Università degli Studi di Padova

Dipartimento di Matematica "Tullio Levi-Civita"

Corso di Laurea in Informatica

wCheckin: Ottimizzazione dell'autenticazione tramite identità digitale

Tesi di laurea

Relatore
Prof. Vardanega Tullio

Laureando
Skënderi Lisien

Anno Accademico 2023 - 2024

Sommario

Il presente lavoro si propone di creare un'applicazione per il check-in digitale, integrando tecnologie di punta fornite da Euronovate come firma digitale. Utilizzando ElectronJs, React e NodeJs (con addons in C++ compilati per NodeJs), l'applicazione mira a facilitare processi di check-in digitali in varie industrie. Lo scopo principale dell'applicazione è semplificare e ottimizzare il processo di check-in attraverso l'utilizzo di dispositivi elettronici, contribuendo così a migliorare l'efficienza operativa all'interno delle aziende.

Struttura del documento
Il documento è suddiviso in 4 capitoli principali:

Primo Capitolo - presentazione dell'azienda, i prodotti, gli strumenti e le tecnologie utilizzate, la loro integrazione e il rapporto con l'innovazione;

Secondo Capitolo - descrizione del progetto di stage, obiettivi, vincoli e pianificazione;

Terzo Capitolo - descrizione delle attività svolte e dei risultati ottenuti durante lo stage;

Quarto Capitolo - valutazione retrospettiva del progetto e competenze acquisite.

Al termine del documento sono presenti:

Acronimi - elenco degli acronimi utilizzati;

Glossario - elenco dei termini tecnici;

Bibliografia - elenco delle fonti consultate.

Convenzioni tipografiche
Le convenzioni tipografiche adottate sono:

- Parole straniere in *corsivo*;
- Parole chiave e termini in **grassetto**;
- Voci di elenchi terminano con un punto e virgola, tranne l'ultima con un punto; se la voce inizia con una parola chiave, il testo segue in minuscolo;
- Acronimi definiti come nota a piè di pagina alla prima occorrenza; i termini tecnici hanno una definizione approfondita nel glossario;
- Immagini numerate con fonti indicate come note a piè di pagina.

"Give me six hours to chop down a tree and I will spend the first four sharpening the axe"

— Abraham Lincoln

Ringraziamenti

Innanzitutto, vorrei esprimere la mia gratitudine al Prof. Vardanega Tullio, relatore della mia tesi, per il sostegno e la disponibilità forniti a partire dalla scelta dello stage fino alla stesura della tesi.

Ringrazio il mio tutor aziendale, Sig. Matteo Gnoato, per la pazienza e la disponibilità dimostrate durante il mio stage.

Desidero ringraziare con affetto i miei genitori per il sostegno, il grande aiuto e per essermi stati vicini in ogni momento durante gli anni di studio.

Voglio anche ringraziare i miei amici per tutti i bellissimi anni passati insieme e le mille avventure vissute. "In particolare, ringrazio mio fratello, il mio migliore amico, per essere sempre stato al mio fianco e per avermi sostenuto in ogni momento."

Padova, Luglio 2024 Skënderi Lisien

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Presentazione dell'azienda

In questo capitolo descrivo l'azienda dove ho svolto lo stage, ***Euronovate Group***. In questo capitolo descrivo l'organizzazione dell'azienda, i suoi processi e le metodologie di lavoro adottate, i prodotti e i servizi offerti, gli strumenti e le tecnologie utilizzati e, infine, l'innovazione offerta dall'azienda. Tutto ciò che descrivo in questo capitolo l'ho appreso osservando l'azienda durante il periodo di stage e ricercando informazioni in modo autonomo.

1.1 L'azienda

1.1.1 Euronovate SA

Fondata nel 2012 e con sede a Lugano (CH), è una società svizzera innovativa, specializzata in soluzioni di ***Digital Trasformation*** con approccio *end-to-end*, combinando soluzioni *software*, *hardware* e servizi di consulenza. L'obiettivo principale è aiutare ogni tipo di azienda ad eliminare tutti i processi e i documenti cartacei passando completamente al digitale, garantendo la stessa validità legale.

1.1.2 eSignWorld

Società che opera al settore della consulenza IT[1], processi e sistemi avanzati di firme elettroniche, fornitura, esercizio e manutenzione di sistemi informativi *hardware* e *software*, specializzata nel campo della produzione a filiera corta di tecnologia grafometrica[2]. In particolare, *eSignWorld* fornisce soluzioni personalizzate e proprietarie nel campo della dematerializzazione documentale e dell'textitInformation Communication Technology, garantendo la possibilità di visualizzare, elaborare e firmare elettronicamente qualsiasi tipo di documento. *eSignworld* offre soluzioni *paperless* con utilizzo di firma elettronica semplice e firma elettronica avanzata, un sistema di composizione e successiva conservazione di documenti in formato elettronico, nonché di firma dei documenti, attraverso un innovativo dispositivo di firma.

[1] Information Technology (IT)
[2] Grafometrica

1.2 Prodotti e servizi

Figura 1.1: Imagine di *ENSign11 NFC*[3]

Euronovate Group offre una serie di prodotti e servizi, tra cui ci focalizzero su *ENSign11 NFC*1.1, un dispositivo di firma elettronica con un modulo NFC[4] integrato che consente la lettura delle carte d'identità, e ENSignSuite, una collezione di software come:

- ***Software***:
  - ***ENSoft***: è un'applicazione desktop utilizzato per effettuare la firma elettronica di documenti, usando i tablet di firma;
  - ***ENSmartSign***: è un'applicazione che permette di firmare documenti senza il bisogno di modificare il documento;
  - ***ENCalligrapher***: è un'applicazione desktop nato per l'esigenza di assicurare e garantire la titolarità delle firme elettroniche;

[3] Fonte: https://www.euronovategroup.com
[4] Near Field Communication

Figura 1.2: Soluzione *ENSmartSign*[5]

- ***Hardware***:
  - ***Tablet di Firma***: dispositivi come ENSign11 e ENSign11 NFC, che permettono di firmare documenti in modo sicuro e legale, usando i software1.2 sopra citati;
  - ***Scanner***: dispositivi come ENScan, che permettono di digitalizzare documenti in modo rapido e preciso, per archiviarli e condividerli in formato elettronico, togliendo la necessità di documenti cartacei;

1.3 Organizzazione aziendale

L'azienda è organizzata in diversi dipartimenti, ognuno dei quali si occupa di un aspetto specifico del funzionamento aziendale. I dipartimenti principali sono:

- **Direzione**: responsabile delle decisioni strategiche e della supervisione complessiva dell'azienda. Coordina le attività dei vari dipartimenti per assicurare il raggiungimento degli obiettivi aziendali;
- **PM**[6]: gestisce i progetti aziendali, assicurando che vengano completati nei tempi e nei budget stabiliti. Monitora l'avanzamento dei progetti e risolve eventuali problemi che possono sorgere. Comunica con i clienti per garantire che le loro esigenze siano soddisfatte e con il team di sviluppo per garantire che i requisiti del progetto siano rispettati;
- **Risorse umane**: si occupa del reclutamento e della gestione del personale, garantendo che l'azienda abbia le risorse necessarie per raggiungere i suoi obiettivi;

[5] Fonte: https://www.euronovate.com/
[6] Project Manager (PM)

- **Marketing e vendite**: responsabile della promozione dei prodotti e dei servizi dell'azienda, sviluppando strategie di marketing efficaci per raggiungere i clienti. Collabora con il dipartimento di vendite per generare lead e aumentare le vendite;
- **Sistemisti**: si occupa della gestione e manutenzione dei sistemi informatici e delle infrastrutture tecnologiche dell'azienda. Assicura che i sistemi siano sempre operativi e sicuri, e che i dati siano protetti da minacce esterne;
- **Sviluppo**: si occupa della progettazione e dello sviluppo di nuovi prodotti o servizi. Collabora con il dipartimento di QA[7] per assicurare che i prodotti soddisfino gli standard di qualità. Viene guidata dal PM o dal *Team Leader*;
- **QA**: garantisce la qualità dei prodotti e dei servizi attraverso controlli e test rigorosi. Lavora per migliorare continuamente i processi produttivi e ridurre i difetti. Collabora strettamente con il team di sviluppo per garantire che i prodotti soddisfino gli standard di qualità;
- **Supporto**: fornisce assistenza tecnica e supporto ai clienti post-vendita. Gestisce anche il servizio clienti e risponde alle richieste di assistenza. Fornisce assistenza anche al personale interno per risolvere eventuali problemi tecnici;

Ogni dipartimento dell'azienda interagisce costantemente con gli altri per garantire il buon funzionamento e il raggiungimento degli obiettivi comuni. La direzione coordina le attività, il *PM* collabora con sviluppo e *QA* per rispettare qualità e scadenze, e le risorse umane assicurano le competenze necessarie. *Marketing* e vendite promuovono nuovi prodotti, mentre i sistemisti mantengono le infrastrutture tecniche operative.

1.4 Processi interni

L'azienda usa come metodologia per il ciclo di vita dei *software* il metodo *Agile*. Il lavoro è organizzato in *sprint*, che sono periodi di tempo definiti in cui vengono pianificate e svolte attività specifiche. Gli *sprint* hanno una durata variabile, che può essere al massimo di due settimane o al minimo di una settimana.

I principali processi interni dell'azienda sono i seguenti:

- **Sviluppo**:
  - **Analisi dei requisiti**: quest'attività prevede la raccolta e l'analisi dettagliata dei requisiti del sistema, per assicurarsi che il software soddisfi le esigenze degli utenti finali. Di solito si fa proposta da un cliente o da i *stakeholder*, e vengono tradotte in specifiche funzionali e non funzionali;
  - **Progettazione**: in questa tappa, si definisce l'architettura del software, scegliendo le tecnologie più adatte e progettando il sistema in modo che soddisfi i requisiti identificati nell'attività di analisi. Si definiscono anche i moduli e i componenti del sistema, e le interfacce tra di essi;

[7] Quality Assurance

  - **Codifica**: durante la codifica, i programmatori scrivono il codice sorgente del software, implementando le funzionalità definite nelle attività precedenti secondo le migliori pratiche di programmazione.

- **QA**:
  - **Pianificazione dei test**: prima di iniziare lo sviluppo, si pianificano le attività di test per assicurare che ogni parte del software venga verificata e validata secondo gli standard di qualità;
  - **Esecuzione dei test**: i test vengono eseguiti sistematicamente per identificare malfunzionamenti, errori o difetti nel software. Questo include test di unità, integrazione, sistema e accettazione;
  - **Risoluzione dei problemi**: i problemi rilevati durante i test vengono documentati e risolti in collaborazione con il team di sviluppo, per garantire che il prodotto finale soddisfi i requisiti di qualità;
  - **Assicurazione della qualità Continua**: la QA monitora costantemente i processi di sviluppo e test per migliorare continuamente la qualità del software e prevenire la ricorrenza di errori noti.

Figura 1.3: Processo di QA[8]

- **Manutenzione e supporto**:
  - **Supporto post-rilascio**: dopo il rilascio del software, viene fornito supporto continuo ai clienti per risolvere eventuali problemi e rispondere a domande relative all'uso del prodotto;
  - **Gestione dei bug**: monitoraggio e tracciamento dei bug segnalati dagli utenti attraverso un sistema di gestione dei ticket, con priorità assegnata in base alla gravità e all'impatto sul cliente;

[8]Fonte: https://www.testbytes.net

- **Aggiornamenti e patch**: sviluppo e distribuzione di aggiornamenti e patch per correggere i bug rilevati, migliorare le funzionalità esistenti e garantire la sicurezza del software;
- **Miglioramento continuo del codice**: revisione periodica del codice per identificare aree di miglioramento, implementare best practices e refactoring per migliorare la qualità e la manutenibilità del software;
- **Feedback dei clienti**: raccolta e analisi del feedback dei clienti per capire le loro esigenze e aspettative, utilizzando queste informazioni per pianificare miglioramenti e nuove funzionalità.

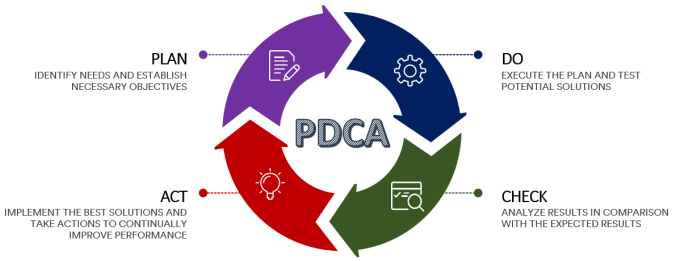

Figura 1.4: Ciclo di Miglioramento Continuo[9]

Tutti i processi interni sono collegati strettamente per garantire il miglioramento continuo 1.4 del prodotto e la soddisfazione del cliente. Il *team* di sviluppo collabora con il team di *QA* per garantire che il software soddisfi gli standard di qualità, mentre il team di supporto fornisce assistenza ai clienti post-vendita per risolvere eventuali problemi e garantire un'esperienza utente positiva. Il *feedback* dei clienti viene utilizzato per guidare il miglioramento continuo del prodotto, identificando aree di miglioramento e nuove funzionalità da implementare.

1.5 Strumenti e tecnologie

L'azienda utilizza una serie di strumenti e tecnologie integrati nei suoi flussi di lavoro per supportare i processi interni e lo sviluppo dei prodotti. Di seguito, una panoramica su come questi strumenti si integrano e contribuiscono all'efficienza aziendale:

- **Sistema operativo**: l'azienda utilizza Windows come sistema operativo principale per lo sviluppo e il testing del software. Nei anni recenti l'azienda ha iniziato a utilizzare anche Linux(Ubuntu) per alcuni progetti, *cross-platform* e per la gestione dei server;
- ***Atlassian***: l'azienda utilizza la suite di strumenti Atlassian per la gestione dei progetti e delle attività, la documentazione e la collaborazione tra i team. Questi strumenti sono Jira per la gestione dei progetti e come un sistema

[9] Fonte: https://www.linkedin.com

di tracciamento degli incidenti (ITS)[10], Confluence per la documentazione, BitBucket per il controllo delle versioni;

- **CI/CD**: l'azienda utilizza *Jenkins* e *Travis CI* come strumenti per l'integrazione continua e la distribuzione continua del *software*. Questo strumento automatizza i processi di test e build, garantendo che ogni modifica al codice venga verificata incrementamene e che la versione rilasciata sia stabile e funzionante. Questo approccio permette il rilasciato in modo efficiente nelle apposite *repository* dei artefatti;

- ***Artifact repository***: per la gestione dei *repository* di artefatti, *Artifactory* facilita la distribuzione dei pacchetti e la condivisione delle librerie create dall'azienda. Questo garantisce che i componenti software siano facilmente accessibili e riutilizzabili dai clienti ma anche dai team di sviluppo;

- **Distribuzione dei container**: *Docker* e *Kubernetes* sono utilizzati congiuntamente per la creazione, distribuzione, esecuzione e gestione di container software. *Docker* permette di standardizzare gli ambienti di sviluppo e produzione, migliorando la portabilità delle applicazioni. *Kubernetes* assicura la scalabilità, l'affidabilità e la sicurezza dei servizi, orchestrando le applicazioni distribuite usando dei *pods*, garantendo che le applicazioni siano sempre disponibili e funzionanti;

- **Mezzo di comunicazione**: come mezzo di comunicazione interna ma anche esterna, l'azienda utilizza *Discord* per la comunicazione tramite messaggi formali e informali tra i dipendenti, *Gmail* per la comunicazione tramite mail formali con i clienti e internamente, mentre per le riunioni online utilizza *Google Meet*.

L'integrazione degli strumenti e delle tecnologie menzionate permette all'azienda di operare in modo sinergico ed efficiente. *Windows* e *Linux* forniscono basi solide per sviluppo e testing, mentre la suite *Atlassian* (*Jira*, *Confluence*, *BitBucket*) garantisce una collaborazione fluida tra i team. *Jenkins* e *Travis CI* automatizzano test e build, migliorando qualità e velocità del rilascio del *software*. *Artifactory* gestisce i repository di artefatti, facilitando distribuzione e condivisione. *Docker* e *Kubernetes* standardizzano gli ambienti e orchestrano le applicazioni, assicurando scalabilità e affidabilità. *Discord*, *Gmail* e *Google Meet* supportano la comunicazione interna ed esterna, mantenendo tutte le parti aggiornate e connesse. Questa combinazione di strumenti e tecnologie crea un ecosistema interconnesso che supporta tutte le attività di tutti i processi del ciclo di vita del *software*, contribuendo all'efficienza aziendale.

1.6 Innovazione

L'azienda ha una vocazione per l'innovazione tecnologica e nella trasformazione digitale. Inoltre, si impegna a offrire soluzioni in diversi settori, tra cui:

[10] Issue Tracking System (ITS)

- **Tecnologia di *face-matching***: l'azienda sviluppa tecnologie di riconoscimento facciale per l'identificazione e l'autenticazione degli utenti. Queste tecnologie usa algoritmi di intelligenza artificiale per verificare l'identità di una persona, aumentando la sicurezza e semplificando l'accesso a servizi e applicazioni;
- **Tecnologie di firma elettronica**: sono sviluppati sistemi avanzati di firma elettronica, che includono la firma elettronica semplice e avanzata. Queste tecnologie garantiscono la validità legale dei documenti firmati e semplificando il workflow aziendale;
- **Sostenibilità ecologica**: l'azienda è anche orientato alla sostenibilità ecologica, promuovendo tecnologie paperless che riducono drasticamente l'uso della carta e l'impatto ambientale. Le soluzioni paperless non solo migliorano l'efficienza operativa delle aziende, ma facilitano anche l'archiviazione e la gestione dei documenti in formato elettronico;
- **Stage**: l'azienda offre stage a studenti non solo per formare nuove risorse, ma anche per ricevere nuove idee e proposte innovative. Questo permette di sperimentare con nuove idee e tecnologie per migliorare i prodotti e i servizi offerti.

Capitolo 2

Lo stage

Questo capitolo descrive la strategia aziendale di *Euronovate* riguardante gli stage, il progetto che l'azienda ha presentato, i suoi obiettivi, i vincoli e la pianificazione. Inoltre, spiego i motivi che mi hanno spinto a scegliere questo stage tra le varie proposte."

2.1 Strategia aziendale

Euronovate è un'azienda che ha partecipato quasi ogni anno all'evento di STAGE-it, organizzato dall'Università degli Studi di Padova, con l'obiettivo di offrire opportunità di stage a studenti universitari tramite colloqui tra studenti e aziende in presenza.

Le strategie di *Euronovate* in relazione agli stage includono:

- **Integrazione con i prodotti/servizi aziendali**: utilizzare gli stage per sviluppare progetti che integrano le tecnologie e soluzioni offerte dall'azienda. Gli studenti lavorano su applicazioni che possono essere direttamente incorporate nei prodotti e servizi di *Euronovate*, migliorando l'offerta aziendale;
- **Sperimentazione con nuove tecnologie**: utilizzare gli stage come opportunità per testare e sperimentare nuove tecnologie. Gli studenti possono esplorare soluzioni innovative e contribuire all'adozione di nuove tecnologie all'interno dell'azienda;
- **Creazione di nuovi prodotti**: offrire progetti di stage orientati alla creazione di nuovi prodotti o alla significativa innovazione di quelli esistenti. Questo approccio consente all'azienda di sviluppare nuove idee e trasformarle in soluzioni pratiche, beneficiando della creatività degli studenti;
- **Valutazione e sviluppo delle competenze**: utilizzare gli stage per valutare le competenze degli studenti e contribuire al loro sviluppo professionale. Gli stagisti hanno l'opportunità di lavorare su progetti reali, sviluppando competenze pratiche e acquisendo esperienza nel settore;
- **Rimanere aggiornati e sperimentare**: l'azienda ha così un modo di rimanere aggiornata e sperimentare con le nuove tendenze e tecnologie emergenti, e

allo stesso tempo ha potuto entrare in contatto con studenti interessati ad apprendere nuove competenze.

2.2 Premessa e obiettivi

L'idea del progetto di stage nasce dall'esigenza crescente di digitalizzare i processi di identificazione e gestione documentale in vari settori. Con l'avanzamento tecnologico e l'adozione di soluzioni innovative, molti mercati stanno implementando sistemi di gestione elettronica dei documenti e firme digitali. Queste tecnologie sono fondamentali per migliorare l'efficienza e la sicurezza delle operazioni quotidiane.

Figura 2.1: Soluzione *ENSignSoft*[11]

L'azienda aveva ideato un progetto che sfrutterà le tecnologie di firma digitale (ENSignSoft)2.1 una soluzione di firma digitale e i tablet di firma (ENSign11 NFC)1.1 di *Euronovate* per creare un sistema di *check-in* digitale, salvando i documenti digitali e le firme dei visitatori, l'azienda potrà gestire in modo più efficace le informazioni e i dati dei propri ospiti. Il progetto è un applicazione *fullstack*[12], quindi è composta da un *frontend*[13] e *backend*[14] che comunicano tra di loro usando REST[15] Api[16]. Il *frontend* è composto di 2 UI[17] diverse una per il tablet di firma, che sarà usato dai visitatori, e una per il *desktop*, che sarà usato dagli operatori. Il *backend* è composto da un server che si occupa di gestire le richieste del *frontend*. Gli obiettivi dello stage erano:

- **Sviluppo di un'applicazione desktop**: sviluppare un'applicazione desktop che permetta a un'operatore di gestire il processo di *check-in*, aiutando il

[11] Fonte: https://www.euronovategroup.com
[12] FullStack
[13] Front-end
[14] Back-end
[15] Representational State Transfer
[16] Application Program Interface
[17] User Interface

visitatore a compilare il documento per del *check-in*, permettendo di apportare la firma sia al'ingresso che in uscita;

- **Sviluppo di un'applicazione per il tablet**: sviluppare un'applicazione per il tablet che permetta ai visitatori di compilare i dati necessari usando la NFC, o inserendo i dati manualmente nel form, cosi per compilare e firmare i documenti;
- **Integrazione con i dispositivi di firma**: integrare l'applicazione con i dispositivi di firma e il software di *Euronovate*, in modo da poter utilizzare la firma digitale per sigillare i documenti;
- **Gestione dei dati**: gestire i dati dei visitatori in modo sicuro e conforme alle normative sulla *privacy*;

2.3 Vincoli

Durante lo stage, il tutor aziendale ha definito alcuni vincoli che hanno guidato lo sviluppo del progetto. I vincoli principali sono:

- **Tecnologie**: come linguaggio di programmazione, dovevo utilizzare TypeScript[18] per scrivere l'intero stack tecnologico 2.2, tranne la parte del database. Per le altre componenti:
  - **Frontend**: il *frontend* doveva essere sviluppato utilizzando React[19] per la *UI* ed Electron[20] per effettuare il *rendering* della *UI*;
  - **Backend**: il *backend* doveva essere sviluppato con Node.js[21] come ambiente di runtime per il lato *server*, utilizzando Express[22] come framework[23] per la gestione delle richieste *HTTP*;
  - **Database**: il *database* doveva essere sviluppato con PostgreSQL[24];
  - **API**: infine per la comunicazione tra il *frontend* e il *backend*, era necessario utilizzare il protocollo REST API;

[18] TypeScript: https://www.typescriptlang.org/
[19] React: https://reactjs.org/
[20] Electron: https://www.electronjs.org/
[21] Node.js: https://nodejs.org/
[22] Express: https://expressjs.com/
[23] Framework
[24] PostgreSQL: https://www.postgresql.org/

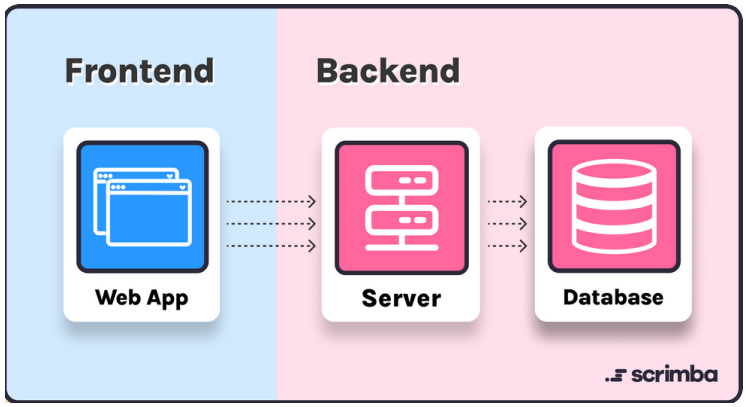

Figura 2.2: Comunicazione tra *frontend* e *backend*[25]

- ***UI design***: per il *design* dell'interfaccia sia *desktop* sia *tablet*, era necessario seguire le linee guida aziendali che includono il logo aziendale, i colori, i bottoni e lo stile di input dei dati;
- **Documentazione**: infine, era richiesto di scrivere documenti tecnici e di presentazione per descrivere il progetto e i risultati ottenuti, al fine di garantire la tracciabilità delle attività svolte e la riproducibilità dei risultati ottenuti. Inoltre, doveva essere redatto un manuale di installazione e manutenzione per permettere a chiunque di installare e mantenere il prodotto.

2.4 Pianificazione

La pianificazione dello stage è stata organizzata in fasi distinte. Ho rispettato la pianificazione definita all'inizio dello stage, seguendo le scadenze e gli obiettivi prefissati. La pianificazione è stata organizzata in prima del inizio dello stage, e ha previsto le seguenti fasi:

Preparazione degli strumenti	Task	1 giorno	29/04/2024	29/04/2024
Formazione Personale	Task	7 giorni	30/04/2024	08/05/2024
Analisi dei Requisiti	Task	4 giorni	09/05/2024	14/05/2024
Progettazione Tecnica	Task	3 giorni	15/05/2024	17/05/2024
Codifica	Task	29 giorni	20/05/2024	27/06/2024
Documentazione	Task	1 giorno	28/06/2024	28/06/2024
Demo (opzionale)	Task	1 giorno	01/07/2024	01/07/2024

Figura 2.3: Pianificazione dello stage

- **Preparazione degli strumenti**:
  - Installazione ambienti di sviluppo e di versionamento;

[25] Fonte: https://scrimba.com

  - Abilitazione strumenti aziendali (*account git*, indirizzo mail, spazio *google drive*).

- **Formazione personale**:
  - Approfondimento strumenti di sviluppo e di versionamento del codice;
  - Approfondimento dello sviluppo desktop con ElectronJs;
  - Tecniche di comunicazione *interprocess* per il passaggio dei dati tra diverse tecnologie;
  - Approfondimento sull'uso di React per la realizzazione delle interfacce grafiche.
- **Analisi dei requisiti**;
  - Analisi del processo funzionale;
  - Approfondimento con il DPO[26] per la gestione dei dati sensibili;
  - Stesura dei casi d'uso e dei requisiti.
- **Progettazione tecnica**:
  - Architettura generale dell'applicativo *client-server*;
  - Definizione dei servizi *backend*.
- **Codifica e *deployment***:
  - Sviluppo seguendo le norme di codifica aziendali;
  - Realizzazione dei *dockerfile*.
- **Documentazione**:
  - Stesura del manuale di manutenzione;
  - Stesura del manuale di installazione.
- **Demo (opzionale)**

L'attività di codifica è divisa in 6 sprint, ognuno della durata di 1 settimana. Ogni sprint prevede i seguenti compiti:

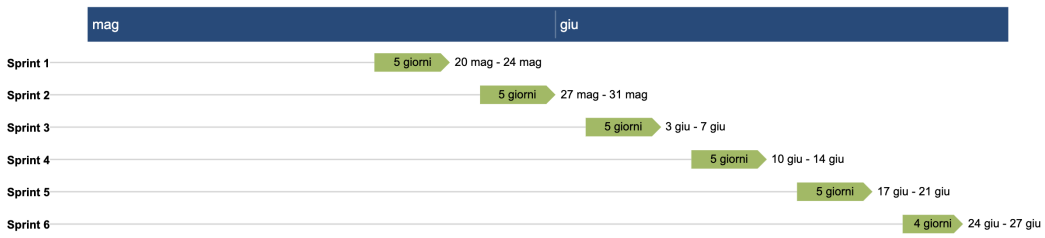

Figura 2.4: Organizzazione degli sprint

[26] Data Protection Officer

- ***Sprint* 1** : struttura generale del progetto, realizzazione dei servizi backend per il salvataggio e il recupero dei dati. scrittura componente per l'integrazione del modulo NFC;
- ***Sprint* 2** : realizzazione della UI desktop;
- ***Sprint* 3** : realizzazione della UI sul device di firma. Gestione della fotocamera del device e chiamate ai servizi di facematch e liveness. Gestione degli errori in caso di fallimento;
- ***Sprint* 4** : realizzazione del template e gestione dei dati per la compilazione automatica del documento PDF da firmare;
- ***Sprint* 5** : realizzazione del flusso di firma digitale per sigillare con il timestamp il documento firmato;
- ***Sprint* 6** : scrittura del dockerfile, deploy della parte backend su AWS, consolidamento della soluzione, aumento della qualità complessiva con la gestione degli errori e le migliorie di UI.

2.5 Scelta dello stage

I motivi che mi hanno spinto a scegliere questo stage tra le varie proposte sono:

- **Introduzione nel mondo lavorativo**: prima dello stage non avevo mai avuto esperienze lavorative, quindi lo stage mi ha permesso di entrare in contatto con il mondo del lavoro. Ho avuto l'opportunità di creare nuove relazioni professionali e di lavorare con persone più esperte, imparando nuove competenze e acquisendo nuove conoscenze;
- **Tecnologia fullstack**: durante i colloqui avuti con stage-it, quando mi veniva chiesto quale fosse il mio interesse tra *frontend* e *backend*, non sapevo rispondere. Questo stage mi ha permesso di lavorare su entrambi i lati, sviluppando un'applicazione *fullstack* che integra il *frontend* e il *backend*. Ho avuto l'opportunità di capire meglio cosa mi piace fare e di approfondire le tecnologie fullstack;
- **Stage che produce un prodotto finale**: tra i vari stage, volevo sceglierne uno che mi permettesse di sviluppare un prodotto finale, in modo da poter vedere i risultati del mio lavoro. Questo stage mi ha permesso di sviluppare un'applicazione che può essere utilizzata in ambito reale, permettendomi di vedere i risultati del mio lavoro e di capire come le mie azioni possono influenzare il prodotto finale.

Infine, avendo queste motivazioni, ho fissato alcuni obiettivi personali che volevo raggiungere, per lo stage che avevo scelto:

- Imparare lavorare in un team, collaborando con altri sviluppatori e condividendo conoscenze e competenze;
- Sviluppare un'applicazione *fullstack* che integra il *frontend* e il *backend*;

- Sviluppare un *backend* che comunica con il frontend usando REST API;
- Creare prodotto finale che può essere utilizzato in ambito reale;
- Imparare nuove tecnologie come React, Electron e NodeJs.

Capitolo 3

Svolgimento dello stage

Questo capitolo descrive il modo in cui ho lavorato durante lo stage e le interazione con il tutor aziendale. Si racconta le attività svolte, come l'analisi dei requisiti, la progettazione, la codifica e infine il risultato ottenuto.

3.1 Modalità operative

3.1.1 Modo di lavoro

Con una pianificazione 2.4, abbiamo concordato riunioni quasi giornaliere, tranne durante l'attività di codifica, per discutere lo stato di avanzamento del lavoro e chiarire eventuali dubbi. Durante l'attività di codifica, il tutor aziendale ha suddiviso il lavoro in sei *sprint*, ciascuno della durata di una settimana. Per ogni *sprint*, il tutor aziendale ha fissato una riunione settimanale per definire i task da svolgere e una riunione finale per discutere e valutare il lavoro svolto e il rendimento.

Durante lo stage, per qualsiasi dubbio o problema che non riuscivo a risolvere da solo, potevo contattare il tutor aziendale. Se il problema riguardava i *software* o l'*hardware* forniti dall'azienda, potevo porre domande anche ai colleghi che avevano contribuito alla creazione o che avevano esperienza con l'utilizzo di tali strumenti.

3.1.2 Strumenti utilizzati

Durante lo svolgimento dello stage, ho utilizzato diversi strumenti per la comunicazione, la gestione del codice sorgente e la gestione dei task. Di seguito sono elencati gli strumenti principali utilizzati:

- ***Software* di sviluppo**
  Gli strumenti di sviluppo sono stati fondamentali per la scrittura del codice, il testing delle *API*, la gestione dei container e il controllo della versione del codice:
  - ***Visual Studio Code***: l'ho utilizzato questo ambiente di sviluppo integrato (IDE)[27] per scrivere il codice sorgente, grazie alla sua flessibilità e

[27] Integrated Development Environment

alle numerose estensioni disponibili, che vienne integrato perfettamente con *Git* per il controllo della versione;

- ***Postman***: l'ho utilizzato per testare le *API* sviluppate in *Express*, un'strumento essenziale per garantire che le comunicazioni tra frontend e backend fossero corrette;
- ***Docker desktop***: usato per creare, gestire e distribuire container, facilitando la portabilità e l'isolamento dell'applicazione, specialmente durante i test di integrazione e distribuzione;
- ***Git***: sistema di controllo versione che ha permesso di tracciare le modifiche al codice e collaborare con altri sviluppatori, integrandosi con *Visual Studio Code* e *Bitbucket*;
- ***Bitbucket***: servizio di hosting per progetti che utilizzano *Git*, utile per la gestione centralizzata del codice sorgente e la collaborazione in un *team*;
- ***Figma***: utilizzato nella creazione di alcuni risorse grafiche utilizzate nell'interfaccia utente;
- ***StarUML***: utilizzato per creare diagrammi UML[28], fornendo una chiara rappresentazione visiva delle strutture e delle relazioni del software, che sono state implementate nel codice. I diagrammi sono stati utilizzati nella documentazione del analisi dei requisiti e della progettazione;
- ***SDK***: gli SDK[29] forniti dall'azienda sono stati utilizzati per integrare e utilizzare prodotti specifici dell'azienda, come *ENSoft* e comunicare con il modulo *NFC*;
- ***Node.js***: ambiente di esecuzione *JavaScript* utilizzato per eseguire il codice sorgente;
- ***npm***: gestore di pacchetti *JavaScript* utilizzato per installare le dipendenze del progetto.

- **Linguaggi di programmazione e librerie**
  I linguaggi di programmazione e le librerie hanno fornito le fondamenta tecniche per lo sviluppo dell'applicazione, lavorando insieme per creare un'applicazione coesa e funzionale:

  - ***HTML***: linguaggio di markup per la creazione delle pagine *web*, utilizzato in combinazione con React per costruire l'interfaccia utente;
  - ***CSS***: linguaggio di stile per la formattazione delle pagine *web*, utilizzato insieme a React per stilizzare i componenti;
  - ***TypeScript***: linguaggio di programmazione utilizzato per sviluppare l'intero progetto, sia il *frontend* che il *backend*;
  - ***Electron***: *framework* utilizzato per creare applicazioni desktop multipiattaforma, permettendo l'integrazione di tecnologie *web* sviluppate con *React* e *TypeScript*;

[28] Unified Modeling Language
[29] Software Development Kit

  - ***React***: libreria *JavaScript* per la costruzione di interfacce utente dinamiche e interattive, utilizzata in combinazione con *Vite* per il *frontend*;
  - ***Vite***: strumento di *packaging* e *bundling*, utilizzato per la creazione del *frontend* dell'applicazione;
  - ***Electron-vite***: strumento di sviluppo utilizzato per la creazione di applicazioni *desktop* con *Electron* e *Vite*;
  - ***PostgreSQL***: sistema di gestione di database relazionali utilizzato per memorizzare i dati dell'applicazione, interagendo con il backend sviluppato in *Node.js* ed *Express*;
  - ***Express***: framework per *Node.js* utilizzato per creare le *API* del backend, facilitando la gestione delle richieste *HTTP* e l'interazione con *PostgreSQL*.

- **Strumenti di documentazione**:
  - **LaTeX**: Utilizzato per la creazione di documenti tecnici e di progetto, assicurando una presentazione professionale e ben formattata. La documentazione è stata fondamentale durante l'attività di analisi dei requisiti e di progettazione, garantendo una chiara comprensione delle funzionalità richieste e delle scelte architetturali.

- **Strumenti di comunicazione**
  La comunicazione è stata un aspetto cruciale per il successo del progetto, facilitata da diversi strumenti che hanno permesso una collaborazione efficace e continua:
  - ***Google meet***: utilizzato per le riunioni a distanza con il tutor aziendale e i colleghi, garantendo una comunicazione chiara e frequente;
  - ***Discord***: utilizzato per comunicazioni semi-formali, permettendo discussioni tecniche rapide e collaborazioni quotidiane;
  - ***Telegram***: utilizzato per comunicazioni informali, facilitando la condivisione di aggiornamenti rapidi e il coordinamento giornaliero.

3.2 Attività svolte

3.2.1 Analisi dei requisiti

Prima di iniziare la progettazione e la codifica del progetto, ho svolto un'analisi dei requisiti per capire meglio cosa l'azienda si aspettasse dal progetto e quali fossero le funzionalità richieste. Ho svolto l'analisi dei requisiti in collaborazione con il tutor aziendale tramite colloqui in presenza. Il tutor ha fornito una panoramica generale del progetto e ha chiarito i requisiti principali. Durante l'analisi dei requisiti, ho identificato i requisiti funzionali, i requisiti qualitativi e quelli di vincolo, e infine i casi d'uso. Ho creato un documento di analisi dei requisiti che includeva una descrizione dettagliata delle funzionalità richieste, i casi d'uso e i requisiti. Ho archiviato questo documento nella *repository* del progetto in *Bitbucket*.

3.2.1.1 Casi d'uso

I casi d'uso sono descrizioni dettagliate di come un utente interagisce con un sistema per raggiungere un obiettivo specifico. Essi rappresentano gli scenari in cui il sistema viene utilizzato e aiutano a comprendere i requisiti funzionali del progetto.

Ho documentato i casi d'uso utilizzando diagrammi *UML* e descrizioni testuali. Per ciascun caso d'uso, ho specificato gli attori coinvolti, le precondizioni, le sequenze di eventi, le post-condizioni e le varianti. Successivamente, ho associato ogni caso d'uso ai requisiti corrispondenti, assicurando così che ogni funzionalità richiesta fosse adeguatamente descritta e compresa.
I casi d'uso mi hanno aiutato in diversi modi:

- **Chiarezza dei requisiti**: hanno fornito una chiara comprensione delle funzionalità richieste, riducendo le ambiguità e assicurando che tutti i membri del team avessero una visione comune degli obiettivi del progetto;
- **Pianificazione e prioritizzazione**: hanno facilitato la pianificazione e la prioritizzazione delle attività, permettendo di identificare le funzionalità più critiche da implementare in primo luogo;
- **Verifica e validazione**: hanno supportato la verifica e la validazione del software, fornendo scenari concreti per i test e assicurando che il sistema soddisfacesse i requisiti degli utenti finali.

Nomlencatura
I casi d'uso sono identificati e composti in questa maniera:

UC[Numero]-[Descrizione]

- **UC**: Acronimo che sta per "*Use Case*", indicando che si tratta di un caso d'uso.
- **Numero**: Identificativo numerico del caso d'uso per una facile referenziamone, può identificare anche dei sotto casi d'uso.
- **Descrizione**: Breve descrizione del caso d'uso, che indica le azioni e gli obiettivi che il sistema deve compiere per soddisfare le esigenze dell'utente.

Attori
Durante l'analisi dei requisiti, ho identificato due attori principali coinvolti nel sistema:

- **Utente**: l'utente finale che utilizza l'applicazione sul tablet, per effettuare il *check-in* digitale;
- **Operatore/Dipendete**: l'operatore o il dipendete che utilizza l'applicazione su *desktop* per aiutare l'utente durante il *check-in* digitale.

Figura 3.1: Diagramma degli attori

Diagramma dei casi d'uso
Questo documento include solo i casi d'uso che illustrano i principali requisiti degli utenti. Non sono stati inclusi tutti i casi d'uso identificati durante lo stage, ma solo quelli fondamentali per una comprensione essenziale:

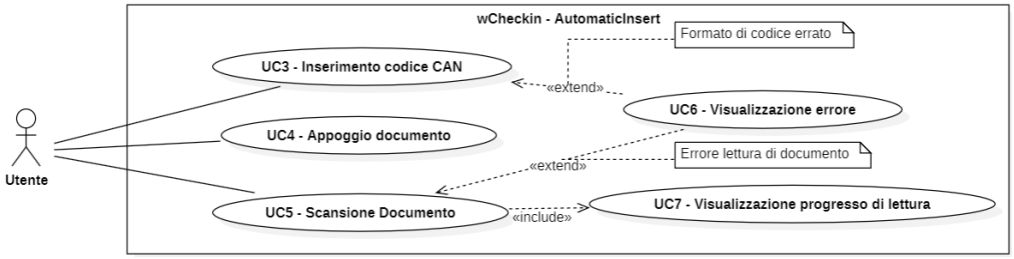

Figura 3.2: Diagramma dei casi d'uso

UC3 - Inserimento codice *CAN*

- **Descrizione**: l'utente inserisce il codice *CAN*, che si trova sul documento, per identificare il documento;
- **Attori coinvolti**: utente;
- **Precondizioni**: l'utente ha scelto di effettuare un *check-in*;
- **Postcondizioni**: l'utente ha inserito il codice *CAN*;
- **Estensioni**: viene presentato un messaggio di errore se il codice *CAN* inserito ha caratteri che non sono numeri o se è più lungo di 6 cifre;
- **Scenario principale**:
  1. L'utente inserisce il proprio codice *CAN* che si trova sul documento, usando la tastiera virtuale;
  2. L'utente preme il pulsante "*Enter*" per confermare il codice inserito.

UC4 - Appoggio documento

- **Descrizione**: l'utente appoggia il documento sul modulo *NFC* del tablet per iniziare la scansione;
- **Attori coinvolti**: utente;
- **Precondizioni**: l'utente ha scelto di effettuare un *check-in*;
- **Postcondizioni**: l'utente ha appoggiato il documento sul modulo *NFC*;
- **Scenario principale**:
  1. L'utente appoggia il documento sul modulo *NFC* del tablet;

UC5 - Scansione documento

- **Descrizione**: il sistema acquisisce i dati del documento tramite la scansione del documento;
- **Attori coinvolti**: utente;
- **Precondizioni**: l'utente ha appoggiato il documento sul modulo *NFC* 3.2.1.1 e ha inserito il codice CAN 3.2.1.1;
- **Postcondizioni**: il sistema ha acquisito i dati del documento e l'utente visualizza l'esito della scansione;
- **Estensione**: viene presentato un messaggio di errore se la scansione del documento non è andata a buon fine;
- **Scenario principale**:
  1. L'utente preme il pulsante "*Scan*" per acquisire i dati del documento;
  2. L'utente visualizza il progresso di lettura (**UC7**)3.2.1.1.
  3. L'utente visualizza l'esito della scansione.

UC7 - Visualizzazione progresso di lettura

- **Descrizione**: l'utente visualizza il progresso di lettura del documento;
- **Attori coinvolti**: utente;
- **Precondizioni**: l'utente ha iniziato la scansione del documento;
- **Postcondizioni**: l'utente visualizza il progresso di lettura del documento;
- **Scenario principale**:
  1. L'utente visualizza il progresso di lettura del documento.

3.2.1.2 Requisiti

I requisiti rappresentano le specifiche che il sistema deve soddisfare per rispondere alle esigenze degli utenti finali e degli *stakeholder*. Emergono da un'analisi dettagliata condotta all'inizio del progetto per identificare le caratteristiche e le capacità del sistema. I requisiti fungono da fondamento per la progettazione, lo sviluppo e il *testing* del *software*, assicurando l'implementazione di tutte le funzionalità necessarie e il rispetto delle aspettative degli utenti.

Come menzionato sopra ho documentato i requisiti in un documento di analisi dei requisiti. Ho associato a ciascun requisito con uno dei casi d'uso identificati, per garantire che ogni funzionalità richiesta fosse adeguatamente descritta e compresa.

Benefici dei Requisiti
I requisiti mi hanno aiutato in diversi modi:

- **Chiarezza degli Obiettivi**: Hanno fornito una chiara definizione delle funzionalità necessarie, assicurando che il team di sviluppo comprendesse esattamente cosa doveva essere costruito.
- **Gestione delle Aspettative**: Hanno aiutato a gestire le aspettative degli stakeholder, definendo chiaramente ciò che il sistema avrebbe e non avrebbe fatto.
- **Pianificazione del Progetto**: Hanno facilitato la pianificazione del progetto, permettendo di stimare tempi e risorse necessari per implementare ciascun requisito.
- **Valutazione della Qualità**: Hanno fornito una base per la valutazione della qualità del software, permettendo di verificare che ogni funzionalità implementata soddisfacesse i criteri di accettazione definiti.

Nomenclatura
I requisiti sono identificati e composti in questa maniera:

R[Tipologia][Codice]

- **R:** requisito;
- **Tipologia:**
  - **F:** funzionale;
  - **V:** di vincolo;
  - **Q:** di qualità.
- **Codice:** numero progressivo che identifica il requisito. Nell casi dei sotto requisiti, il codice è composto da due numeri separati da un punto, [CodiceRequisito].[CodiceSottoRequisito].

Lista dei requisiti
Questo documento include solo i requisiti chiave funzionali, di vincolo e di qualità. Non sono stati inclusi tutti i requisiti raccolti durante lo stage, ma solo quelli essenziali per una chiara comprensione del progetto.

Requisiti funzionali

Codice	Descrizione	Tipo
RF3	L'applicazione tablet deve salvare i dati del *check-in* in un database	Obbligatorio
RF4	L'applicazione tablet deve salvare i dati del *check-out* in un database	Opzionale
RF5	L'applicazione tablet deve permettere l'inserimento dei dati principali manualmente usando una tastiera digitale	Obbligatorio
RF6	L'applicazione tablet deve permettere l'acquisizione dei dati principali tramite *NFC*	Obbligatorio
RF8	L'applicazione desktop deve permettere la ricerca dei dati del *check-in* usando il numero del documento	Obbligatorio
RF11	L'applicazione deve permettere di firmare il documento di *check-in* per l'ingresso	Obbligatorio
RF12	L'applicazione deve permettere di firmare il documento di *check-in* per l'uscita	Opzionale

Tabella 3.1: Requisiti funzionali

Requisiti di vincoli

Codice	Descrizione	Tipo
RV1	Le applicazione sia *desktop* che tablet, deve essere sviluppato in *ElectronJS*	Obbligatorio
RV2	La parte grafica dell'applicazioni deve essere sviluppato in *React*	Obbligatorio
RV3	L'applicazione deve sfruttare i *driver* fatto da *Euronovate* per comunicare con il tablet	Obbligatorio
RV4	L'applicazione deve essere sviluppato usando *Typescript*	Obbligatorio

Tabella 3.2: Requisiti di vincolo

Requisiti di qualità

Codice	Descrizione	Tipo
RQ1	Il codice sorgente deve essere salvato in una *repository* di *BitBucket*	Obbligatorio
RQ2	Deve essere fornito un manuale d'installazione	Obbligatorio
RQ3	Deve essere fornito un manuale di manutenzione	Obbligatorio
RQ4	I colori dell'applicazione devono essere conformi con le linee guida aziendali	Obbligatorio
RQ5	I bottoni dell'applicazione devono avere il stile come riportato nel documento di linee guide aziendali	Obbligatorio
RQ6	I campo di inserimento devono avere il stile come riportato nel documento di linee guide aziendali	Desiderabile
RQ7	La tastiera digitale dovrebbe essere in italiano	Desiderabile

Tabella 3.3: Requisiti di qualità

3.2.2 Progettazione

Dopo l'attività di analisi dei requisiti, ho avviato l'attività di progettazione tecnica con l'assistenza del tutor aziendale. Scopo della progettazione tecnica è definire l'architettura del sistema, i suoi componenti principali e le interazioni tra di essi, basandosi sui requisiti identificati e sulle tecnologie selezionate. Al termine di quest'attività, ho redatto un documento nel quale descrivo le scelte architetturali, i diagrammi *UML*, i pattern di progettazione adottati e le tecnologie selezionate. Successivamente procederò con la descrizione dell'architettura seguendo un approccio *top-down*.

3.2.2.1 Architettura

Fullstack

Iniziando con l'approccio sopra descritto, il progetto è una *fullstack application* simile al modello *client-server*, composta da due parti principali: il *frontend* e il *backend*. Il *frontend* rappresenta l'interfaccia utente dell'applicazione, attraverso cui gli utenti interagiscono con il sistema e visualizzano i dati. Il *backend*, invece, funge da *server* gestore delle richieste utente, elaborando i dati e comunicando con il *database* per memorizzare e recuperare le informazioni necessarie. Queste due parti sono connesse tramite *API*, che consentono la comunicazione bidirezionale tra il *frontend* e il *backend*.

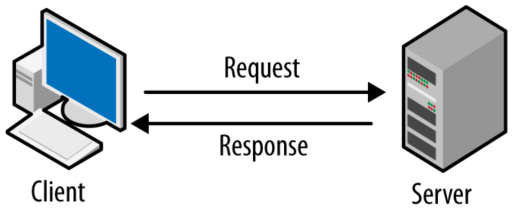

Figura 3.3: Architettura *Client-Server*[30]

[30]Fonte: Fonte: https://github.io/

Frontend
Il *frontend* è l'interfaccia utente dell'applicazione che permette all'utente di interagire con il sistema. L'ho sviluppato utilizzando *React* e viene renderizzato sia nell'applicazione *desktop* che in quella *tablet* mediante *Electron*. Per la gestione del progetto e la configurazione, ho utilizzato *Vite*. Questo strumento offre i seguenti vantaggi:

- Sviluppo rapido grazie all'hot reload.
- Configurazione semplice e flessibile.
- Tempi di build ridotti.

Electron è un *framework* che consente di creare applicazioni *desktop multi piattaforma*, utilizzando il motore di rendering *Chromium* e il motore di esecuzione *Node.js*. È composto da due processi principali: il processo principale e il processo di rendering. Il processo principale gestisce le operazioni a livello di sistema operativo, come la comunicazione con il *driver* del tablet e il *backend*, mentre il processo di *rendering* è responsabile della visualizzazione dell'interfaccia utente e dell'interazione con l'utente.

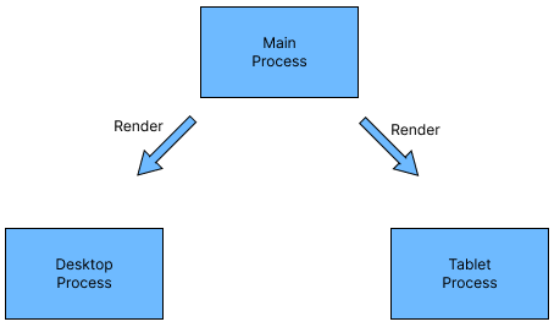

Figura 3.4: Electron Architecture

La comunicazione tra i processi di rendering avviene tramite *IPC* (*Inter-Process Communication*), come mostrato nella Figura 3.5, dove il processo principale funge da *broker* per lo scambio di messaggi tra i due processi. Questo permette al processo *desktop* di comunicare con il processo tablet per trasmettere i dati inseriti dall'operatore nel form e avviare la firma del documento di *check-in*.

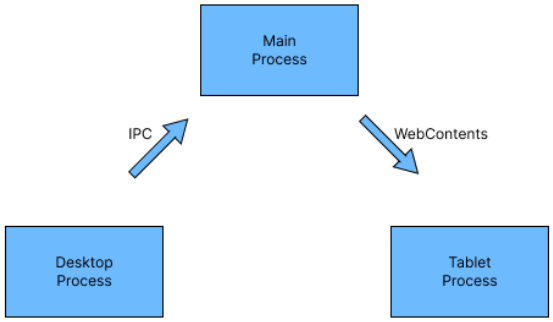

Figura 3.5: Electron Architecture

React viene utilizzato per la creazione dell'interfaccia grafica. Ho scelto di adottare un approccio basato sulla programmazione funzionale e ho suddiviso l'interfaccia utente in piccole unità indipendenti chiamate componenti, ciascuna responsabile di una parte specifica dell'interfaccia. Questo approccio ha permesso di creare componenti riutilizzabili e modulari.

Entrambi i processi sono delle SPA[31], il che significa che l'interfaccia utente viene caricata una sola volta e le successive interazioni avvengono senza ricaricare la pagina. Questo permette un'esperienza utente più fluida e reattiva.

Backend
Il *backend* è il *server* con il *database* che gestisce le richieste dell'utente. Il *server* elabora i dati di *check-in* e comunica con il *database* per memorizzare e recuperare queste informazioni. Ho utilizzato *Express*, un *framework* per gestire le richieste *HTTP*, definendo middleware[32], *routing* e *controller* per una struttura organizzata del codice. Le richieste del *frontend* vengono inviate al *backend* tramite *API REST*, utilizzando i seguenti metodi:

- **/generatePdf**:
  - **{POST}** /: Genera un *PDF* con i dati passati. Chiama il servizio *wTexFusion* tramite una richiesta *POST*. La risposta se positiva ritorna il *PDF* generato in formate *base64*.
- **/ingresso**:
  - **{POST} /registerIngresso**: Registra nella tabella dei utenti se non esiste già, e nella tabella dei *check-in* mette i dati come *pdf* con la firma e *entry_time* e *exit_time*.
- **/search**:
  - **{GET} /user/:document_number**: Cerca un utente nel database tramite il numero di documento, se non viene passato un numero di documento ritorna tutti gli utenti;

[31] Single Page Application
[32] Middleware

  - **{GET} /checkin/:document_number**: Cerca un *check-in* nel database tramite il numero di documento.

- **/uscita**:
  - **{GET} /**: Ritorna i *check-in* con il numero di documento approssimato. Se non viene passato un numero di documento ritorna tutti i checkin. Ritorna i checkin del stesso giorno;
  - **{GET} /:document_number**: Ritorna il *pdfBase64* del checkin cosi sia pronto per la firma di uscita;
  - **{PUT} /**: Fa l'aggiornamento del *pdfBase64* con la firma di uscita.
- **{GET} /status**: Ritorna semplicemente un messaggio di "Server Active" per verificare che il server sia attivo con status 200.

Tutti questi route, tranne */status*, sono protetti con un *middleware* che controlla se il token passato è valido e se l'utente ha i permessi per fare la richiesta.

La database utilizzato è *PostgreSQL*, è e composto da due tabelle:

- **utente**: contiene i dati degli utenti registrati;
  - **id**: id dell'utente;
  - **name**: nome dell'utente;
  - **surname**: cognome dell'utente;
  - **document_number**: numero di documento dell'utente;
  - **document_type**: tipo di documento dell'utente;
- **checkins**: contiene i dati dei checkin degli utenti.
  - **id**: id del checkin;
  - **pdf**: pdf generato con i dati dell'utente;
  - **entry_time**: orario di ingresso dell'utente;
  - **exit_time**: orario di uscita dell'utente;
  - **document_number**: numero di documento dell'utente;
  - **company**: azienda dell'utente;
  - **reason_of_visit**: motivo della visita dell'utente;
  - **host**: host dell'utente.

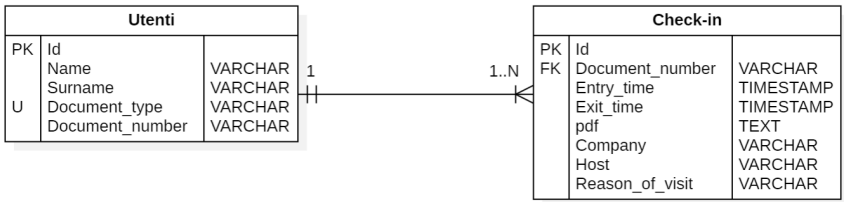

Figura 3.6: Database Schema

Tutto la parte di *backend* è stata containerizzato con *Docker* per garantire la portabilità e la scalabilità del sistema.

3.2.3 Codifica

Durante l'attività di codifica, ho implementato le funzionalità del progetto seguendo le specifiche definite durante l'attività di progettazione. Ho scritto il codice sorgente utilizzando *TypeScript* e ho seguito le linee guida di codifica stabilite dall'azienda. Ho applicato i principi di programmazione pulita per garantire che il codice fosse leggibile, manutenibile e ben strutturato.

Ho utilizzato *Visual Studio Code* come ambiente di sviluppo, il quale mi ha aiutato a scrivere il codice in modo efficiente e a eseguire il debug delle applicazioni. Ho impiegato *Git* come sistema di controllo di versione per tenere traccia delle modifiche al codice.

Ho creato un file *docker-compose.yml* che definisce i servizi principali, il database e il *server*, insieme alle dipendenze necessarie per far funzionare il sistema. Il servizio del database è configurato nel seguente modo:

Listing 3.1: Database Service in docker-compose.yml

```
db:
image: postgres:13
environment:
  POSTGRES_DB: mydatabase
  POSTGRES_USER: myuser
  POSTGRES_PASSWORD: mypassword
volumes:
  - ./backend/database/createTable.sql:/docker-entrypoint-initdb.d/init.
      sql
ports:
  - "5432:5432"
healthcheck:
  test: ["CMD-SHELL", "pg_isready -U myuser"]
  interval: 30s
  timeout: 30s
  retries: 3
```

Mentre il servizio del *server* ha un *Dockerfile* che definisce l'immagine di base e le dipendenze necessarie per eseguire il *server*. Il *Dockerfile* è configurato in questa maniera:

Listing 3.2: Dockerfile for Server

```
# Use an official Node.js runtime as a parent image
FROM node:lts-alpine3.19

# Set the working directory
WORKDIR /usr/src/app

# Copy package.json and package-lock.json from the root of the backend
    directory
COPY ../package*.json ./

# Install dependencies
```

```
RUN npm install

# Copy the rest of the application code from the server directory
COPY . .

# Compile TypeScript code
RUN npm run build

# Expose the port the app runs on
EXPOSE 3000

# Command to run the application
CMD ["npm", "start"]
```

Ho dovuto creare anche un template, scritto in LaTeX per generare il *PDF* con i dati del check-in. Ho integrato il template con il servizio *wTexFusion* per generare il *PDF* dove sia possibile di effettuare la firma digitale.

Figura 3.7: Template PDF with Black Border

3.2.4 Verifica e validazione

Verifica e validazione sono due attività fondamentali per garantire la qualità del *software*. La verifica consiste nel controllare che il *software* sia conforme ai requisiti specificati, esaminando il codice, i documenti di progetto e altre risorse correlate per individuare e correggere eventuali errori. La validazione, invece, si occupa di verificare che il *software* soddisfi le esigenze e le aspettative degli utenti finali, assicurando che il prodotto finito funzioni correttamente nel suo ambiente operativo previsto.

3.2.4.1 Verifica

Durante l'attività di verifica ho utilizzato i test statici per controllare il codice. I test statici includono la verifica della sintassi, l'analisi del codice per individuare possibili errori logici, l'uso di strumenti di linting per assicurare la conformità agli standard di codifica e l'analisi statica per rilevare vulnerabilità di sicurezza. Questi test mi hanno permesso di individuare e correggere vari problemi nel codice prima dell'esecuzione del codice.

Non ho utilizzato test dinamici automatici, come test unitari o di integrazione automatizzati. Tuttavia, ho effettuato test manuali approfonditi per assicurare che tutte le funzionalità implementate funzionassero correttamente. Questi test includono:

- Test manuali tramite *Postman* per verificare che le *API* funzionassero correttamente e restituissero i dati attesi;
- Test manuali dell'interfaccia utente, *desktop* e tablet, per verificare che tutte le funzionalità fossero accessibili e funzionassero correttamente;
- Test manuali della firma digitale per verificare che il processo di firma funzionasse correttamente e che il documento firmato fosse valido.
- Test manuali del *PDF* generato per verificare che i dati fossero corretti e che il layout fosse conforme al template.

3.2.4.2 Validazione

L'attività di validazione è stata eseguita al termine dell'attività di codifica. Il tutor aziendale di stage e un altro dipendente dell'azienda, hanno supervisionato quest'attività. Durante l'attività di validazione, abbiamo esaminato e testato l'applicativo in modo approfondito per garantire che soddisfacesse tutte le specifiche e gli obiettivi prefissati. La validazione ha incluso una serie di test pratici e funzionali, mirati a verificare l'aderenza del prodotto ai requisiti e alle aspettative degli utenti finali.

3.3 Risultato

3.3.1 Metriche di quantità

Alla fine dello stage, ho valutato i risultati ottenuti in base ai requisiti soddisfatti e alle funzionalità implementate. La tabella seguente riporta i requisiti soddisfatti e le funzionalità corrispondenti implementate:

Tipo	Quantità	Soddisfatti
Funzionali	24	24
Di Vincolo	4	4
Di Qualità	7	7

Tabella 3.4: Requisiti soddisfatti

Nella seguente tabella sono riportate le metriche dei risultati ottenuti durante lo stage. Queste metriche includono i dettagli sul frontend, sul backend e sulla documentazione, fornendo una visione quantitativa dei risultati ottenuti.

Attività	Descrizione	Metrica
	Frontend	
Componenti *React*	Numero di componenti sviluppati	20
Linee di Codice	Totale linee di codice per il *frontend*	3,052
File creati/modificati	Numero di file creati o modificati	42
	Backend	
Routes *REST API*	Numero di routes sviluppate	5
Middleware di Sicurezza	Numero di routes protette da *middleware*	4
Linee di Codice	Totale linee di codice per il backend	520
File creati/modificati	Numero di file creati o modificati	23
	Documentazione	
Documenti creati	Numero di documenti creati	3
Numero di pagine	Totale pagine per la documentazione	32
Linee di Codice	Totale linee di codice LaTeX per la documentazione	1,064

Tabella 3.5: Metriche delle attività svolte

3.3.2 Interfaccia *desktop*

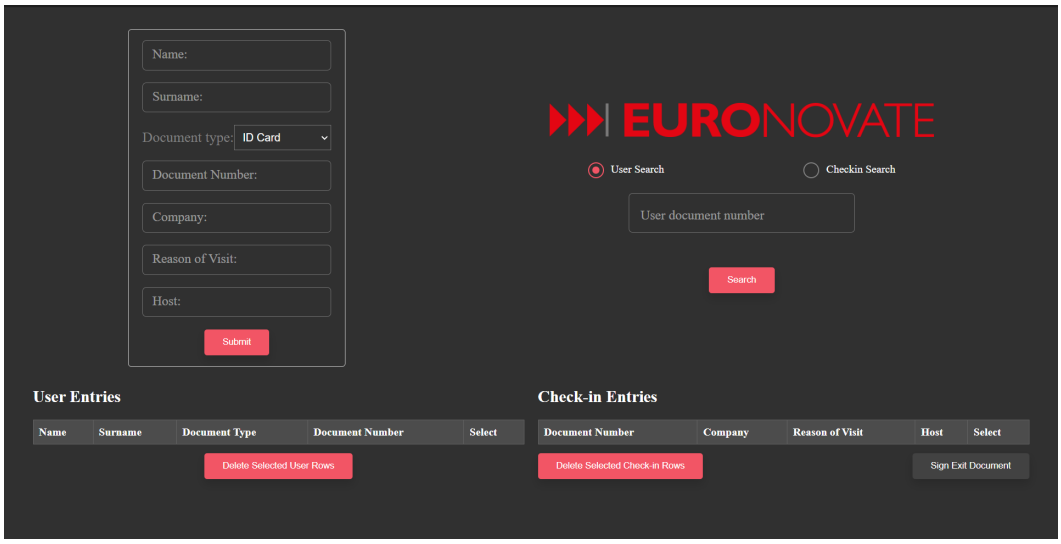

Figura 3.8: Interfaccia utente - Desktop

3.3.3 Interfaccia tablet

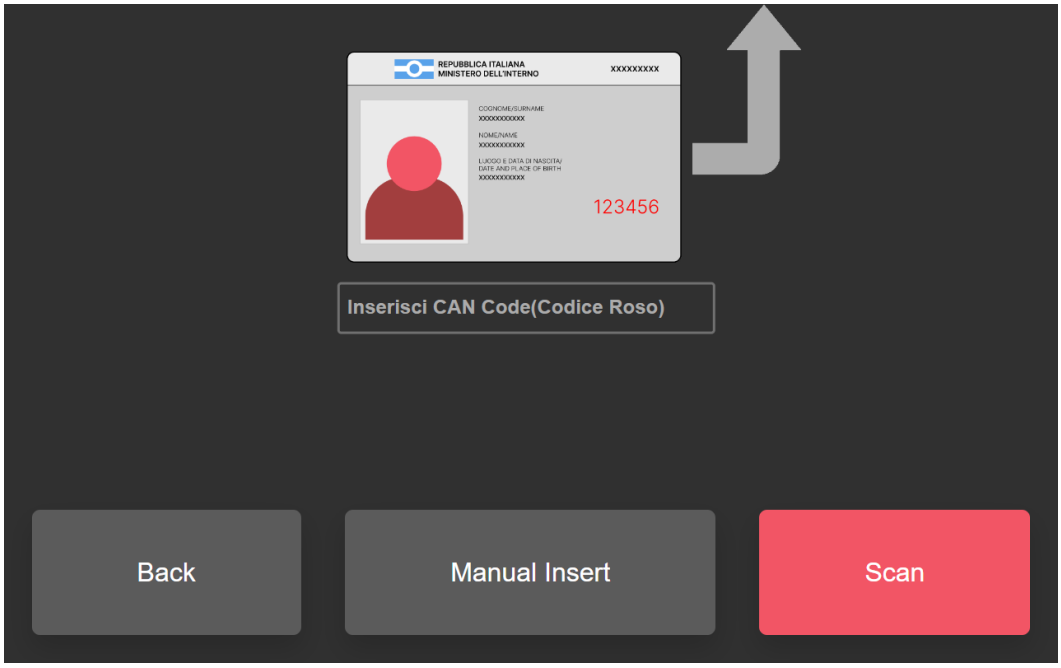

Figura 3.9: Interfaccia utente - Tablet

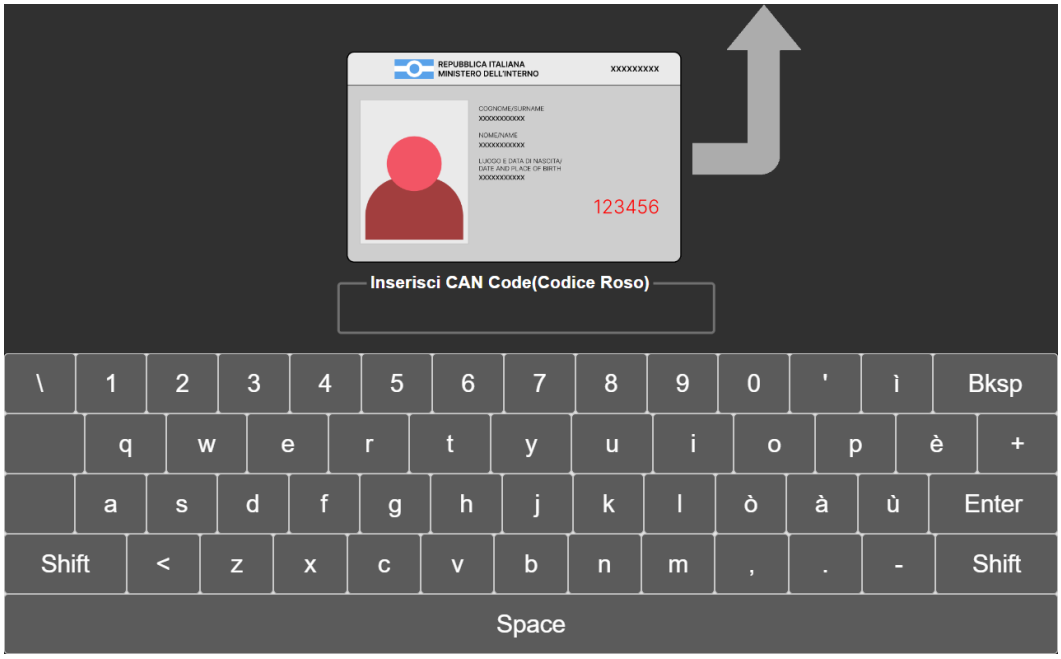

Figura 3.10: Interfaccia utente - Tablet

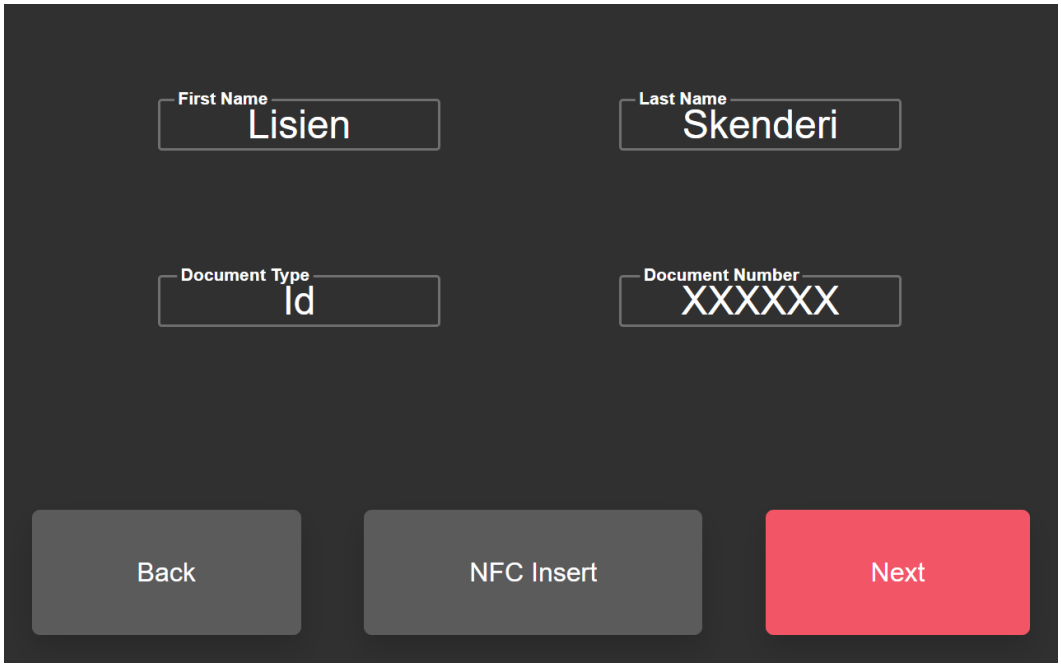

Figura 3.11: Interfaccia utente - Tablet

3.3.4 Firma digitale

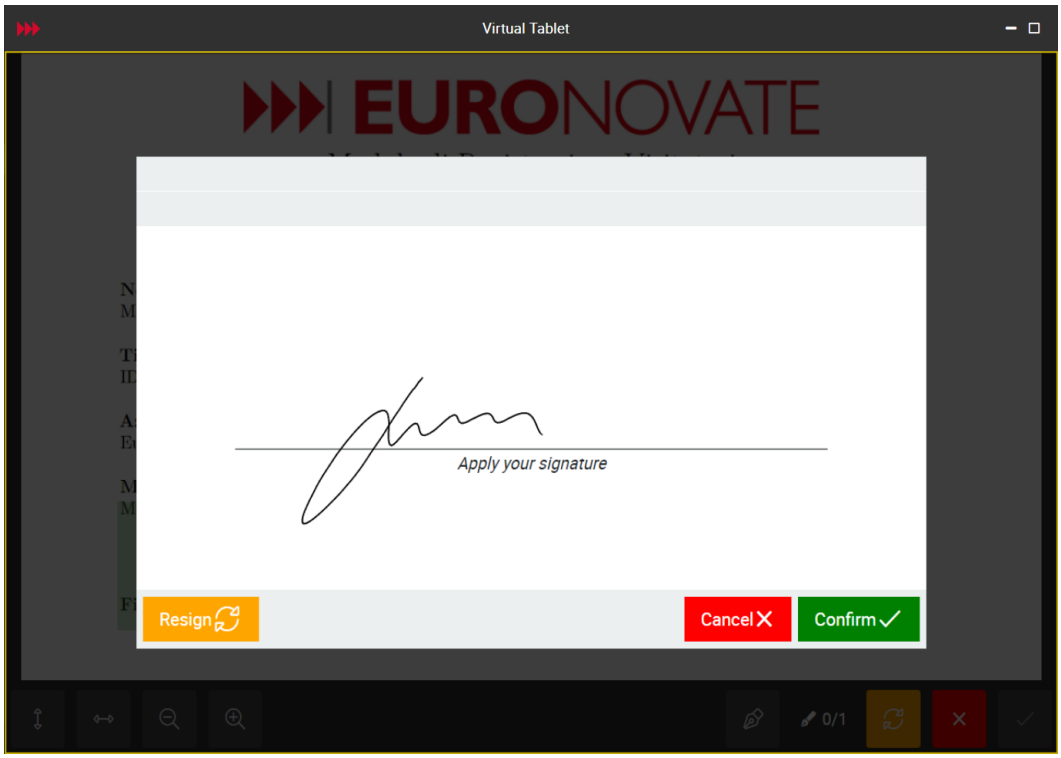

Figura 3.12: Interfaccia utente - Tablet

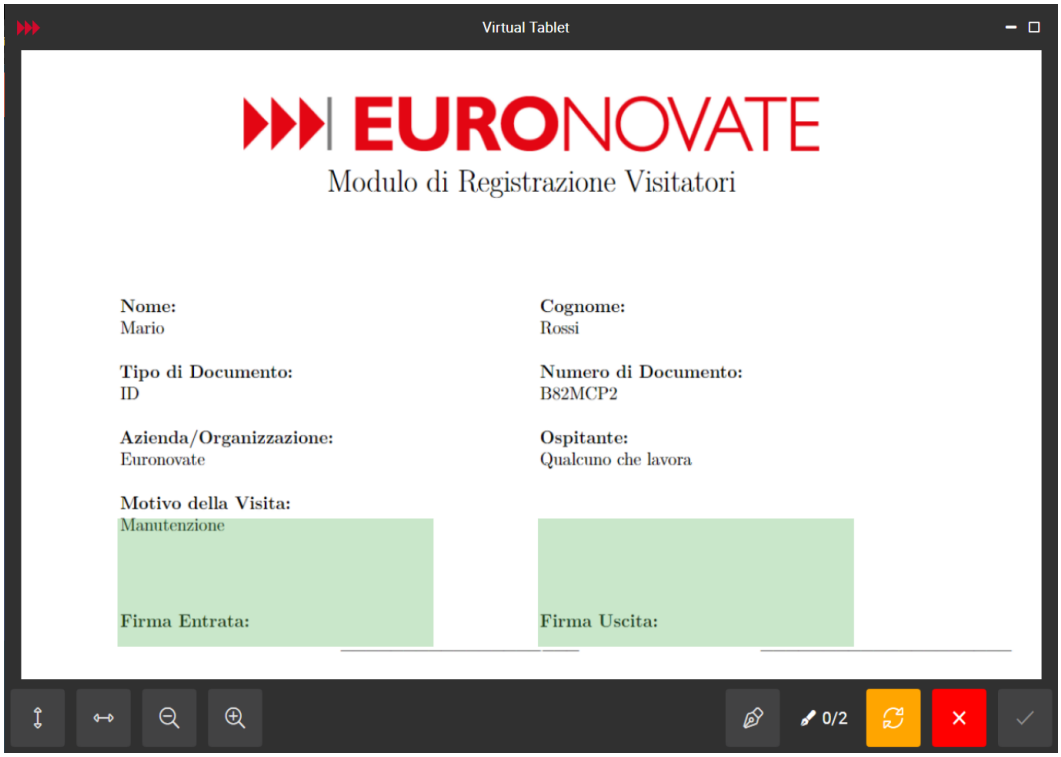

Figura 3.13: Interfaccia utente - Tablet

Figura 3.14: Template PDF with Black Border

Capitolo 4

Retrospettiva

Questo capitolo si concentra sulla retrospettiva del progetto di stage, valutando i risultati ottenuti, gli obiettivi raggiunti e le competenze acquisite durante il percorso di stage. Inoltre, verrà analizzato il rapporto tra l'università e il mondo del lavoro, valutando l'efficacia del percorso di stage come strumento di formazione e inserimento nel mondo del lavoro.

4.1 Obiettivi raggiunti

Valuto positivamente il lavoro svolto durante lo stage e ritengo di aver raggiunto con successo tutti gli obiettivi prefissati, sia quelli del progetto che quelli personali. Sono soddisfatto dei risultati ottenuti e ritengo che il prodotto sviluppato sia pronto per l'uso in un contesto reale. Di seguito sono elencati gli obiettivi del progetto e gli obiettivi personali raggiunti:

4.1.1 Obiettivi del progetto

- **Sviluppo di un'applicazione desktop**: ho sviluppato un'applicazione desktop funzionale che permette agli operatori di gestire l'ingresso, compilando il documento per il visitatore e abilitando la firma digitale. L'applicazione permette a i operatori di abilitare anche la firma durante l'uscita, garantendo un controllo completo del processo. La sua interfaccia utente è stata progettata per essere intuitiva e facile da usare, migliorando l'efficienza del processo;
- **Sviluppo di un'applicazione per il tablet**: è stata creata un'applicazione per il tablet che consente ai visitatori di inserire i propri dati utilizzando la tecnologia NFC o manualmente tramite un modulo. È possibile firmare i documenti digitalmente utilizzando il tablet, non solo per l'ingresso ma anche per l'uscita. Questa applicazione è stata progettata per essere intuitiva e facile da usare, migliorando l'esperienza del visitatore;
- **Integrazione con i dispositivi di firma**: l'applicazione è stata integrata con successo con i dispositivi di firma e il software di *Euronovate*. Questa integrazione permette di utilizzare la firma digitale per sigillare i documenti in modo sicuro e conforme agli standard aziendali;

- **Gestione dei dati**: ho implementato un sistema di gestione dei dati dei visitatori che garantisce la sicurezza e la conformità alle normative sulla *privacy*, garantendo la protezione dei dati personali dei visitatori.

4.1.2 Obiettivi personali

- **Imparare a lavorare in un team**: ho collaborato con altri sviluppatori, condividendo conoscenze e competenze, migliorando così le mie abilità di lavoro di squadra;
- **Sviluppo di un'applicazione *fullstack***: ho sviluppato un'applicazione *fullstack* che integra il *frontend* e il *backend*, garantendo una comunicazione efficace tramite REST API;
- **Creare un prodotto finale utilizzabile in ambito reale**: il prodotto sviluppato è pronto per l'uso in un contesto reale, dimostrando la sua funzionalità e robustezza;
- **Imparare nuove tecnologie**: durante lo stage, ho appreso nuove tecnologie come React, Electron e NodeJs, ampliando le mie competenze tecniche.

4.2 Competenze acquisite

Durante il periodo di stage, ho acquisito una serie di competenze tecniche e trasversali che hanno contribuito al mio sviluppo professionale e personale. Di seguito sono riportate le principali competenze acquisite:

4.2.1 Competenze tecniche

- **React**: ho imparato a utilizzare React per lo sviluppo di interfacce utente dinamiche e reattive. Questa competenza mi ha permesso di creare componenti riutilizzabili e di gestire lo stato dell'applicazione in modo efficiente;
- **Electron**: ho acquisito familiarità con Electron per lo sviluppo di applicazioni desktop multipiattaforma. Questo mi ha consentito di creare un'applicazione desktop che funziona sia su Windows che su macOS;
- **Frontend**: ho sviluppato competenze nel design e nell'implementazione del frontend, creando interfacce utente intuitive e funzionali. Ho utilizzato HTML, CSS e JavaScript, oltre a framework come React;
- **Backend**: ho lavorato sullo sviluppo del backend, creando un server che gestisce le richieste del frontend e comunica con un database PostgreSQL. Ho utilizzato Node.js per costruire API REST-ful;
- **REST-API**: ho imparato a progettare e implementare API RESTful per la comunicazione tra il frontend e il backend, garantendo la sicurezza e l'efficienza della trasmissione dei dati;

- **Integrazione con SDK e altri software**: ho acquisito competenze nell'integrazione di SDK e altri software, come i dispositivi di firma digitale di Euronovate, per arricchire le funzionalità dell'applicazione.

Valutazione delle competenze tecniche acquisite
Le competenze tecniche acquisite durante lo stage sono state fondamentali per il successo del progetto e per il mio sviluppo professionale. Queste competenze mi saranno utili in futuro, sia nel mondo del lavoro che nella vita personale.

4.2.2 *Soft skills*

- **Comunicazione con i colleghi**: ho migliorato le mie capacità di comunicazione lavorando in team, condividendo idee, e ricevendo feedback costruttivi;
- **Gestione del tempo**: ho migliorato la mia capacità di gestire il tempo e le risorse, pianificando e organizzando il lavoro in modo efficiente;
- **Adattabilità**: ho imparato a essere flessibile e ad adattarmi rapidamente ai cambiamenti delle esigenze del progetto e dell'ambiente di lavoro.

Valutazione delle competenze trasversali acquisite
Le competenze trasversali acquisite durante lo stage sono state fondamentali per il successo del progetto e per il mio sviluppo professionale. Queste competenze mi saranno utili in futuro, sia nel mondo del lavoro che nella vita personale.

4.3 Rapporto tra l'università e il mondo del lavoro

Durante il mio stage, ho avuto modo di osservare il rapporto tra le competenze acquisite durante il percorso di studi universitari e quelle che sono richieste nel mondo del lavoro. Posso dire che il rapporto tra le due set di competenze non sono così distanti come avevo immaginato. In seguito sono riportati alcuni aspetti che ho notato durante il mio percorso di stage:

- **Gestione del progetto**: durante il corso di laurea, in particolare nei corsi di **Ingegneria del Software** e **Tecnologie *Web***, ho appreso i principi della gestione del progetto che ho potuto applicare durante lo stage in un contesto reale;
- **Comunicazione**: l'università mi ha preparato a comunicare con i miei compagni e professori del corso attraverso lavori di gruppo e lezioni frontali. Durante lo stage, questa abilità è risultata fondamentale per collaborare con altri sviluppatori e comunicare con il mio tutor aziendale;
- **Gestione del tempo**: nei corsi universitari, ho imparato a gestire il mio tempo tra lezioni, studio, progetti e vita personale. Questa capacità è stata cruciale durante lo stage, dove ho dovuto bilanciare diverse attività e scadenze, assicurandomi di mantenere un alto livello di produttività e qualità del lavoro;

- **Tecnologie utilizzate**: durante il mio percorso di studi, ho acquisito conoscenze su varie tecnologie. Tuttavia, durante lo stage, ho avuto l'opportunità di lavorare con delle librerie e *framework* per lo sviluppo *web*, ma anche con un paradigma come la programmazione funzionale, che non erano state trattate in dettaglio durante i corsi. Quest'esperienza mi ha fatto comprendere l'importanza di continuare ad auto-apprendere e adattarsi alle nuove tecnologie.

Acronimi e abbreviazioni

API Application Program Interface. 40

DPO Data Protection Officer. 40

IDE Integrated Development Environment. 41

IT *Information Technology*. 1

ITS *Issue Tracking System*. 7

NFC Near Field Communication. 41

PM *Projcet Manager*. 3

QA Quality Assurance. 42

REST Representational State Transfer. 42

SDK Software Development Kit. 42

SPA Single Page Application. 42

UI *User Interface*. 42

UML *Unified Modeling Language*. 42

Glossario

API in informatica con il termine *Application Programming Interface API* (ing. interfaccia di programmazione di un'applicazione) si indica ogni insieme di procedure disponibili al programmatore, di solito raggruppate a formare un set di strumenti specifici per l'espletamento di un determinato compito all'interno di un certo programma. La finalità è ottenere un'astrazione, di solito tra l'hardware e il programmatore o tra software a basso e quello ad alto livello semplificando così il lavoro di programmazione. 10, 39

Back-end Il *Back-end* è la parte di un'applicazione web che gestisce le funzionalità lato server, come l'elaborazione dei dati, l'accesso al database e la logica di business. Il back-end è responsabile della gestione delle richieste dei client, dell'elaborazione dei dati e della generazione delle risposte da inviare al client. Il back-end comunica con il front-end tramite API (Application Programming Interface) e fornisce i dati e le funzionalità necessarie per l'applicazione web.*Backend*. URL: https://it.wikipedia.org/wiki/Backend. 10

Data Protection Officer Figura professionale incaricata di garantire che un'organizzazione rispetti le normative sulla protezione dei dati personali. Svolge un ruolo chiave nella conformità alle leggi sulla privacy, come il GDPR. Le sue responsabilità includono la supervisione delle strategie di protezione dei dati, la formazione del personale, la conduzione di audit e fungere da punto di contatto per le autorità di controllo.*Data Protection Officer (DPO)*. URL: https://www.garanteprivacy.it/home/faq/data-protection-officer. 13, 39

Framework Un *Framework* è un insieme di librerie, strumenti e linee guida che forniscono un'infrastruttura per lo sviluppo di applicazioni software. I framework sono progettati per semplificare lo sviluppo di applicazioni, fornendo funzionalità comuni come la gestione delle comunicazioni, la sicurezza e la scalabilità. I framework sono utilizzati per accelerare lo sviluppo di applicazioni, riducendo il tempo e lo sforzo necessari per scrivere il codice da zero.*Framework*. URL: https://it.wikipedia.org/wiki/Framework. 11

Front-end Il *Front-end* è la parte di un'applicazione web che gestisce l'interfaccia utente e le funzionalità lato client, come la visualizzazione dei contenuti, la gestione degli eventi e l'interazione con l'utente. Il front-end è responsabile della presentazione dei dati e delle funzionalità all'utente, e della gestione

delle interazioni utente con l'applicazione web. Il front-end comunica con il back-end tramite API (Application Programming Interface) e riceve i dati e le funzionalità necessarie per l'applicazione web.*Frontend*. URL: https://it.wikipedia.org/wiki/Frontend. 10

FullStack Un *Full Stack Developer* è un professionista che ha competenze sia nello sviluppo front-end che nello sviluppo back-end di applicazioni web. Un Full Stack Developer è in grado di lavorare su tutti gli aspetti di un'applicazione web, dalla progettazione dell'interfaccia utente alla gestione del database, dalla scrittura del codice lato client alla configurazione del server. I Full Stack Developer sono in grado di lavorare su progetti complessi e di gestire l'intero ciclo di vita dello sviluppo software.*Full stack*. URL: https://it.wikipedia.org/wiki/Full_stack. 10

Grafometrica Tipo di firma elettronica avanzata che cattura e utilizza dati biometrici durante il processo di firma. Questi dati includono caratteristiche uniche della firma di una persona, come la pressione esercitata, la velocità, l'accelerazione e l'angolo della penna durante la scrittura. Questi parametri biometrici vengono registrati tramite dispositivi speciali, come tablet o penne digitali, e associati al documento firmato, creando un legame forte e sicuro tra il firmatario e il documento.*Grafometrica*. URL: https://it.wikipedia.org/wiki/Firma_grafometrica. 1

Integrated Development Environment Un *Integrated Development Environment IDE* è un software che fornisce un ambiente di sviluppo integrato per la scrittura, il test e il debug di programmi. Un IDE combina un editor di testo con funzionalità di compilazione, debugging e testing, fornendo un ambiente di sviluppo completo per la creazione di software. Gli IDE sono utilizzati da programmatori e sviluppatori per scrivere codice, testare le applicazioni e gestire i progetti di sviluppo software.*Integrated Development Environment*. URL: https://it.wikipedia.org/wiki/Ambiente_di_sviluppo_integrato. 16, 39

Middleware Il *Middleware* è un software che si colloca tra il sistema operativo e le applicazioni, fornendo un'interfaccia tra i due. Il middleware è progettato per semplificare lo sviluppo di applicazioni distribuite, fornendo funzionalità comuni come la gestione delle comunicazioni, la sicurezza e la scalabilità. Il middleware è utilizzato per integrare sistemi eterogenei, fornire servizi di messaggistica e gestire la distribuzione delle applicazioni.*Middleware*. URL: https://it.wikipedia.org/wiki/Middleware. 26

Near Field Communication La *Near Field Communication NFC* è uno standard di comunicazione wireless a corto raggio che permette lo scambio di dati tra dispositivi elettronici posti a breve distanza tra loro, tipicamente meno di 4 cm. La tecnologia NFC è basata sullo standard RFID (Radio-Frequency Identification) e permette la comunicazione tra due dispositivi NFC o tra un dispositivo e un tag NFC.*Near Field Communication*. URL: https://it.wikipedia.org/wiki/Near_Field_Communication. 2, 39

Quality Assurance in informatica con il termine *Quality Assurance QA* (ing. assicurazione della qualità) si indica l'insieme delle attività svolte per garantire che un prodotto soddisfi i requisiti di qualità richiesti. L'obiettivo è garantire che il prodotto soddisfi le aspettative del cliente e che sia privo di difetti. 4, 39

REST *Representational State Transfer REST* è uno stile architetturale per i sistemi distribuiti, come il World Wide Web. Il termine è stato introdotto da Roy Fielding nella sua tesi di dottorato del 2000. REST si basa su un insieme di principi che definiscono come le risorse sono definite e indirizzate. Le risorse sono identificate tramite URI (Uniform Resource Identifier) e possono essere manipolate tramite rappresentazioni. REST è basato sul concetto di risorsa, che è un concetto chiave per la progettazione di sistemi RESTful. Le risorse sono identificate da URI e possono essere manipolate tramite rappresentazioni, che possono essere trasferite tra client e server.*Representational State Transfer*. URL: https://it.wikipedia.org/wiki/Representational_State_Transfer. 10, 39

Software Development Kit Un *Software Development Kit SDK* è un insieme di strumenti e librerie di sviluppo software che permettono ai programmatori di creare applicazioni per una piattaforma specifica. Gli SDK includono strumenti per la scrittura, il test e il debug del codice, librerie di funzioni e API per l'accesso alle funzionalità del sistema, e documentazione per la creazione di applicazioni. Gli SDK sono utilizzati per sviluppare applicazioni per sistemi operativi, piattaforme hardware e servizi cloud.*Software Development Kit*. URL: https://it.wikipedia.org/wiki/Software_development_kit. 17, 39

Single Page Application Una *Single Page Application SPA* è un'applicazione web che carica una sola pagina HTML e aggiorna dinamicamente il contenuto della pagina senza ricaricare l'intera pagina. Le SPA sono progettate per fornire un'esperienza utente più fluida e reattiva, riducendo i tempi di caricamento e migliorando le prestazioni dell'applicazione. Le SPA sono spesso utilizzate per applicazioni web complesse, come applicazioni di produttività, applicazioni di social media e applicazioni di e-commerce.*Single-page application*. URL: https://it.wikipedia.org/wiki/Single-page_application. 26, 39

User Interface in informatica con il termine *User Interface UI* (ing. interfaccia utente) si indica il punto di interazione tra l'utente e il sistema informatico. L'interfaccia utente è il mezzo attraverso il quale l'utente interagisce con il sistema, permettendo di comunicare con il computer e di controllare le operazioni che il sistema esegue. L'interfaccia utente può essere di diversi tipi, come ad esempio una interfaccia grafica, una interfaccia a riga di comando o una interfaccia vocale.*Interfaccia utente*. URL: https://it.wikipedia.org/wiki/Interfaccia_utente. 10

Unified Modeling Language L'*Unified Modeling Language UML* è un linguaggio di modellazione e specifica basato sul paradigma orientato agli oggetti. UML è utilizzato per descrivere, progettare e documentare i sistemi software, fornendo un insieme di notazioni grafiche per rappresentare i concetti e le relazioni tra

gli elementi di un sistema. UML è stato sviluppato da un consorzio di aziende e organizzazioni nel 1997 ed è diventato uno standard ISO nel 2005. *Unified Modeling Language*. URL: https://it.wikipedia.org/wiki/Unified_Modeling_Language. 17, 39

Bibliografia

Siti web consultati

Backend. URL: https://it.wikipedia.org/wiki/Backend (cit. a p. 40).

Data Protection Officer (DPO). URL: https://www.garanteprivacy.it/home/faq/data-protection-officer (cit. a p. 40).

Framework. URL: https://it.wikipedia.org/wiki/Framework (cit. a p. 40).

Frontend. URL: https://it.wikipedia.org/wiki/Frontend (cit. a p. 41).

Full stack. URL: https://it.wikipedia.org/wiki/Full_stack (cit. a p. 41).

Grafometrica. URL: https://it.wikipedia.org/wiki/Firma_grafometrica (cit. a p. 41).

Integrated Development Environment. URL: https://it.wikipedia.org/wiki/Ambiente_di_sviluppo_integrato (cit. a p. 41).

Interfaccia utente. URL: https://it.wikipedia.org/wiki/Interfaccia_utente (cit. a p. 42).

Middleware. URL: https://it.wikipedia.org/wiki/Middleware (cit. a p. 41).

Near Field Communication. URL: https://it.wikipedia.org/wiki/Near_Field_Communication (cit. a p. 41).

Representational State Transfer. URL: https://it.wikipedia.org/wiki/Representational_State_Transfer (cit. a p. 42).

Single-page application. URL: https://it.wikipedia.org/wiki/Single-page_application (cit. a p. 42).

Software Development Kit. URL: https://it.wikipedia.org/wiki/Software_development_kit (cit. a p. 42).

Unified Modeling Language. URL: https://it.wikipedia.org/wiki/Unified_Modeling_Language (cit. a p. 43).