modifiche future.
RD2	La schedulazione proposta deve ottimizzare la stampa raggruppando i lavori con lo stesso ugello.
RD2	La schedulazione proposta deve ottimizzare la stampa raggruppando i lavori con lo stesso colore.
RD3	Al termine dello sviluppo l'algoritmo implementato deve essere validato utilizzando set di dati realistici.
RD4	La schedulazione proposta deve ottimizzare il numero di pezzi stampati nel piatto, minimizzando gli sprechi e il ritardo nelle consegne.

Tabella 4.2: Tabella del tracciamento dei requisiti desiderabili.

4.3.3 Tabella dei requisiti facoltativi

Requisito	Descrizione
RF0	La pianificazione deve avvenire automaticamente ad intervalli di tempo prefissati e modificabili dall'utente.
RF1	La pianificazione deve poter essere calcolata a partire da uno stato iniziale, schedulando solo i lavori mancanti.
RF2	L'utente deve essere in grado di modificare i parametri dello schedulatore dal software gestionale.

Tabella 4.3: Tabella del tracciamento dei requisiti facoltativi.

Capitolo 5

Progettazione e codifica

La fase di progettazione e ccodifica è stata la più significativa per il progetto, in quanto ha occupato la maggior parte del tempo a disposizione. Attraverso un incontro con il tutor aziendale sono stati definiti i *framework*$_G$ e le librerie da utilizzare per la codifica, inoltre sono state prese decisioni in merito all'architettura del sistema e ai *design pattern*$_G$ da utilizzare.

5.1 Tecnologie e strumenti

Di seguito viene data una panoramica delle tecnologie e strumenti utilizzati.

5.1.1 Linguaggi e framework

5.1.1.1 Python

Linguaggio di programmazione ad alto livello, orientato agli oggetti e adatto a sviluppare applicazioni distribuite. Viene utilizzato nello sviluppo web, nell'analisi dei dati, nel machine learning e nello sviluppo di automazioni.

5.1.1.2 FastAPI

Framework web per la codifica di *API* basato su Python[1]. Offre elevate prestazioni, validazione dell'input attraverso l'utilizzo della libreria Pydantic[2] e documentazione automatica seguendo lo standard OpenAPI[3].

5.1.2 Librerie

5.1.2.1 Black

Libreria che permette di formattare il codice Python, utilizzata per imporre un *code style*$_G$ unico e automatizzabile.

5.1.2.2 isort

Libreria che permette di formattare le importazioni delle librerie Python, utilizzata per imporre un *codestyle* unico e automatizzabile.

5.1.2.3 SQLModel

Libreria che permette di interagire con basi di dati relazionali attraverso il codice scritto in Python con la tecnica di programmazione *Object-Relational Mapping (ORM)*$_G$.

5.1.2.4 PyJWT

Libreria che permette di codificare e decodificare *JSON Web Token (JWT)*$_G$, utilizzati per autenticare utenti o servizi.

[1] *Python*. Sito ufficiale Python. URL: https://www.python.org/ (visitato il giorno 08/11/2025).

[2] *Pydantic*. Sito ufficiale Pydantic. URL: https://docs.pydantic.dev/latest/ (visitato il giorno 08/11/2025).

[3] *OpenAPI*. Sito ufficiale dcoumentazione OpenAPI. URL: https://swagger.io/specification/ (visitato il giorno 08/11/2025).

5.1.2.5 Pydantic

Libreria che aggiunge funzionalità legate alla validazione dei dati in Python, permette di definire con precisione la struttura dei dati e gestire i casi in cui si riceva un input non conforme alle aspettative.

5.1.2.6 Google OR-Tools

Libreria che contiene strumenti per risolvere problemi di *ottimizzazione combinatoria*$_G$. Permette di modellare un problema attraverso diversi linguaggi di programmazione (C++, Python, C# o Java) e utilizzare diversi risolutori commerciali o *open source*$_G$.

5.1.3 DevOps e server di deploy

5.1.3.1 Docker

Piattaforma utilizzata per creare ambienti isolati e riproducibili dove eseguire le applicazioni. Consente di standardizzare lo sviluppo locale e il *deploy*$_G$, riducendo differenze tra ambienti e semplificando l'esecuzione dei servizi.

5.1.3.2 Uvicorn

Server di tipo *Asynchronous Server Gateway Interface (ASGI)*$_G$ per applicazioni web Python asincrone. Utilizzato per eseguire l'*API* in sviluppo e in produzione, supporta la concorrenza.

5.1.4 Persistenza dei dati

5.1.4.1 MySQL

Sistema di gestione di basi di dati relazionali usato per la persistenza. Supporta transazioni e indici, ed è integrato con l'applicazione tramite *orm* per la definizione dello schema e l'accesso ai dati.

5.2 Scelte implementative

5.2.1 Progettazione della base di dati

5.2.1.1 Base di dati del modulo di pianificazione

Assieme al tutor aziendale è stata presa la decisione di non utilizzare la base di dati del modulo di pianificazione per la persistenza dei risultati generati, questo perchè la coda di stampa viene visualizzata direttamente all'interno del software gestionale e risulta pertanto più semplice e intuitivo modificare direttamente il *database* di quest'ultimo. La base di dati del modulo di pianificazione ha il solo compito di memorizzare i *client*$_G$ registrati e autorizzati a chiamare le *API* esposte, a tale scopo è stata creata una sola tabella denominata `clients` e così strutturata:

- `id`: intero auto-incrementale utilizzato come chiave primaria, generato automaticamente dal *database*.
- `client_id`: stringa obbligatoria indicizzata e univoca che identifica il client registrato.
- `client_secret`: stringa obbligatoria che contiene la password utilizzata dal client per l'autenticazione (vedi 5.3).
- `created_at`: campo di tipo `DateTime` per tracciare la data di registrazione.
- `is_active`: valore booleano che indica se il client è abilitato, è valorizzato a *true* di default.

Ciò permette, in futuro, di aggiungere eventuali altri *client* autorizzati ad utilizzare il modulo.

5.2.1.2 Base di dati del software gestionale

La progettazione della base di dati è risultata fondamentale per permettere all'utente di visualizzare correttamente le code di stampa pianificate. Il software

gestionale esistente (e già sviluppato al momento della codifica del progetto) possiede una base di dati già strutturata e utilizzata sia dall'azienda ospitante che da alcuni clienti dell'azienda stessa (in quanto il software è *multi tenant*$_G$). Per questo motivo l'ampliamento della base di dati esistente è stato eseguito cercando di mantenere al minimo il numero di modifiche.

Sono state aggiunte due tabelle per tracciare gli output della pianificazione e il legame con gli ordini:

- La tabella `scheduling_results` così formata:
  - `id`: intero auto-incrementale usato come chiave primaria;
  - `user_id`: chiave esterna verso la tabella `users` contenente i dati degli utenti registrati;
  - `data`: campo `json` obbligatorio che memorizza il payload completo della soluzione;
  - `is_accepted`: boolean obbligatorio valorizzato di default a *false* per indicare l'approvazione dell'utente;
  - `created_at`: campo di tipo `DateTime` per tracciare la data di ricezione del risultato;
  - `updated_at`: campo di tipo `DateTime` per tracciare la data dell'ultima modifica risultato.
- La tabella `scheduling_results_orders` così formata:
  - `id`: intero auto-incrementale usato come chiave primaria;
  - `order_id`: chiave esterna verso la tabella `orders` contenente gli ordini ricevuti;
  - `scheduling_result_id`: chiave esterna verso `scheduling_results`.

5.2.2 Design pattern adottati

Qui vengono descritti i *design pattern comportamentali* e di *inversione del controllo* adottati.

5.2.2.1 Architettura del sistema

Il *design pattern architetturale*$_G$ scelto per la codifica del progetto è il *pattern a layer*$_G$. Tale scelta è stata presa per le seguenti motivazioni:

- Permette una buona separazione delle responsabilità, ogni *layer* non conosce infatti l'implementazione degli altri;
- Permette una maggior semplicità di sviluppo, in quanto è un *design pattern architetturale* conosciuto e di semplice comprensione;
- Permette una buona manutenibilità del codice, ogni *layer* non viene influenzato dai cambiamenti apportati ad altri layer (a patto che i *contratti* tra i layer rimangano invariati);
- Permette una buona testabilità, ogni *layer* può essere isolato e testato separatamente;
- Il progetto non richiede una elevata *scalabilità*$_G$ del sistema.

I *layer* implementati all'interno del sistema sono i seguenti:

- *API layer*: rappresenta la logica che riceve le richieste HTTP e orchestra le chiamate ai servizi disponibili;
- *Business layer*: rappresenta la logica di business dell'applicazione, riceve i dati dall'*API layer* ed effettua le operazioni necessarie;
- *Persistence layer*: rappresenta la connessione tra il *business layer* e il *database*, si occupa di eseguire le *query* orchestrare l'utilizzo del *object-relational mapper*;
- *Database layer*: rappresenta il *database* dell'applicativo, dove i dati sono salvati e prelevati.

Ogni *layer* interagisce solo con i *layer* sottostanti, pertanto è stato adottato un approccio di tipo *closed layer*.

5.2.2.2 Inversione del controllo

È stato fatto utilizzo del sistema di *Dependency Injection* messo a disposizione dal *framework* FastAPI. In particolare il file jwt.py espone le rotte `/verifyToken` e `/token` e delega le operazioni di verifica e generazione dei *JSON Web Token* alle funzioni fornite dalla classe `JWTDependencies`. Tramite l'oggetto `Depends` il router richiede le dipendenze `get_verified_payload` per verificare il *token* e `get_new_token` per generarne uno nuovo, lasciando che FastAPI risolva e inietti gli oggetti necessari prima dell'esecuzione dell'endpoint. Questo approccio sposta all'esterno la creazione delle dipendenze, permettendo di sostituire facilmente l'implementazione durante i test o in fase di configurazione dell'applicazione.

```
#Import necessari
from fastapi import APIRouter, Depends

from app.dependencies.jwt.jwt_dependencies import
↪  jwt_dependencies

#Definizione di un router API
jwt = APIRouter()
#Rotta API per verificare il token
@jwt.post("/verifyToken")
async def check_token(payload: dict =
↪  Depends(jwt_dependencies.get_verified_payload)):
    return {"message": "Token verified", "payload": payload}
#Rotta API per ottenere un nuovo token
@jwt.post("/token")
async def get_token(payload: dict =
↪  Depends(jwt_dependencies.get_new_token)):
    return payload
```

Codice 5.1: File jwt.py

```python
class JWTDependencies:
    #Funzione che controlla il token e lancia un'eccezione se
    ↪  non è valido
    def get_verified_payload(
        self,
        credentials: HTTPAuthorizationCredentials =
        ↪  Security(HTTPBearer()),
    ) -> Dict[str, Any]:
        service = JWTService()
        success, result =
        ↪  service.verify_token(credentials=credentials)
        if not success:
            raise HTTPException(status_code=401,
            ↪  detail=result["message"])
        return result

    #Funzione per il rilascio di un nuovo token
    def get_new_token(
        self,
        client_id: str = Form(...),
        client_secret: str = Form(...),
        session: Session = Depends(db_dependencies.get_session),
    ) -> Dict[str, Any]:
        service = JWTService(session=session)
        success, result = service.generate_token(
            client_id=client_id, client_secret=client_secret
        )
        if not success:
            raise HTTPException(status_code=401,
            ↪  detail=result["message"])
        return result
```

Codice 5.2: Classe JWTDependencies

5.2.2.3 Design pattern comportamentali

È stato usato lo *strategy pattern* per definire una interfaccia per l'algoritmo di ordinamento. Tale scelta permetterà in futuro, se necessario, di riscrivere completamente la logica di schedulazione mantenendo la compatibilità con la restante logica del modulo, a patto che venga implementata la stessa interfaccia. La definizione viene fornita nel file `ordering_alg.py`.

```python
from abc import ABC, abstractmethod
from datetime import datetime
from typing import Dict, List, Optional, Tuple

from app.models import Job, Printer, SchedulingResult


class OrderingAlg(ABC):
    @abstractmethod
    def schedule_jobs(
        self,
        jobs: List[Job],
        printers: List[Printer],
        base_datetime: datetime,
        initial_schedule: Optional[SchedulingResult] = None,
        is_last_schedule: bool = False,
        future_capacity_by_order: Optional[Dict[int, int]] =
        ↪  None,
    ) -> Tuple[bool, SchedulingResult]:
        pass
```

Codice 5.3: Interfaccia schedule_jobs

La lista dei parametri accettati è la seguente:

- `jobs`: lista di lavori da pianificare;

- `printers`: lista di stampanti disponibili;
- `base_datetime`: data e ora di riferimento da cui far partire la schedulazione;
- `initial_schedule`: eventuale risultato già calcolato da integrare;
- `is_last_schedule`: valore booleano che indica se l'elaborazione corrente è l'ultima;
- `future_capacity_by_order`: dizionario opzionale con chiave l'identificativo dell'ordine e valore la capacità già riservata in pianificazioni future;

5.2.3 Struttura delle cartelle

5.3 Sicurezza del sistema

Capitolo 6

Verifica e validazione

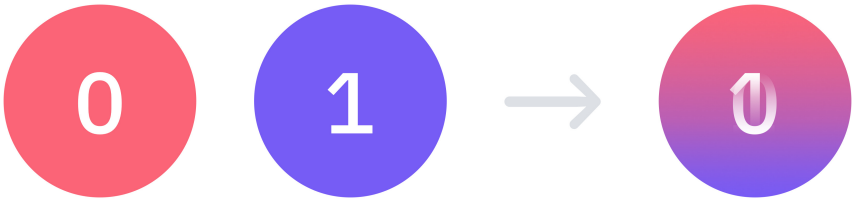


Figura 6.1: Lorem

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna,

vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Lorem ipsum:

```
def recur_fibo(n):
   if n <= 1:
       return n
   else:
       return(recur_fibo(n-1) + recur_fibo(n-2))
nterms = 10
if nterms <= 0:
   print("Plese enter a positive integer")
else:
   print("Fibonacci sequence:")
   for i in range(nterms):
       print(recur_fibo(i))
```

Codice 6.1: Fibonacci recursive

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Capitolo 7

Conclusioni

7.1 Consuntivo finale

Esempio di aggiunta di un termine con glossario e acronimo:

Lorem *Software Development Kit (SDK)*$_G$ ispum dolor.

Nel successivo utilizzo, apparirà solo l'acronimo:

Lorem *SDK*$_G$.

Nel caso si voglia invece mettere solo il termine per esteso, si può usare:

Lorem *Software Development Kit*$_G$.

7.2 Raggiungimento degli obiettivi

Esempio di termine con solo acronimo

Lorem *Termine solo acronimo (TSA)*$_G$, ipsum dolor sit amet

termine costruito senza acronimo: Lorem *Nome del termine*$_G$, ipsum dolor sit amet

7.3 Conoscenze acquisite

Lorem Ipsum dolor Lorem *Application Program Interface (API)*$_G$

Lorem Ipsum dolor Lorem *Application Program Interface*$_G$

Si può consultare il file *glossary_acronyms.tex* per alcuni esempi.

7.4 Valutazione personale

7.5 Valutazione personale

Bibliografia

Testi

James P. Womack, Daniel T. Jones. *Lean Thinking, Second Editon*. Simon & Schuster, Inc., 2010.

Articoli

Einstein, Albert, Boris Podolsky e Nathan Rosen. «Can Quantum-Mechanical Description of Physical Reality be Considered Complete?» In: *Physical Review* 47.10 (1935), pp. 777–780. DOI: 10.1103/PhysRev.47.777.

Sitografia

Documentazione ufficiale Conventional Commits 1.0.0. URL: https://www.conventionalcommits.org/en/v1.0.0/ (cit. a p. 6).

Framework Scrum. URL: https://it.wikipedia.org/wiki/Scrum_(informatica) (cit. a p. 7).

Framework Scrum. Licenza: CC BY-SA 4.0; l'immagine SVG originale è stata convertita in PNG per l'inclusione nel documento. URL: https://commons.wikimedia.org/wiki/File:Scrum_process.svg (visitato il giorno 23/10/2025).

Kanban board. Licenza: CC BY-SA 4.0; l'immagine SVG originale è stata convertita in PNG per l'inclusione nel documento. URL: https://commons.wikimedia.org/wiki/File:Abstract_Kanban_Board.svg (visitato il giorno 25/10/2025).

Manifesto Agile. URL: http://agilemanifesto.org/iso/it/ (cit. a p. 7).

OpenAPI. Sito ufficiale dcoumentazione OpenAPI. URL: https://swagger.io/specification/ (visitato il giorno 08/11/2025) (cit. a p. 25).

Pattern a layer. Spiegazione pattern a layer. URL: https://dev.to/yasmine_ddec94f4d4/understanding-the-layered-architecture-pattern-a-comprehensive-guide-1e2j#1-separation-of-concerns (visitato il giorno 08/11/2025).

Pydantic. Sito ufficiale Pydantic. URL: https://docs.pydantic.dev/latest/ (visitato il giorno 08/11/2025) (cit. a p. 25).

Python. Sito ufficiale Python. URL: https://www.python.org/ (visitato il giorno 08/11/2025) (cit. a p. 25).

Sito Mugalab. URL: https://mugalab.com/ (cit. a p. 1).

Sito Spazio Dev. URL: https://spaziodev.eu/ (cit. a p. 1).

Sito ufficiale Git. URL: https://git-scm.com/ (cit. a p. 5).

Sito ufficiale Gitea. URL: https://about.gitea.com/ (cit. a p. 5).

Sito ufficiale Plane. URL: https://plane.so/ (cit. a p. 8).

Spiegazione completa Gitflow workflow. URL: https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow (cit. a p. 5).

Telegram. Sito ufficiale Telegram. URL: https://telegram.org (visitato il giorno 23/10/2025) (cit. a p. 9).