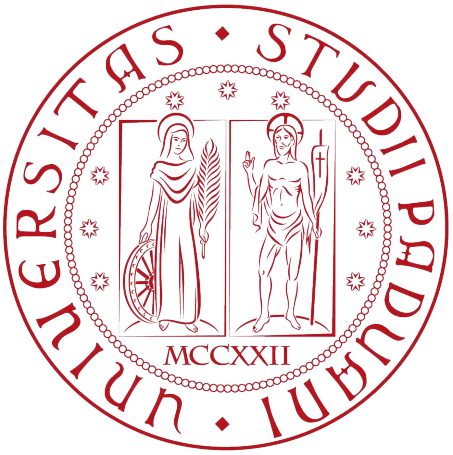
Università degli Studi di Padova

Dipartimento di Matematica "Tullio Levi-Civita"

Corso di Laurea in Informatica



Sviluppo di un'applicazione per l'accesso a servizi online tramite lettura automatica di documenti di identità elettronici

Tesi di laurea triennale

Relatore

Prof. Luigi De Giovanni

Laureando

Marco Galtarossa (1096393)

Anno Accademico 2021-2022

I guai sono come i fogli di carta igienica: ne prendi uno, ne vengono dieci.

— Woody Allen

Ero rimasto senza benzina, avevo una gomma a terra, non avevo i soldi per prendere il taxi, la tintoria non mi aveva portato il tait, c'era il funerale di mia madre, era crollata la casa, c'è stato un terremoto, una tremenda inondazione, le cavalette, non è stata colpa mia, lo giuro su Dio!

— Jake Blues, dal film: The Blues Brother

Dedicata ai miei genitori e a Sara.

Sommario

Il presente documento descrive il lavoro svolto durante il periodo di stage, della durata di circa trecento ore, presso l'azienda Sanmarco Informatica S.p.A.
Lo stage si è svolto lavorando su un nuovo prodotto della Sanmarco Informatica denominato Nuvolhub, destinato a essere un portale di gestione per vari prodotti dell'azienda.
L'obiettivo primario da raggiungere è stata la creazione di una funzione per poter estrarre dati dalla lettura di una Carta d'identità elettronica (CIE$_G$) o di un passaporto. In particolare, è stato di interesse leggere e interpretare il codice MRZ$_G$ (Machine Readable Zone), un codice alfanumerico presente nei documenti sopracitati. In questo codice sono presenti molti dati sensibili che identificano il possessore del documento. Questi dati, vengono poi usati, per aiutare l'utente nella registrazione presso il portale Nuvolhub.
Per questo obiettivo, è stato realizzato un PoC$_G$ (Proof of Concept). Successivamente è stata integrata la funzione con i sistemi già esistenti dell'azienda.
Il secondo obiettivo della tesi è stato di realizzare l'accesso al portale Nuvolhub tramite *"Entra con CIE"*, un servizio dello stato italiano, che tramite un lettore di carte abilitato, permette di fare l'accesso ai servizi registrati tramite CIE$_G$. Per usufruire di questo servizio, l'azienda deve mandare una richiesta agli organi statali competenti.
Per questo obiettivo, è stato realizzato un PoC$_G$. Successivamente è stata integrata la funzione con i sistemi già esistenti dell'azienda.

Convenzioni tipografiche
Riguardo la stesura del testo, relativamente al documento 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 le occorrenze dei termini riportati nel glossario viene utilizzata la seguente notazione: esempio di utilizzo di un termine nel glossario CIE$_G$.

Ringraziamenti

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

Voglio ringraziare poi la mia ragazza per sopportarmi da tanto tempo e i miei amici per tutti i bellissimi anni passati insieme e le mille avventure vissute.

Padova, Dicembre 2022 Marco Galtarossa

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

In questo capitolo verrà descritta l'azienda, il contesto applicativo e l'ambito in cui l'azienda opera

1.1 Descrizione dell'azienda

Sanmarco Informatica (logo in Figura 1.1) è un'azienda italiana che si occupa di consulenza digitale e sviluppo software, specializzata nella progettazione e nella realizzazione di soluzioni integrate a supporto della riorganizzazione di tutti i processi aziendali e professionali.
Sanmarco Informatica segue più di 2000 aziende e ha più di 500 dipendenti divisi in 6 sedi in tutt'Italia. L'ambizione e la volontà di rinnovarsi hanno permesso all'azienda di evolversi attraverso esperienze e scelte imprenditoriali di successo, che individuano nella specializzazione del proprio capitale umano l'elemento centrale.

Figura 1.1: Logo di Sanmarco Informatica

1.2 Contesto Applicativo

Sanmarco opera in molteplici ambiti:

* cybersecurity e data protection;
* gestione documentale (firma digitale, fattura elettronica, conservazione digitale, ecc.);
* qualità e governance;
* project management (gestione e avanzamento progetti, carico lavoro risorse, ecc.);

- process architect (process design, organization charts, ecc.);
- fabbrica digitale;
- amministrazione e finanza,operations, logistica e commerciale;
- business intelligence e IoT (Internet of Things);
- sviluppo web, e-commerce e app.

Per ogni prodotto, l'azienda suddivide il lavoro in vari team. Per lo stage, sono stato inserito nel team che si occupa dello sviluppo e assistenza dei prodotti inerenti la firma digitale.
La firma digitale è uno strumento di cui nessuna azienda può farne a meno. In particolare, Sanmarco mette a disposizione il suo software denominato FirmaE, che serve per firmare digitalmente documenti.
Questo può avvenire sia all'interno del portale web, sia tramite web service anche da altri prodotti. Questo rende il software integrabile da chiunque.

1.3 Organizzazione del lavoro

In questa sezione verranno illustrati i processi aziendali conosciuti durante lo stage e le principali tecnologie utilizzate da Sanmarco Informatica.

1.3.1 Modello di sviluppo

L'azienda adotta il modello di sviluppo Agile[22] che si basa sull'interazione continua con gli stakeholder$_G$ e predilige:

- gli individui e le loro interazioni piuttosto che i processi e gli strumenti;
- il software funzionante piuttosto che una documentazione esaustiva;
- la collaborazione con il cliente piuttosto che la negoziazione dei contratti;
- essere propositivi verso il cambiamento piuttosto che rifiutarlo.

L'idea di questo modello non si basa quindi su uno sviluppo sequenziale ma sul concetto di rivedere di continuo le specifiche, adeguandole durante l'avanzamento dello sviluppo software. In questo modo si possono apportare agilmente modifiche mediante metodologie iterative ed incrementali, con un continuo scambio di pareri ed informazioni tra gli sviluppatori ed il committente. Il progetto di stage è stato portato avanti utilizzando il metodo Scrum$_G$[22].
Scrum$_G$ è il metodo più diffuso tra i modelli di sviluppo Agile e prevede di dividere il progetto in blocchi di lavoro detti sprint, alla fine dei quali si crea un incremento del prodotto software. Esso prevede vari meeting per controllare il lavoro svolto e organizzare il lavoro da svolgere nel prossimo periodo. Inoltre, nel corso dello sviluppo del progetto, l'azienda organizza diversi meeting con gli stakeholder$_G$ e i membri del progetto per valutare l'andamento dello sprint. Gli incontri legati a questo modello di sviluppo che vengono svolti di solito sono:

- sprint planning meeting: tenuto ad inizio sprint per poter organizzare il lavoro;
- daily scrum: riunione giornaliera del team di sviluppo per monitorare l'andamento dello sprint;

* sprint review: tenuto a fine sprint per valutare i risultati ottenuti e quali cambiamenti apportare per lo sprint successivo.

1.3.2 Strumenti per l'organizzazione e la comunicazione

Per l'organizzazione del lavoro, il team utilizza la suite Atlassian[13]. In particolare, di questa suite, si vanno a usare i seguenti software:

* **Confluence**: viene utilizzato come contenitore di documenti, dove si vanno a scrivere analisi, verbali delle riunioni, ecc. In alternativa viene usato anche Google Drive ma principalmente si utilizza Confluence;
* **Jira**: in questo software vengono create le issue. Esistono varie tipologie di issue:
  - funzionalità;
  - task;
  - bug;
  - analisi.

  Le issue sono suddivise per sprint mensili;
* **Bitbucket**: è il repository dove vengono caricati i file sorgenti dei progetti. È basato su Git.

Le comunicazioni interne avvengono prevalentemente tramite la chat di *Gmail*, o tramite email. Inoltre, per l'inserimento di ferie, permessi, malattia, ecc, è a disposizione un portale interno aziendale.

Strumenti di sviluppo

Il team si occupa dell'intero processo di sviluppo e test di un prodotto, perciò va a lavorare, sia lato back-end$_G$, sia lato front-end$_G$, anche con personalizzazioni per i clienti.
Per lo sviluppo del back-end$_G$, viene utilizzato il linguaggio Java e come IDE$_G$ viene utilizzato Eclipse[17]. Mentre per lo sviluppo del front-end$_G$ viene utilizzato il linguaggio Typescript, in particolare, usando il framework$_G$ Angular$_G$[12][22] tramite l'IDE$_G$ Visual Studio Code[20].

1.4 Propensione all'innovazione

Essendo Sanmarco Informatica una società IT, l'innovazione è fondamentale. Nello sviluppo delle nuove tecnologie, essere sempre aggiornati è un obbligo. Questo permette di fornire ai clienti soluzioni sempre più innovative ed efficaci. Sanmarco organizza frequenti webinar$_G$ nei quali i dipendenti che hanno maggiori conoscenze su specifici argomenti, le condividono con i colleghi.

1.5 Organizzazione del testo

Il secondo capitolo riassume gli obiettivi da raggiungere durante lo stage e ne approfondisce la descrizione. Approfondisce l'organizzazione del lavoro e il progetto da sviluppare.

Il terzo capitolo approfondisce il processo di analisi dei requisiti e il tracciamento di essi.

Il quarto capitolo approfondisce i processi di progettazione e codifica, descrivendo anche le tecnologie e gli strumenti utilizzati.

Il quinto capitolo riassume il raggiungimento degli obiettivi, approfondisce le nozioni apprese nel periodo di stage e descrive le conclusioni tratte alla fine del periodo di stage.

Capitolo 2

Lo Stage

In questo capitolo verrà trattato lo scopo e l'offerta dello stage

2.1 Vantaggi Aziendali

Per Sanmarco è molto importante attivare stage per studenti universitari, perché in questo modo può formare nuove risorse nel campo dell'informatica, sempre alla ricerca di innovazione e sviluppo.
Per l'azienda è un beneficio poter avere personale proveniente da un ambiente universitario perché questo permette di avere vari punti di vista sui processi e le tecnologie utilizzate durante lo sviluppo di un software o durante altre attività di gestione.

2.2 L'offerta di stage

2.2.1 Scopo dello stage

Sanmarco Informatica sta sviluppando un portale web per fornire servizi in modalità software as a service su piattaforma cloud. Il portale consente ad un'azienda (organizzazione) di registrarsi e creare uno spazio all'interno del quale i propri utenti possono pubblicare dei documenti, che devono essere firmati elettronicamente utilizzando la firma elettronica qualificata (FEQ). Per poter effettuare la firma, è necessario essere in possesso di un certificato rilasciato da una Certification Authority (CA). La procedura per il rilascio del certificato prevede l'inserimento delle proprie generalità all'interno di un portale, allegando copia di un documento di identità (CIE$_G$ o passaporto). Per velocizzare il caricamento dei dati sul portale, si vuole implementare una funzione che faccia un riconoscimento ottico della sezione specificatamente presente (Machine Readable Zone) e compili automaticamente i campi nel portale, lasciando all'utente solamente la verifica degli stessi prima della conferma.
La tesi in oggetto avrà come obiettivi, una volta verificati i requisiti, l'implementazione di un prototipo di applicazione web che permetta di inserire il documento di identità e recuperi i dati dallo stesso, per popolare i campi della form. Inoltre, una volta registrato l'utente, si vuole dargli la possibilità di autenticarsi al portale utilizzando la funzionalità NFC$_G$ della CIE$_G$.

2.2.2 Obiettivi del progetto

Prodotti attesi I prodotti previsti dal piano di lavoro sono:

* analisi dell'infrastruttura richiesta;
* analisi omnicomprensiva in termini di:
  - inquadramento;
  - requisiti applicativi e tecnici;
  - database (se necessario);
  - strumenti/applicativi di terze parti;
  - casi d'uso.
* studio di fattibilità;
* implementazione della soluzione.

Livelli di priorità Gli obiettivi sono differenziati in base a diversi livelli di priorità:

* **O** - obiettivi obbligatori, ai quali riservare la massima priorità;
* **D** - obiettivi desiderabili, non vincolanti ma dal considerevole valore aggiunto.

Le sigle sopra descritte vengono riportate nella seguente Tabella 2.1 che esplicita le priorità dei vari obiettivi.

Tabella 2.1: Tabella degli obiettivi

ID	Descrizione
Obiettivi obbligatori	
O01	Analisi documentazione tecnica
O02	Analisi requisiti applicativi e tecnici e diagramma di flusso
O03	Mockup del front-end$_G$
O04	Implementazione delle parti back-end$_G$ e front-end$_G$ per la lettura delle informazioni dalla foto del documento
O05	Implementazione della parte front-end$_G$ per l'upload dell'immagine del documento e la visualizzazione delle informazioni in esso contenute
Obiettivi desiderabili	
D01	Analisi documentazione tecnica Middleware CIE$_G$
D02	Analisi requisiti applicativi e tecnici e diagramma di flusso
D03	Implementazione front-end$_G$ per l'interfacciamento con il lettore NFC$_G$
D04	Implementazione front-end$_G$ e back-end$_G$ per il login tramite CIE$_G$

Vincoli

Vincoli metodologici Lo stage si è svolto presso la sede di Grisignano di Zocco di Sanmarco Informatica. In questo modo è stato assicurato un contatto diretto con il tutor e con gli altri sviluppatori del team. Ciò ha giovato notevolmente allo svolgimento delle attività avendo avuto la possibilità di chiarire i dubbi istantaneamente. I confronti con il tutor ed il team si sono svolti giornalmente e mensilmente secondo la metodologia Scrum$_G$. A metà stage è stata richiesta la presentazione di un PoC$_G$ con relativa documentazione di quanto fatto fino a quel momento.

Vincoli tecnologici Sanmarco ha fornito un computer per poter svolgere tutte le attività che abbiamo predisposto. La documentazione è stata condivisa attraverso cartelle drive, in modo da caricare i documenti completati e condividerli immediatamente con gli altri membri di progetto. É stato realizzato un accesso ai sistemi e all'intranet dell'azienda per poter lavorare in sincronia con il team.
Per quanto riguarda la comunicazione, è stato fornito un account aziendale Gmail, di cui si usa l'app di messaggistica interna alla mail in modo da comunicare facilmente con un tutti i dipendenti di Sanmarco.

Vincoli temporali L'orario di lavoro previsto da Sanmarco era dalle 8.30 alle 18.00 con un ora e mezza di pausa pranzo dalle 12.30 fino alle 14.00. L'accesso alla sede avveniva tramite badge personale consegnatomi all'inizio dello stage. Le ore complessive dello stage era pari a circa 300 h, distribuite dal 11-07-2022 al 7-09-2022.

2.2.3 Pianificazione del lavoro

* Prima Settimana (40 ore):
  - formazione su procedure interne aziendali; (4h)
  - incontro con persone coinvolte con il progetto per discutere i requisiti e le richieste di implementazione; (4h)
  - ricerca, studio e documentazione per inquadramento progetto; (8h)
  - introduzione ai linguaggi di sviluppo (se necessario); (20h)
  - introduzione agli ambienti di sviluppo. (4h)
* Seconda Settimana (40 ore):
  - analisi dei requisiti applicativi e tecnici e diagramma di flusso; (24h)
  - mockup del front-end$_G$; (16h)
  - MILESTONE: INQUADRAMENTO DEL PROBLEMA COMPLETATO.
* Terza Settimana (40 ore):
  - studio risorse open source lettura ottica MRZ$_G$; (16h)
  - verifica ed integrazione risorse open source su progetto pilota. (24h)
* Quarta Settimana (40 ore):
  - implementazione back-end$_G$ per registrazione utente con upload immagine documento e lettura delle informazioni contenute; (20h)

  - implementazione front-end$_G$ per registrazione utente e upload del documento; (20h)
  - MILESTONE: COMPLETAMENTO OBIETTIVI MINIMI.

* Quinta Settimana (40 ore):
  - ricerca, studio e documentazione Middleware CIE$_G$; (8h)
  - test e verifica funzionalità del Middleware. (32h)

* Sesta Settimana (40 ore):
  - analisi dei requisiti applicativi e tecnici e diagramma di flusso; (24h)
  - mockup del front-end$_G$. (16h)

* Settima Settimana (40 ore):
  - sviluppo parte front-end$_G$ per l'interfacciamento con il lettore NFC$_G$; (32h)
  - implementazione back-end$_G$.(8h)

* Ottava Settimana (24 ore):
  - implementazione prototipo web di login tramite CIE$_G$ e NFC$_G$;
    * implementazione back-end$_G$; (8h)
    * implementazione front-end$_G$. (16h)
  - MILESTONE: COMPLETAMENTO OBIETTIVI MASSIMI.

La Tabella 2.2 riporta una sintesi dei tempi previsti a inizio stage per il raggiungimento degli obiettivi.

Tabella 2.2: Tabella dei tempi previsti

Attività	**Tempo previsto**
Analisi requisiti e stesura documentazione	92h
Studio documentazione tecnica	80h
Formazione su procedure aziendali	4h
Formazione linguaggi di sviluppo	20h
Formazione ambienti di sviluppo	4h
Implementazione parte back-end$_G$	36h
Implementazione parte front-end$_G$	68h
Totale ore	304h

Capitolo 3

Analisi dei requisiti

In questo capitolo verranno elencati i casi d'uso delle funzionalità implementate con i relativi requisiti e il loro tracciamento

3.1 Casi d'uso

Per lo studio dei casi d'uso del prodotto sviluppato sono stati creati dei diagrammi, detti dei casi d'uso (in inglese *Use Case Diagram*). Sono diagrammi di tipo UML$_G$ dedicati alla descrizione del sistema e delle funzioni o dei servizi offerti, e di come gli utilizzatori interagiscono con il sistema.
Lo strumento utilizzato per la realizzazione di tali diagrammi è starUML[16].

3.1.1 Attori dei casi d'uso

Dopo un'attenta analisi, si è definito che per le funzionalità offerte sono presenti unicamente attori primari, in quanto non ci sono collegamenti con nessun framework$_G$ o libreria esterna. Di conseguenza i possibili attori dei casi d'uso analizzati sono i seguenti attori primari:

- **utente non autenticato**: indica che l'utente non ha ancora effettuato l'autenticazione o la registrazione all'interno della web application;
- **utente autenticato**: indica che l'utente ha effettuato l'autenticazione all'interno della web application e che quindi può accedere a diverse funzionalità altrimenti inaccessibili.

La Figura 3.1 rappresenta lo scenario principale che descrive le azioni che i due attori possono effettuare

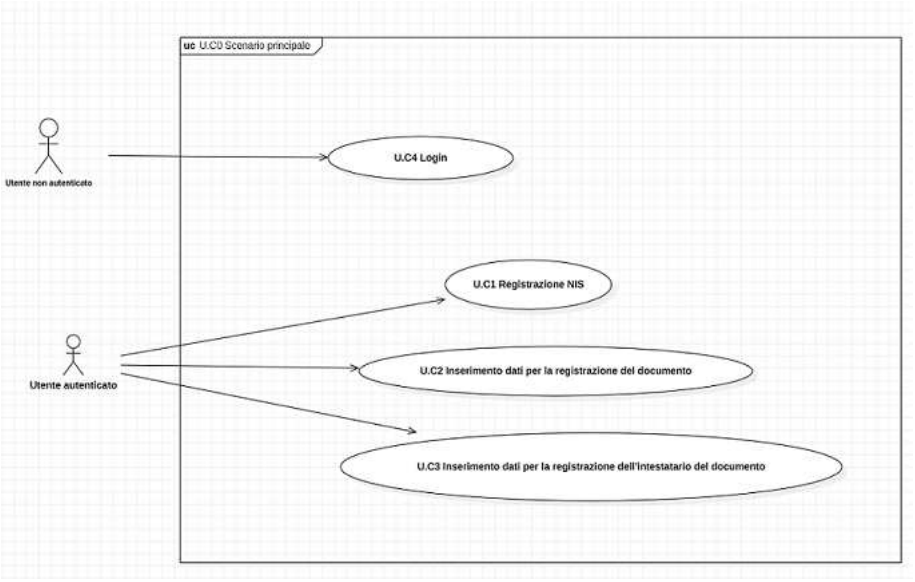


Figura 3.1: U.C0 Scenario principale

3.1.2 Descrizione dei casi d'uso

UC1: Registrazione NIS (Figura 3.2)

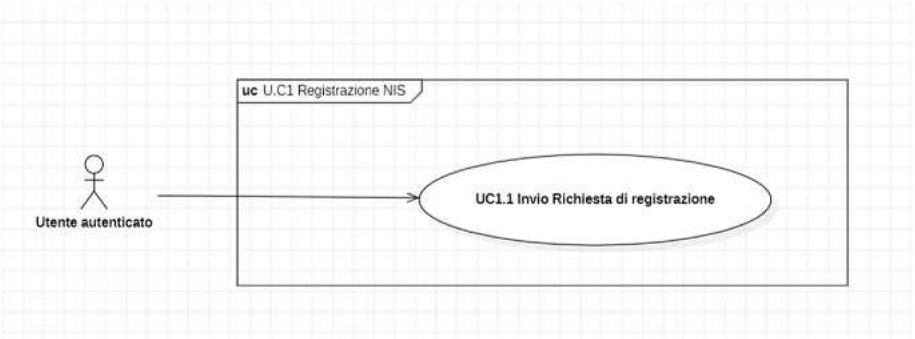


Figura 3.2: U.C1 Registrazione NIS della CIE

Attori Principali: utente autenticato.

Precondizioni: l'utente è autenticato.

Descrizione: l'utente manda la richiesta di registrazione tramite il click sull'apposito link.

Postcondizioni: il NIS_G dell'utente risulta registrato nella piattaforma.

UC1.1: Invio richiesta di accesso

Attori Principali: utente autenticato.

Precondizioni: l'utente è autenticato e ha inviato la richiesta di accesso.

Descrizione: il sistema ritorna una risposta positiva o negativa, a seconda che la registrazione sia eseguita o meno.

Estensioni: viene effettuato un controllo e risulta che la registrazione non è stata eseguita correttamente, si presenta un messaggio di errore (UC8).

Postcondizioni: il NIS_G dell'utente è registrato.

UC2: Inserimento dati per la registrazione del documento (Figura 3.3)

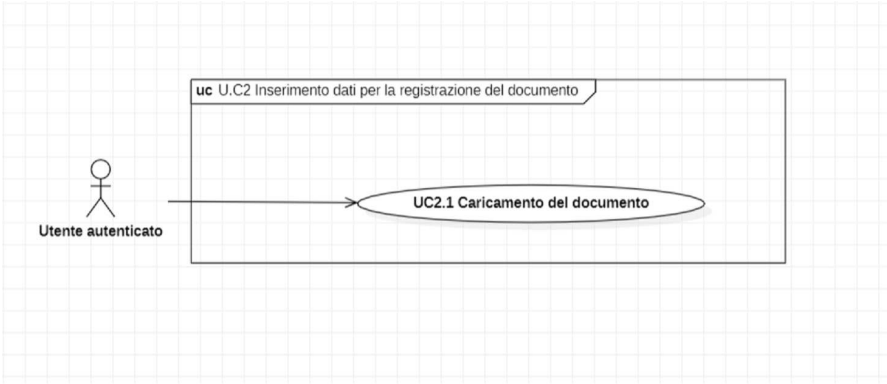


Figura 3.3: U.C2 Caricamento del documento

Attori Principali: utente autenticato.

Precondizioni: l'utente autenticato si trova nella seconda pagina della maschera di registrazione e deve caricare il documento.

Descrizione: l'utente autenticato deve caricare il documento.

Postcondizioni: l'utente visualizzerà l'elaborazione della lettura del documento.

UC2.1: Inserimento dati per la registrazione del documento (Figura 3.4)

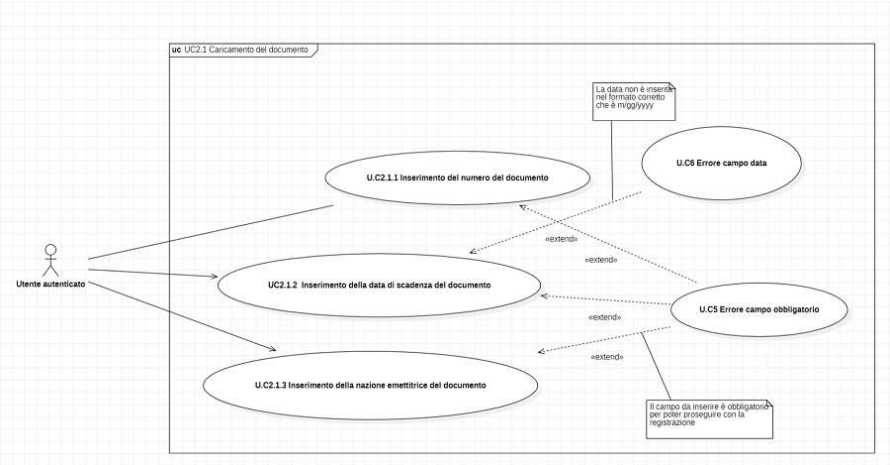


Figura 3.4: U.C2.1 Inserimento dei dati per la registrazione del documento

Attori Principali: utente autenticato.

Precondizioni: l'utente autenticato si trova nella seconda pagina della maschera di registrazione e ha caricato il documento.

Descrizione: l'utente autenticato ha caricato il documento e si ritrova compilati i campi nel seguente modo:

- inserimento del numero del documento (UC2.1.1);
- inserimento della data di scadenza del documento (UC2.1.2);
- inserimento della nazione emettitrice del documento (UC2.1.3).

Nel caso non siamo corretti l'utente li inserirà manualmente.

Postcondizioni: l'utente ha completato o confermato la compilazione dei campi e può procedere con i prossimi passi per la registrazione.

UC2.1.1: Inserimento del numero del documento

Attori Principali: utente autenticato.

Precondizioni: l'utente è autenticato e non ha compilato questo campo.

Descrizione: l'utente autenticato attende la fine dell'elaborazione e, nel caso non vada a buon fine, dovrà inserire manualmente il numero del documento.

Estensioni: viene effettuato un controllo e risulta che questo campo non è stato compilato si presenta quindi un messaggio di errore e viene fornita la possibilità di inserire nuovamente il dato (UC5).

Postcondizioni: il sistema o l'utente ha inserito il numero del documento.

UC2.1.2: Inserimento della data di scadenza del documento

Attori Principali: utente autenticato.

Precondizioni: l'utente non è autenticato e non ha compilato questo campo.

Descrizione: l'utente autenticato attende la fine dell'elaborazione e, nel caso non vada a buon fine, dovrà inserire manualmente la data di scadenza del documento.

Estensioni:

* viene effettuato un controllo e risulta che questo campo non è stato compilato quindi si presenta un messaggio di errore e viene fornita la possibilità di inserire nuovamente il dato (UC5);
* viene effettuato un controllo sul campo inserito e risulta non essere valido quindi si presenta un messaggio di errore e viene fornita la possibilità di inserire nuovamente il dato (UC6).

Postcondizioni: il sistema o l'utente ha inserito la data di scadenza del documento.

UC2.1.3: Inserimento della nazione emettitrice del documento

Attori Principali: utente autenticato.

Precondizioni: l'utente è autenticato e non ha compilato questo campo.

Descrizione: l'utente autenticato attende la fine dell'elaborazione e, nel caso non vada a buon fine, dovrà inserire manualmente la nazione emettitrice del documento.

Estensioni: viene effettuato un controllo e risulta che questo campo non è stato compilato, si presenta quindi un messaggio di errore e viene fornita la possibilità di inserire nuovamente il dato (UC5).

Postcondizioni: il sistema o l'utente ha inserito la nazione emettitrice del documento.

UC3: Inserimento dati per la registrazione dell'intestatario del documento (Figura 3.5)

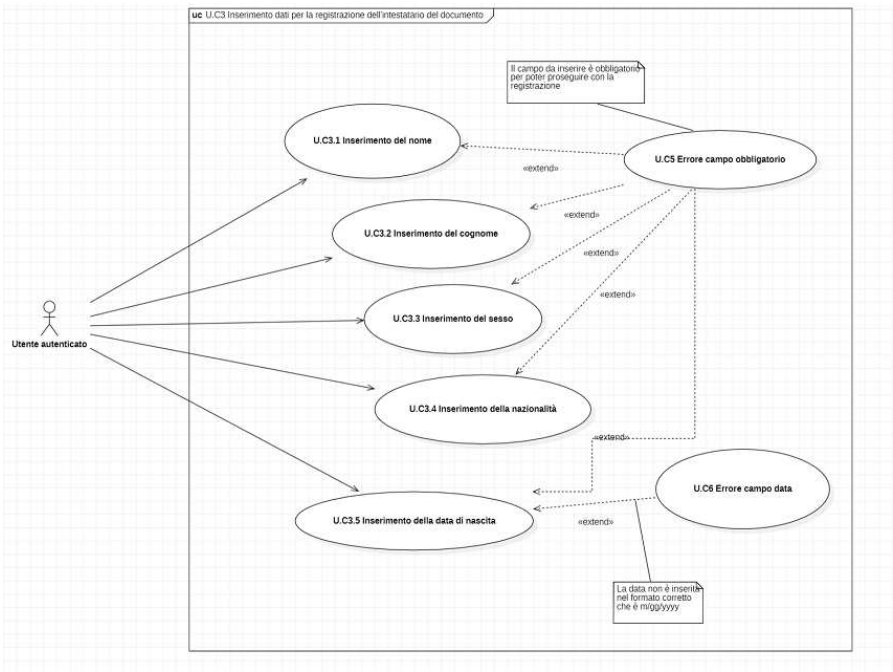


Figura 3.5: U.C3 Inserimento dati per la registrazione dell'intestatario del documento

Attori Principali: utente autenticato.

Precondizioni: l'utente autenticato si trova nella terza pagina della maschera di registrazione.

Descrizione: l'utente autenticato ha già compilato i dati relativi all'azienda e ha caricato il documento e si trova compilati i campi nel seguente modo:

- inserimento del nome (UC3.1);
- inserimento del cognome (UC3.2);
- inserimento del sesso (UC3.3);
- inserimento della nazionalità (UC3.4);
- inserimento della data di nascita (UC3.5).

Nel caso non siamo corretti l'utente li inserirà manualmente.

Postcondizioni: l'utente ha completato o confermato la compilazione dei campi e può procedere con i prossimi passi per la registrazione.

UC3.1: Inserimento del nome

Attori Principali: utente autenticato.

Precondizioni: l'utente è autenticato e non ha compilato questo campo.

Descrizione: l'utente autenticato verifica l'esattezza del dato, nel caso non sia corretto, dovrà inserire manualmente il proprio nome.

Estensioni: viene effettuato un controllo e risulta che questo campo non è stato compilato, si presenta quindi un messaggio di errore e viene fornita la possibilità di inserire nuovamente il dato (UC5).

Postcondizioni: il sistema o l'utente ha inserito il proprio nome.

UC3.2: Inserimento del cognome

Attori Principali: utente autenticato.

Precondizioni: l'utente è autenticato e non ha compilato questo campo.

Descrizione: l'utente autenticato verifica l'esattezza del dato, nel caso non sia corretto, dovrà inserire manualmente il proprio cognome.

Estensioni: viene effettuato un controllo e risulta che questo campo non è stato compilato, si presenta quindi un messaggio di errore e viene fornita la possibilità di inserire nuovamente il dato (UC5).

Postcondizioni: il sistema o l'utente ha inserito il proprio cognome.

UC3.3: Inserimento del sesso

Attori Principali: Utente autenticato.

Precondizioni: l'utente è autenticato e non ha compilato questo campo.

Descrizione: l'utente autenticato verifica l'esattezza del dato, nel caso non sia corretto, dovrà inserire manualmente il proprio sesso.

Estensioni: viene effettuato un controllo e risulta che questo campo non è stato compilato, si presenta quindi un messaggio di errore e viene fornita la possibilità di inserire nuovamente il dato (UC5).

Postcondizioni: il sistema o l'utente ha inserito il proprio sesso.

UC3.4: Inserimento della nazionalità

Attori Principali: Utente autenticato.

Precondizioni: l'utente è autenticato e non ha compilato questo campo.

Descrizione: l'utente autenticato verifica l'esattezza del dato, nel caso non sia corretto, dovrà inserire manualmente la propria nazionalità.

Estensioni: viene effettuato un controllo e risulta che questo campo non è stato compilato, si presenta quindi un messaggio di errore e viene fornita la possibilità di inserire nuovamente il dato (UC5).

Postcondizioni: il sistema o l'utente ha inserito la propria nazionalità.

UC3.5: Inserimento della data di nascita

Attori Principali: utente autenticato.

Precondizioni: l'utente non è autenticato e non ha compilato questo campo.

Descrizione: l'utente autenticato attende la fine dell'elaborazione e nel caso non vada a buon fine dovrà inserire manualmente la propria data di nascita.

Estensioni:

* viene effettuato un controllo e risulta che questo campo non è stato compilato quindi si presenta un messaggio di errore e viene fornita la possibilità di inserire nuovamente il dato (UC5);
* viene effettuato un controllo sul campo inserito e risulta non essere valido quindi si presenta un messaggio di errore e viene fornita la possibilità di inserire nuovamente il dato (UC6).

Postcondizioni: il sistema o l'utente ha inserito la data di nascita.

UC4: Login (Figura 3.6)

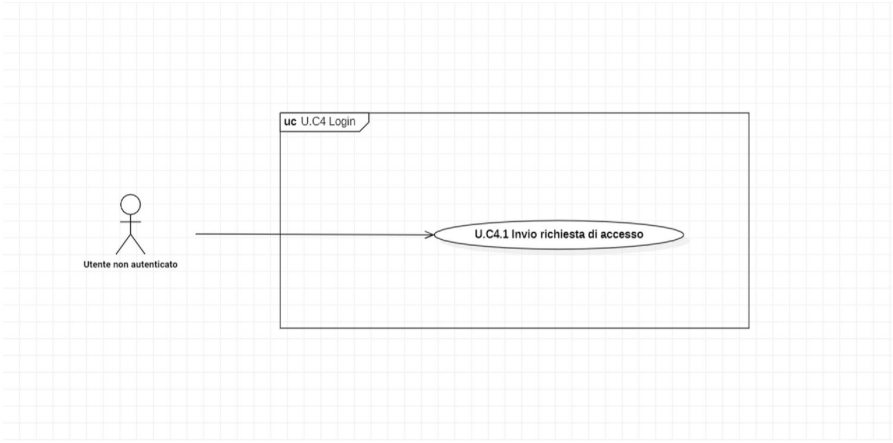


Figura 3.6: U.C4 Login

Attori Principali: utente non autenticato.

Precondizioni: l'utente non è autenticato ma è presente nei registri di sistema.

Descrizione: l'utente manda la richiesta di accesso tramite il click sull'apposito link. Se censito l'utente viene autenticato nel sistema.

Postcondizioni: l'utente risulta autenticato nella piattaforma.

UC4.1: Invio richiesta di accesso

Attori Principali: utente non autenticato.

Precondizioni: l'utente non è autenticato e ha inviato la richiesta di accesso.

Descrizione: il sistema ritorna una risposta positiva o negativa, nel caso in cui l'utente sia censito nei sistemi o meno.

Estensioni: viene effettuato un controllo e risulta che l'utente non è censito e si presenta un messaggio di errore (UC7).

Postcondizioni: l'utente è autenticato e accede alla dashboard.

UC5: Errore campo obbligatorio

Attori Principali: utente autenticato o utente non autenticato.

Precondizioni: l'utente non ha inserito alcun carattere nei campi dati obbligatori.

Descrizione: l'utente non ha inserito alcun carattere in un campo dati obbligatorio e viene informato del mancato riempimento del campo dati.

Postcondizioni: all'utente viene mostrato un messaggio che lo avvisa del mancato riempimento del campo dati obbligatorio.

UC6: Errore campo data

Attori Principali: utente autenticato o utente non autenticato.

Precondizioni: l'utente ha inserito una data invalida.

Descrizione: l'utente ha inserito una data di nascita non valida in quanto non risulta essere nel formato corretto e viene informato dell'errore nell'inserimento del campo dati.

Postcondizioni: all'utente viene mostrato un messaggio che lo avvisa dell'inserimento non conforme del campo data.

UC7: Errore utente non presente nel database

Attori Principali: utente non autenticato.

Precondizioni: l'utente ha premuto correttamente il link inviando la richiesta al database.

Descrizione: l'utente ha premuto correttamente il link inviando la richiesta al database, ma viene effettuato un controllo e risulta non essere presente nel database. Quindi viene informato l'utente dell'errore.

Postcondizioni: all'utente viene mostrato un messaggio che lo avvisa della non corrispondenza nel database.

UC8: Errore nella registrazione NIS

Attori Principali: utente autenticato.

Precondizioni: l'utente ha premuto correttamente il link inviando la richiesta al database.

Descrizione: l'utente ha premuto correttamente il link inviando la richiesta al database ma come risposta viene restituito un http error.

Postcondizioni: all'utente viene mostrato un messaggio che lo avvisa dell'errore nella registrazione.

3.2 Tracciamento dei requisiti

Come risultato di un'attenta analisi dei requisiti e i relativi casi d'uso effettuata sul progetto sono stati individuati diversi requisiti. Questi sono stati suddivisi per classificazione e tipologia, per questo motivo si utilizza un codice identificativo per distinguerli, che è così strutturato:

R[classificazione][tipologia][codice]

La descrizione del codice è la seguente:

* **R**: acronimo per requisito;

* **classificazione**: individua la classificazione del requisito e può essere:

  F = funzionale;

  Q = qualitativo;

  V = di vincolo.

* **tipologia**: individua la tipologia del requisito e può essere:

  O = obbligatorio;

  D = desiderabile.

Nelle tabelle sono riassunti i requisiti funzionali (Tabella 3.1), qualitativi (Tabella 3.2) e di vincolo (Tabella 3.3) e il loro tracciamento con i casi d'uso delineati in fase di analisi.

Tabella 3.1: Tabella del tracciamento dei requisiti funzionali

Requisito	Descrizione	Fonte
RFO1	L'utente autenticato deve caricare il proprio documento	UC2
RFO1.1	L'utente autenticato carica il proprio documento	UC2.1
RFO1.1.1	L'utente autenticato attende la fine dell'elaborazione, o inserisce il numero del documento	UC2.1.1
RFO1.1.2	L'utente autenticato attende la fine dell'elaborazione, o inserisce la data di scadenza del documento	UC2.1.2
RFO1.1.3	L'utente autenticato attende la fine dell'elaborazione, o inserisce la nazione emettitrice del documento	UC2.1.3
RFO2	L'utente autenticato ha caricato il proprio documento ed è nella terza pagina di registrazione	UC3
RFO2.1	L'utente autenticato controlla la correttezza del dato inserito dopo l'elaborazione o inserisce manualmente il proprio nome	UC3,1
RFO2.2	L'utente autenticato controlla la correttezza del dato inserito dopo l'elaborazione o inserisce manualmente il proprio cognome	UC3.2
RFO2.3	L'utente autenticato controlla la correttezza del dato inserito dopo l'elaborazione o inserisce manualmente il proprio sesso	UC3.3
RFO2.4	L'utente autenticato controlla la correttezza del dato inserito dopo l'elaborazione o inserisce manualmente la propria nazionalità	UC3.4
RFO2.5	L'utente autenticato controlla la correttezza del dato inserito dopo l'elaborazione o inserisce manualmente la propria data di nascita	UC3.5
RFD1	L'utente autenticato vuole registrare il proprio NIS_G della CIE_G	UC1
RFD1.1	L'utente autenticato registra il NIS_G	UC1.1
RFD2	L'utente autenticato vuole fare il login	UC4
RFD2.1	L'utente autenticato fa il login	UC4.1

Tabella 3.2: Tabella del tracciamento dei requisiti qualitativi

Requisito	Descrizione	Fonte
RQO1	Il codice sorgente prodotto deve essere disponibile in una repository privata interna su Jira	Interna
RQO2	Deve essere prodotto un documento tecnico che spieghi il funzionamento del codice prodotto	Interna

Tabella 3.3: Tabella del tracciamento dei requisiti di vincolo

Requisito	Descrizione	Fonte
RVO1	Il front-end$_G$ deve essere sviluppato tramite il framework$_G$ Angular$_G$	Interna
RVO2	Il back-end$_G$ deve essere sviluppato tramite il linguaggio Java	Interna

Capitolo 4

Progettazione e codifica

Di seguito viene data una panoramica delle tecnologie, degli strumenti utilizzati per lo sviluppo del front-end e del back-end, del codice scritto e delle scelte fatte per realizzare le funzioni.

4.1 Tecnologie, librerie e linguaggi di programmazione

Di seguito viene data una panoramica delle tecnologie e degli strumenti utilizzati per lo sviluppo del front-end$_G$ e del back-end$_G$.

4.1.1 Tecnologie

Eclipse

Eclipse[17][22] è un ambiente di sviluppo integrato multi-linguaggio e multipiattaforma. Eclipse può essere utilizzato per la produzione di software di vario genere, si passa infatti da un completo IDE$_G$ per il linguaggio Java (JDT, "Java Development Tools") a un ambiente di sviluppo per il linguaggio C++ (CDT, "C/C++ Development Tools") e a plug-in che permettono di gestire XML, JavaScript, PHP e di progettare graficamente una GUI per un'applicazione JAVA.
Il programma, comunemente definito *workbench*, è scritto in linguaggio Java. La piattaforma di sviluppo è incentrata sull'uso di plug-in, delle componenti software ideate per uno specifico scopo e chiunque può sviluppare e modificare i vari plug-in.
Nella versione base, è possibile programmare in Java, usufruendo di comode funzioni di aiuto quali: completamento automatico, suggerimento dei tipi di parametri dei metodi, possibilità di accesso diretto a CVS e riscrittura automatica del codice (funzionalità questa detta di Refactoring) in caso di cambiamenti nelle classi.

Angular

Angular 2+, o solo Angular$_G$[12][22], è un framework$_G$ open source per lo sviluppo di applicazioni web, e costituisce l'evoluzione di AngularJS, sviluppato principalmente da Google. Il linguaggio principalmente usato per sviluppare con framework$_G$ è **TypeScript**. Questa tecnologia permette di progettare e implementare progetti

strutturati per la realizzazione di interfacce utente, con immediati vantaggi in termini di robustezza del codice, testabilità e manutenibilità, creando applicazioni veloci e performanti. Angular$_G$ mette a disposizione un ambiente per la creazione veloce di app, moduli, componenti, il supporto degli IDE$_G$, nonché dei numerosi plugin facilitano la stesura del codice. Angular$_G$ è composto da diversi componenti che messi insieme formano un'applicazione:

* **moduli**: un modulo è un contenitore di funzionalità che ha il compito di esporre queste funzionalità ad altri moduli. Questa suddivisione in moduli torna utile in quanto le applicazioni sono spesso suddivise in diverse aree tematiche (clienti, fornitori, fatture, spedizioni, ecc.). Avere perciò una buona suddivisione dei componenti velocizza il lavoro e permette una più rapida comprensione del software. Non è comunque obbligatorio creare moduli;

* **component**: un component è una classe che gestisce una view della nostra applicazione. Il component si preoccupa di recuperare i dati da renderizzare sulla view e gestisce l'interazione dell'utente con la view. Il pattern ModelViewViewModel e il component ViewModel sono equivalenti. Sono presenti vari tipi di component:

  - **template**: un template è un frammento di HTML che renderizza i dati contenuti nel component che lo gestisce. Questi dati vengono renderizzati attraverso un'apposita sintassi di binding che è una peculiarità di Angular$_G$. Oltre a renderizzare i dati, il template è anche responsabile del collegamento tra gli eventi sulla UI è il loro gestore nel component. Nel pattern MVVM, possiamo paragonare il template alla view;
  - **servizi**: un servizio è una classe che svolge un compito ben preciso all'interno dell'applicazione;
  - **metadati**: i metadati sono il collegamento tra il component e il template. Poiché il component non è altro che una classe e il template non è altro che un frammento di HTML, i metadati sono il collante fra i due componenti cioè sono quello che Angular$_G$ utilizza per fare in modo che i componenti si parlino;
  - **direttive**: una direttiva permette di customizzare il codice HTML e come questo viene renderizzato sul browser. Possiamo per esempio creare dei tag HTML custom e poi fare in modo che una direttiva li trasformi in codice HTML interpretabile dai browsers. Possiamo per esempio definire degli attributi su tag esistenti per estenderne il funzionamento (ad esempio aggiungere stili, classi css e altro ancora).

Oltre a questi componenti c'è anche un'altra caratteristica nell'architettura di Angular$_G$ che lo rende molto potente: la Dependency Injection. Ogni classe che creiamo può avere una dipendenza da un'altra classe. Angular$_G$ ha al suo interno un motore di dependency injection che si occupa di fornire queste dipendenze alla nostra classe sollevandoci dal compito di fornirle.

DBeaver

DBeaver[15] è un programma usato per la gestione di database, durante lo svolgimento del progetto il programma è stato usato per la gestione di un database SQL$_G$ MariaDB.

Spring

Il framework Spring[21][22] è una piattaforma Java che fornisce un'infrastruttura di supporto per sviluppatori in Java che facilita e velocizza lo sviluppo dell'applicazione. I benefici di Spring sono:

* dipendenze esplicite ed evidenti grazie alla *Dependency Injection*;
* i contenitori legati all'*Inversion of Control* tendono ad essere leggeri;
* non reinventa nulla, prende quello che è già esistente e lo rende disponibile;
* è organizzato in modo modulare, quindi il programmatore utilizza i pacchetti che gli interessano;
* è un *model view controller* framework ben formato;
* fornisce un'interfaccia di gestione delle transizioni.

Durante lo stage sono stati utilizzati i seguenti moduli: Spring Boot (Sezione 4.1.1), Spring Data (Sezione 4.1.1, Spring Data JPA (Sezione 4.1.1) e Spring Data REST (Sezione 4.1.1).

Spring Boot Spring Boot è un progetto Spring che ha lo scopo di rendere più semplice lo sviluppo e l'esecuzione delle applicazioni Spring. Un'applicazione richiede molti *metadati* per la configurazione e può risultare pesante, Spring Boot invece alleggerisce questo meccanismo fornendo una configurazione automatica. Infatti esso è basato su opinioni e convenzioni proprie per avere una configurazione minima con la possibilità di riscriverle se necessario. Alcune caratteristiche principali sono legate ai seguenti aspetti:

* *starter dependencies*, ovvero configurazione automatica delle librerie e delle dipendenze;
* configurazione automatica di *bean* e componenti e delle loro relazioni;
* *actuator*, per ispezionare un applicazione in esecuzione.

Spring Boot semplifica la gestione delle dipendenze. La classe detta *Spring Boot Application* esegue l'avvio dell'applicazione etichettando la classe di configurazione, abilitando la scansione e l'identificazione automatica dei componenti, creando infine i componenti mancanti e necessari alla configurazione dell'applicazione.
Per riassumere, i vantaggi offerti da Spring Boot sono:

* possibilità di incorporare applicazioni web server per cui non è necessario l'uso di file WAR, acronimo di *Web Application Archive*;
* configurazione Maven semplificata;
* configurazione automatica se possibile;
* fornitura di caratteristiche non funzionali come metriche o configurazioni esternalizzate.

Spring Data Spring Data è un progetto ad alto livello di Spring con lo scopo di unificare e facilitare l'accesso a diversi livelli di archiviazione, ovvero database sia relazionali che non relazionali. Spring Data fornisce interfacce generiche per gli aspetti legati a *Crud Repository* e *Paging And Sorting Repository* per ottenere implementazioni specifiche dall'archivio di persistenza. Grazie alle repository, basterà scrivere l'interfaccia con i metodi di ricerca, definiti in base ad un determinato insieme di convenzioni che possono variare in base al tipo di archiviazione in uso. Sarà Spring a fornire un'implementazione appropriata di tale interfaccia in fase di esecuzione.

Spring Data JPA Spring Data JPA aiuta a implementare facilmente sistemi basati su archiviazione di tipo JPA. Il framework$_G$, usato per il linguaggio Java, gestisce la persistenza dei dati in un database relazionale che usano le *Java Platform*, *Standard Edition* e *Java Enterprise Edition*. Questo modulo infatti si occupa del supporto per i livelli di accesso ai dati basati su JPA e rende più semplice la creazione di applicazioni basate su Spring che utilizzano tecnologie di accesso ai dati.
Per molto tempo, implementare un livello di accesso ai dati è risultato complicato: era necessario scrivere molto codice per eseguire *query* semplici, impaginazione e controllo dei dati. Spring Data JPA mira a ridurre lo sforzo all'importazione effettivamente necessario. Lo sviluppatore scriverà le interfacce del *repository*, inclusi i metodi personalizzati e sarà Spring a fornire l'implementazione automaticamente.

Spring Data REST Spring Data REST semplifica la creazione di servizi Web REST basati su *repository* di Spring Data, analizza il modello di dominio dell'applicazione ed espone le risorse HTTP basate su *hypermedia* per gli aggregati contenuti nel modello. Spring Data REST definisce automaticamente gli *endpoint* di aggiunta, visualizzazione, rimozione e modifica. Così facendo viene eliminato molto del lavoro manuale solitamente associato a queste attività e rende semplice l'implementazione delle funzionalità CRUD$_G$.

Visual Studio Code

Visual Studio Code è un editor di codice sorgente sviluppato da Microsoft nel 2015. Esso può essere utilizzato con diversi linguaggi di programmazione, infatti incorpora diverse funzioni che variano dal linguaggio con cui si sta programmando. Un punto di forza di questo editor è la sua grande malleabilità, infatti l'utente può installare o disinstallare diversi plugin in funzione del linguaggio che sta utilizzando. Un altro punto di forza è la sua integrazione con Git, facilitando il controllo di versione del lavoro svolto dal programmatore che ha la possibilità di controllare le modifiche fatte, risolvere eventuali conflitti e salvare il lavoro nella repository. Altri punti di forza di questo editor sono i seguenti:

* *intelliSense*, estensione che fornisce suggerimenti di completamento per variabili, metodi e moduli. In particolare, per i metodi, fornisce una breve spiegazione dei parametri accettati e del loro funzionamento;
* *debug* semplificato grazie ai plugin messi a disposizione e all'interfaccia grafica intuitiva;
* modifica veloce delle righe di codice grazie ad avvisi su codice sospetto, possibilità di selezionare più righe contemporaneamente e suggerimenti sui parametri;
* navigazione e *refactoring* del codice semplice.

4.1.2 Librerie

Tess4J

Tess4J[19] è un Java JNA wrapper per Tesseract OCR API_G.
La libreria *Tesseract OCR*[22] è stata sviluppata originariamente come software proprietario dalla Hewlett-Packard tra il 1985 e il 1995, poi non venne più aggiornato nel decennio successivo. Fu poi rilasciato come open source nel 2005 da Hewlett Packard e dall'Università del Nevada a Las Vegas, rilasciato con la licenza Apache, versione 2.0. Lo sviluppo di Tesseract è attualmente sponsorizzato da Google.
Come tutti i programmi OCR_G (Optical Character Recognition), anche Tesseract serve a convertire il testo contenuto in un'immagine, ottenuta usualmente da uno scanner, in caratteri comprensibili ad un elaboratore di testi. Inizialmente limitato ai soli caratteri ASCII, dall'ottobre 2011 Tesseract supporta i caratteri UTF-8 e riconosce 33 lingue.
È stato usato un Java JNA Wrapper in quanto Tesseract OCR è scritto in C++. Uno dei vincoli tecnologici era l'uso di linguaggio Java, poichè la funzione doveva integrarsi con un programma scritto in questo linguaggio. Tess4J si può importare tramite Maven Central Repository[8], aggiungendo al pom.xml del progetto la seguente dipendenza:

```
1 <dependency>
    <groupId>net.sourceforge.tess4j</groupId>
        <artifactId>tess4j</artifactId>
      <version>5.3.0</version>
  </dependency>
```

Listing 4.1: Dependency tess4j.

La libreria si occupa di fare un riconoscimento ottico dei caratteri (OCR_G) da vari tipi di formati di file passati in Input.
L'estensione dei file riconosciuti deve essere di tipo:

* TIFF, JPEG, GIF, PNG, and BMP image formats;
* multi-page TIFF images;
* pdf document format.

Per permettere alla libreria di leggere correttamente dei caratteri, oltre al file da leggere, bisogna passare anche il *dizionario* della lingua con cui interpretare i vari caratteri che si andranno a leggere. Questo dizionario va scaricato ed è presente nella repository di GitHub di Tesseract-OCR fornita da Google[4]. Successivamente va importanto nella root del progetto.

OpenCV

*"**OpenCV**[18] (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code. The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos ecc".*

Per questo prototipo la libreria è stata utilizzata per modificare l'immagine in input, applicandoli diversi algoritmi di filtraggio all'immagine, ottimizzando la lettura da parte della libreria Tess4J. La libreria per effettuare le sue conversioni vuole che i file passati siano converti in un oggetto denominato *Mat*. Mat è una matrice particolare proprietaria della libreria risultato della divisione del file in una matrice.

4.1.3 Linguaggi di programmazione

Java

Java[22] è un linguaggio di programmazione generico utilizzato per lo sviluppo web. Java è uno dei linguaggi di programmazione più popolari in uso, in particolare, per le applicazioni web client-server. Java è stato presentato nel 1995: creato da James Gosling per l'azienda Sun Microsystems con il nome di Oak, oggi è di proprietà di Oracle, che ne cura la gestione. Si tratta di un linguaggio gratuito, orientato agli oggetti, basato sulle classi e tipizzato staticamente. Java è progettato per avere il minor numero possibile di dipendenze di implementazione, per consentire agli sviluppatori di *"scrivere una volta, eseguire ovunque"* (write once, run anywhere). Infatti il codice è multipiattaforma, e una volta eseguito su una piattaforma, non ha bisogno di essere ricompilato per essere eseguito su un'altra.
Oltre ad essere un linguaggio di programmazione, Java fa anche riferimento all'intero ecosistema che gli gravita attorno, composto da tre componenti fondamentali:

* la Java Virtual Machine (JVM), è un ambiente di esecuzione virtuale, indipendente dalla piattaforma, che converte il bytecode Java in linguaggio macchina e lo esegue;
* il Java Runtime Environment (JRE), è un ambiente runtime necessario per eseguire programmi e applicazioni Java;
* il Java Development Kit (JDK), è il componente principale dell'ambiente Java che contiene JRE insieme al compilatore Java, al debugger Java e ad altre classi.

Typescript

TypeScript[22] è un linguaggio di programmazione open source sviluppato da Microsoft. Più nello specifico, TypeScript è un superset di JavaScript, che aggiunge tipi, classi, interfacce e moduli opzionali al JavaScript tradizionale. Si tratta sostanzialmente di una estensione di JavaScript.
TypeScript è un linguaggio tipizzato, ovvero aggiunge definizioni di tipo statico: i tipi consentono di descrivere la forma di un oggetto, documentandolo meglio e consentendo a TypeScript di verificare che il codice funzioni correttamente.

4.2 Progettazione back-end

Per quanto riguarda la progettazione del back-end$_G$, dovendo integrare la funzione in un software già esistente, è stata usata la configurazione già presente.
Il progetto è ideato seguendo il pattern **MVC** (*model-view-controller*). Il pattern è basato sulla separazione dei compiti fra i componenti software che interpretano tre ruoli principali:

- **model** fornisce i metodi per accedere ai dati utili all'applicazione;
- **view** visualizza i dati contenuti nel model e si occupa dell'interazione con utenti e agenti, questa componente è stata sviluppata usando il framework$_G$ Angular$_G$;
- **controller** riceve i comandi dell'utente e li attua modificando lo stato degli altri due componenti.

Springboot (Sezione 4.1.1) come architettura MVC (Figura 4.1) implementa il patter architetturale DAO[22] (*Data Access Object*), pattern usato per la gestione della persistenza. Si tratta di una classe, con relativi metodi, che rappresenta un'entità tabellare di un database. Questo pattern viene usato principalmente in applicazioni web di tipo Java, per stratificare e isolare l'accesso ad una tabella tramite query, poste all'interno dei metodi della classe, ovvero, al data layer da parte della business logic creando un maggiore livello di astrazione ed una più facile manutenibilità. I metodi del DAO, con le rispettive query, saranno così richiamati dalle classi della business logic. Il vantaggio relativo all'uso del DAO, è il mantenimento di una rigida separazione tra le componenti di un'applicazione, le quali potrebbero essere il "model" e il "controller" in un'applicazione basata su MVC.

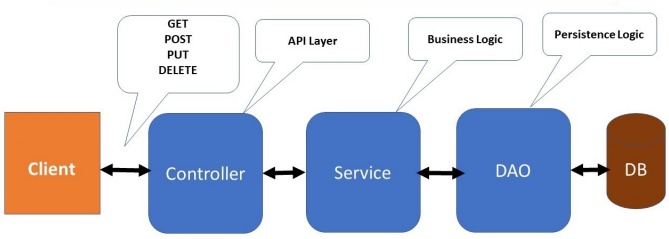


Figura 4.1: Architettuta springboot MVC.

Di seguito vengono descritte le entità, i metodi della classe di Service e i metodi del DAO.

4.2.1 Entità

Le entità rappresentano le entità tabellari del database (Listing 4.2, Listing 4.3). Si potrà notare come in entrambe le entità (Listing 4.2, Listing 4.3) siano presenti le annotazioni di Spring (@Data, @Entity, ecc.) che consentono uno sviluppo più rapido e veloce dell'applicazione.
È inoltre presente un metodo *toDto()* che costruisce un DTO$_G$. Il DTO$_G$, nel progetto, viene usato principalmente per evitare la ridondanza che si viene a creare, se, due entità come User e UserAttachment, fossero in relazione l'una con l'altra. Inoltre, consente una migliore gestione dei dati da inviare al client.

```
@Table(name = "USER", uniqueConstraints = { @UniqueConstraint(
    columnNames = { "CODICE_FISCALE" }) })
@Getter
@Setter
@Entity
@Builder
@NoArgsConstructor
```

```
@AllArgsConstructor
public class User extends JpaEntity implements JpaEntityDto<UserDto> {

    @Column(name = "CODICE_FISCALE")
    private String codiceFiscale;
    @Column(name = "MAIL")
    private String mail;
    @Column(name = "STATO")
    @Convert(converter = StatoUserConverter.class)
    private StatoUser stato;
    @Column(name = "NOME")
    private String nome;
    @Column(name = "COGNOME")
    private String cognome;
    @Column(name = "SESSO")
    private String sesso;
    @Column(name = "DATA_NASCITA")
    private Date dataNascita;
    @Column(name = "LUOGO_NASCITA")
    private String luogoNascita;
    @Column(name = "PROVINCIA_NASCITA")
    private String provinciaNascita;
    @Column(name = "NOTE")
    private String note;
    @Column(name = "NAZIONALITA")
    private String nazionalita;

    @JsonIgnore
    public List<Company> getCompanies() {
        return getCompanyUsers().stream().map(CompanyUser::getCompany).
            toList();
    }

    @OneToMany(cascade = { CascadeType.ALL }, mappedBy = "user",
        orphanRemoval = true)
    @JsonIgnore
    private List<UserAttachment> userAttachments;

    @OneToOne(cascade = { CascadeType.ALL }, mappedBy = "user",
        orphanRemoval = true)
    @JsonIgnore
    private UserCredential userCredential;

    @OneToMany(cascade = { CascadeType.ALL }, mappedBy = "user",
        orphanRemoval = true)
    @JsonIgnore
    private List<VerificationToken> verificationToken;

    @OneToMany(mappedBy = "user")
    @JsonIgnore
    private List<CompanyUser> companyUsers;

    public UserAttachment addAttachment(UserAttachment attachment) {
        if (this.userAttachments == null)
            this.userAttachments = new ArrayList<>();
        this.userAttachments.add(attachment);
        attachment.setUser(this);
        return attachment;
    }

    public void setUserCredential(UserCredential credential) {
        this.userCredential = credential;
        credential.setUser(this);
```

```
    }

    @Override
    public UserDto toDto() {
        getUserAttachments();
        return UserDto.builder().id(getId()).codiceFiscale(codiceFiscale
            ).nome(nome).cognome(cognome)
                .dataNascita(dataNascita).luogoNascita(luogoNascita).
                    provinciaNascita(provinciaNascita).mail(mail)
                .sesso(sesso).stato(stato).nazionalita(nazionalita)
                .note(note)
                .build();
    }
```

Listing 4.2: User Entity.

```
@Table(name = "USER_ATTACHMENT")
@Data
@EqualsAndHashCode(callSuper = true)
@Entity
@SuperBuilder(toBuilder = true)
@AllArgsConstructor
@NoArgsConstructor
public class UserAttachment extends JpaEntity implements JpaEntityDto<
    UserAttachmentDto> {

    @Column(name = "USER_ID", insertable = false, updatable = false)
    private Long userId;
    @Column(name = "S3_KEY")
    private String s3Key;
    @Column(name = "NOME_DOCUMENTO")
    private String nomeDocumento;
    @Column(name = "TIPO")
    @Convert(converter = TipoAttachmentConverter.class)
    private TipoAttachment tipo;
    @Column(name = "STATO")
    @Convert(converter = StatoAttachmentConverter.class)
    private StatoAttachment stato;
    @Column(name = "DATA_SCADENZA_DOCUMENTO")
    private Date dataScadenzaDocumento;
    @Column(name = "NAZIONE_DOCUMENTO")
    private String nazioneDocumento;
    @Column(name = "NUMERO_DOCUMENTO")
    private String numeroDocumento;
    @Column(name = "NIS")
    private String nis;
    @Column(name = "NIS_HASH")
    private String nisHash;
    @Column(name = "NOTE")
    private String note;
    @ManyToOne
    @JoinColumn(name = "USER_ID", referencedColumnName = "ID")
    private User user;

    @JsonBackReference
    public User getUser() {
        return user;
    }

    @Override
    public UserAttachmentDto toDto() {
        return UserAttachmentDto.builder().id(getId()).userId(userId).
            nomeDocumento(nomeDocumento).tipo(tipo)
```

```
                .stato(stato).dataScadenzaDocumento(
                    dataScadenzaDocumento).numeroDocumento(
                    numeroDocumento).nazioneDocumento(nazioneDocumento).
                    note(note).build();
    }
}
```

Listing 4.3: UserAttachment Entity.

4.2.2 Controller

Il Listing 4.4 rappresenta il metodo *uploadImageToDecode*, richiede in ingresso un MultipartFile e restituisce una ResponseEntity di tipo HashMap<String,String>, contenente il risultato della lettura dell'immagine.

```
@PostMapping("/decode-image")
    public ResponseEntity<HashMap<String,String>> uploadImageToDecode(
        @RequestParam("imageFile") MultipartFile file) throws IOException
        , URISyntaxException{
        HashMap<String,String> decodeImagee = userService.decodeImage(
            file);
        return new ResponseEntity<>(decodeImage,HttpStatus.OK);}
```

Listing 4.4: Metodo che restituisce una mappa con i campi ripempiti dalla lettura dell'immagine.

Il Listing 4.5 rappresenta il metodo *isRegistered*, richiede in ingresso un nisAutenticated e restituisce un valore booleano. Il valore sarà *true* se il NIS_G è registrato nel sistema, *false* nel caso non lo sia.

```
@PostMapping("/controlNis")
public ResponseEntity<boolean> isRegistered(NisAutheticated
    nisAutheticated){
  boolean isRegistered = userService.isRegistered();
  return new ResponseEntity<>(isRegistered,HttpStatus.OK);}
```

Listing 4.5: Metodo che controlla se un utente abbia il nis registrato nel sistema.

Il Listing 4.6 rappresenta il metodo *registerNis*, richiede in ingresso un idUser e restituisce una ResponseEntity<void>. Il metodo registra a database il NIS_G e l'$hash_G$,

```
@PostMapping("/registerNis")
public ResponseEntity<void> registerNis(@PathVariable("idUser") Long
    idUser){
  userService.registerNis(idUser);
  return new ResponseEntity<>(HttpStatus.OK);}
```

Listing 4.6: Metodo che registra a database il nis e l'hash di un utente.

4.2.3 Service

Il Listing 4.7 rappresenta il metodo *decodeImage* dove si settano le prime variabili e si fa una distinzione sulla lettura di un file di tipo pdf da un file di tipo immagine. Successivamente una volta identificata la tipologia di documento si lancia la lettura

di quest'ultimo tramite il metodo *readMRZ* (Listing 4.18) nel caso si tratti di un immagine. Con il metodo *readMRZFile* (Listing 4.19) nel caso si tratti di un pdf. Infine si eliminano le immagini prodotte dalla conversione dei documenti.

```
public HashMap<String,String> decodeImage(MultipartFile file) throws
    IOException, URISyntaxException {
        nu.pattern.OpenCV.loadLocally();
        byte[] byteFile= ReadDocumentUtils.compressBytes(file.getBytes()
            );
        User user= new User();
        HashMap<String,String> IDInfo;
        String fileName= file.getOriginalFilename();
        ClassLoader classLoader;
        classLoader = getClass().getClassLoader();
        if(fileName.contains(".pdf")) {
            PDDocument document = PDDocument.load(ReadDocumentUtils.
                decompressBytes(byteFile));
            PDFRenderer renderer = new PDFRenderer(document);
            BufferedImage image = renderer.renderImage(0, 3);
            IDInfo = ReadDocumentUtils.readMRZFile(image,classLoader.
                getResource("tessdata").toURI().getPath(),0);
            document.close();
        }
        else {
            Mat src = Imgcodecs.imdecode(new MatOfByte(ReadDocumentUtils
                .decompressBytes(byteFile)), Imgcodecs.IMREAD_ANYCOLOR);
            IDInfo = ReadDocumentUtils.readMRZ(src,classLoader.
                getResource("tessdata").toURI().getPath(),0);
        }
        File blackhat= new File("blackhat.jpg");File contrast= new File(
            "contrast.jpg");
        File baw= new File("colortogreyscale.jpg");File thresh= new File
            ("thresh.jpg");
        blackhat.delete();contrast.delete();baw.delete();thresh.delete()
            ;
        return IDInfo;
    }
```

Listing 4.7: UserService.

Il Listing 4.8 rappresenta il metodo *isRegistered*. Nella prima parte di questo metodo si estrarre il NIS_G e l'$hash_G$ di un documento appoggiato su un lettore di documenti, mentre nella seconda parte si controlla se esite un utente che abbia il NIS_G registrato nel sistema. In caso positivo si restituisce un *true* in caso contrario *false*.

```
public boolean isRegistered() {
  NisSdk nisSdk = initSdk();
  if(nisSdk.isReady()){
    nisSdk.enroll();nisSdk.access();}
  User user= new User();
  String nis= nisSdk.getNisExit();
  String getHashExit = nisSdk.getHashExit();
  user = userService.getUserByNIS(nis);
  if(user != null) return true;
  else return false;}
```

Listing 4.8: isRegistered.

Il Listing 4.9 rappresenta il metodo *registerNis*. Nella prima parte di questo metodo si estrarre il NIS_G e l'$hash_G$ di un documento appoggiato su un lettore di documenti, mentre nella seconda parte si controlla se esiste un utente. Se quest'ultimo non ha il NIS_G censito a sistema, si registra quello della carta letta e infine si esegue un update dell'utente, inserendo a database il valore del NIS_G.

```
public void registerNis(Long idUser) {
  NisSdk nisSdk = initSdk();
  if(nisSdk.isReady()){
    nisSdk.enroll();
    nisSdk.access();
  }
  User user= new User();
  String nis= nisSdk.getNisExit();
  String hash= nisSdk.getHashExit();
  user = userService.getById(idUser);
  if(user.getNis() == null) {
    user.setNis(nis);
    user.setHash(hash);
    userDao.updateUser(user);
  }
  else return null;
}
```

Listing 4.9: registerNis.

Il Listing 4.10 rappresenta il metodo *getUserByNIS*. Questo chiama il metodo *getUserByNIS* presente nel DAO (Listing 4.11) e restituisce un entità Utente.

```
public User getUserByNIS(String nis) {
  return userDao.getUserByNIS();
}
```

Listing 4.10: getUserByNIS.

4.2.4 Dao

Il Listing 4.11 rappresenta il metodo *getUserByNIS*. Questo metodo contiene la query per estrarre un utente dal database se il NIS_G passato in input è coincidente con il NIS_G salvato nel database.

```
public User getUserByNIS(String nis) {
  TypedQuery<User> query = entityManager.createQuery(
  "select u from User u where u.nis = :nis",User.class);
  query.setParameter("nis",nis);
  return  query.getSingleResult();}
```

Listing 4.11: UserService.

4.3 Parte prima: Lettura di un documento

4.3.1 Codice MRZ

Un *machine-readable passport* (MRP) è un documento leggibile da una macchina i cui dati sono codificati in un formato riconoscibile tramite un OCR_G. La maggior parte dei documenti sono ormai di tipo MRP. Sono standardizzati dal documento $ICAO_G$[1] 9303 (realizzato dall'International Organization for Standardization and the International Electrotechnical Commission as ISO/IEC 7501-1).
Al loro interno è presente un codice denominato *machine-readable zone* (MRZ_G, Figura 4.2), posizionato tipicamente nel fondo delle prime pagine di un documento.
L'$ICAO_G$ 9303 prevede tre tipi di documenti riconducibili all'ISO/IEC 7810:

* il "Tipo 3" è la tipologia tipica dei passaporti. L'MRZ_G è composto da 2 righe, ciascuna composta da 44 caratteri;
* il "Tipo 2" è relativamente raro. L'MRZ_G è composto da 2 righe, ciascuna composta da 36 caratteri;
* il "Tipo 1" è la tipologia tipica delle carte d'identità. L'MRZ_G è composto da 3 righe, ciascuna composta da 30 caratteri.

Le informazioni ricavabili dal codice MRZ_G (Figura 4.2) sono:

* il tipo di documento: posizione 1 della prima riga per entrambi i documenti. C se carta di identità, P se passaporto. Nelle prime CIE_G italiane il tipo del documento è indicato come CI e occupa la posizione 1 e 2 della prima riga;
* la nazione emettitrice del documento: posizione 3÷5 della prima riga per entrambi i documenti;
* il nome e il cognome: posizionato in tutta la terza riga delle CIE_G, posizione 6÷44 nella prima riga dei passaporti;
* il numero del documento: posizione 6÷14 nella prima riga delle CIE_G, posizione 1÷9 nella seconda riga dei passaporti;
* la nazionalità: posizione 6÷18 nella seconda riga delle CIE_G, posizione 11÷13 nella seconda riga dei passaporti;
* la data di nascita: posizione 1÷6 nella seconda riga delle CIE_G, posizione 14÷19 nella seconda riga dei passaporti;
* il sesso: posizione 8 nella seconda riga delle CIE_G, posizione 21 nella seconda riga dei passaporti;
* la data di scadenza del documento: posizione 9÷14 nella seconda riga delle CIE_G, posizione 22÷27 nella seconda riga dei passaporti.

Sono presenti inoltre, dei codici di controllo situati dopo la data di nascita, la data di scadenza del documento, il numero del documento e infine, alla fine della seconda riga è presente un codice di controllo dei tre codici precedenti. I codici servono per verificare la correttezza dei dati inseriti nel documento. Il modo per calcolare i codici di controllo è il seguente:

* si prende in esame uno dei campi controllati dal codice, per esempio il numero di passaporto dell'immagine (Figura 4.2): AE0000005;
* si convertono le lettere in numeri, da A=10 si prosegue progressivamente per le altre lettere dell'alfabeto, perciò: [10, 14, 0, 0, 0, 0, 0, 0, 5];
* si moltiplica la conversione per il patter [7,3,1], perciò: [10*7, 14*3, 0*1, 0*7, 0*3, 0*1, 0*7, 0*3, 5*1]=[70, 42, 0, 0, 0, 0, 0, 0, 5];
* si sommano tutti i numeri usciti: [70, 42, 0, 0, 0, 0, 0, 0, 5]= 117;
* il codice di controllo è il resto della divisione per 10 della somma al punto precedente. Nel caso preso in esempio, il risultato è 7, che coincide con il codice di sicurezza presente nel documento (Figura 4.2).

Figura 4.2: Esempio di codice MRZ nella carta d'identità elettronica (sinistra) e nel passaporto (destra)

4.3.2 Lettura di un documento

Una volta definito Java come linguaggio di sviluppo, la miglior libreria trovata è risultata Tess4J (Sezione 4.1.2, esempio di utilizzo nel Listing 4.12). Anche Amazon propone delle librerie utili a fare OCR_G, ma sono a pagamento e sono più complesse da utilizzare. L'implementazione della lettura OCR_G di un documento tramite questa libreria non è complicata. Infatti, dato in ingresso un file e un dizionario, viene restituita la lettura del file fatta dalla libreria.
La precisione della lettura è direttamente proporzionale alla qualità del documento e del dizionario passato in input. Esiste un altro parametro da poter passare in input, l'*engine*. Questo è un componente interno della libreria che permette di scegliere con quale algoritmo decifrare il documento.
Inoltre, la libreria mette a disposizione metodi per passare in input anche un'area rettangolare specifica, in cui il lettore si andrà a concentrare. Quest'ultima funzionalità descritta non è stata usata nella funzione. Nella repository di GitHub di Tesseract-OCR[4] non è presente un dizionario che interpreti al meglio il codice MRZ_G. Quindi per dare la migliore interpretazione del codice sono state trovate in due repository private due dizionari[6][5], che interpretano con una buona precisione il codice MRZ_G.
Quando un utente carica un immagine del proprio documento, l'immagine può avere molti difetti, può essere: poco definita, troppo scura, troppo chiara, sfocata, ecc. Tutte queste variabili compromettono la corretta lettura del codice. Per risolvere il più possibile questo problema è stata usata la libreria OpenCV[18] per applicare diverse

trasformazioni all'immagine in input. Questo per uniformare le immagini e garantirne la migliore lettura possibile.

```
public class TesseractExample {
    public static void main(String[] args) {
        File imageFile = new File("eurotext.tif");
        ITesseract instance = new Tesseract();  // JNA Interface Mapping
        instance.setDatapath("tessdata"); // path to tessdata directory
        try {
            String result = instance.doOCR(imageFile);
            System.out.println(result);
        } catch (TesseractException e) {
            System.err.println(e.getMessage());
        }
    }
}
```

Listing 4.12: Esempio di implementazione della libreria Tess4J.

4.3.3 Trasformazione di un documento

Come detto precedentemente, per la migliore lettura possibile di un immagine e di conseguenza del codice MRZ_G, bisogna applicare dei filtri. Per farlo, è stata utilizzata la libreria OpenCV che data una matrice di tipo Mat in input, ne restituisce il risultato trasformato secondo la conversione scelta.
Sono state applicate varie combinazioni di filtri, prima il bianco e nero, fondamentale per una buona riuscita della lettura. Successivamente è stato applicato un filtro chiamato *BlackHat* (Sezione 4.3.3). Questo filtro evidenzia le parti nere mettendole in bianco e applicando un filtro di oscuramento sulle altre parti. Infine è stato applicato un filtro che aumenta il contrasto per far risaltare ancora meglio le scritte.
Questo è valido per certe immagini, mentre per altre è sufficiente solo una trasformazione in bianco e nero.
Esistono molteplici filtri e combinazioni, quella trovata è una buona combinazione per svariate immagini prese in input come esempio, ma per avere un ottimo risultato bisognerebbe, su ogni immagine, applicare vari filtri e dizionari fino a trovare la miglior lettura possibile. L'applicazione permette anche di leggere i documenti pdf, in questo caso sono stati applicati due filtri diversi: un filtro chiamato *Threshold semplice Inv* (Sezione 4.3.3) e il filtro della conversione in bianco e nero (Sezione 4.3.3). Con questa combinazione è stato ottenuto un buon risultato su vari pdf presi in input.
Sono state individuate alcune "regole" da rispettare per avere una migliore lettura dell'immagine e di conseguenza una giusta interpretazione del codice MRZ_G:

* l'immagine o il pdf passato, non deve essere troppo piccolo (inteso larghezza e lunghezza delle immagini), altrimenti Tesseract farà fatica a distinguere i caratteri;
* l'immagine è opportuno abbia un rapporto 1:1. Se troppo zoommata, è probabile che legga male i caratteri per via delle imperfezioni create dallo zoom, se troppo poco zoommata è probabile che legga due caratteri al posto di uno;
* meglio se le immagini in input siano di tipo PNG, anche altri formati vanno bene, però l'importante è che siano "ben definite" e non sfocate o mosse;
* l'input delle immagini è opportuno sia a colori e non in bianco e nero, questo perché si occuperà Opencv di fare questa trasformazione. Come detto precedentemente,

per i pdf è stato applicato un filtro diverso che non vuole in input immagini in bianco e nero. Sarebbe meglio passarli a colori, ma in ogni caso la funzione creata è strutturata per accettare anche input in bianco e nero.

Una difficoltà riscontrata nell'implementazione è stata la mancanza, da parte di OpenCV[18], di una dipendenza maven da aggiungere al pom del progetto. Questo costringeva a importare manualmente la libreria all'interno del progetto. Il tutor aziendale ha consigliato di trovare un wrapper della libreria che potesse essere integrato tramite una dipendenza maven. Per cui è stata utilizzata la libreria *openpnp*, aggiungibile al pom tramite la seguente dipendenza[2] (Listing 4.13).

```
<dependency>
  <groupId>org.openpnp</groupId>
  <artifactId>opencv</artifactId>
  <version>3.4.2-1</version>
</dependency>
```

Listing 4.13: Dependency openpnp.

Diamo di seguito l'elenco delle tipologie di conversioni applicate[11]:

Conversione in bianco e nero (Figura 4.3, Listing 4.14)

Questa è la conversione più importante, spesso è sufficiente solo questa conversione per una buona lettura del documento. Molti tipi di conversione richiedono in ingresso un immagine gestita con questa conversione, e comunque in generale è sempre meglio passare in input un immagine convertita con questo filtro.

Figura 4.3: Esempio di conversione in bianco e nero

```
//Creating the empty destination matrix
Mat dst = new Mat();
//Converting the image to grey scale
Imgproc.cvtColor(src, dst, Imgproc.COLOR_RGB2GRAY);
Imgcodecs.imwrite("colortogreyscale.jpg", dst);
```

Listing 4.14: Implementazione della conversione in bianco e nero.

Conversione Threshold (Figura 4.4, Listing 4.15)

Esistono diversi tipi di conversioni di tipo threshold. Nella funzione è stato utilizzato il *Threshold semplice Inv*. Consiste nel convertire le parti nere di un immagine in bianco e convertite ogni altro colore in nero. Particolarità di questa conversione è che vuole in input un immagine già convertita in bianco e nero (Sezione 4.3.3).

Figura 4.4: Esempio di conversione simple threshold inv.

```
file ="colortogreyscale.jpg";
src = Imgcodecs.imread(file);
dst = new Mat();
Imgproc.threshold(src, dst, 50, 80, Imgproc.THRESH_BINARY_INV);
Imgcodecs.imwrite("thresh.jpg", dst);
```

Listing 4.15: Implementazione della conversione in simple threshold inv.

Conversione Morfologica (Figura 4.5, Listing 4.16)

Esistono diversi tipi di conversioni morfologiche, nella funzione è stata utilizzata una conversione morfologica denominata *BlackHat*. BlackHat viene utilizzata per migliorare gli oggetti scuri inseriti su uno sfondo luminoso.

Figura 4.5: Esempio di conversione morfologica di tipo blackhat

```
String file ="colortogreyscale.jpg";
src = Imgcodecs.imread(file);
dst = new Mat(src.rows(), src.cols(), src.type());
Mat kernel = Mat.ones(5,5, CvType.CV_32F);
//Applying dilate on the Image
Imgproc.morphologyEx(src, dst, Imgproc.MORPH_BLACKHAT, kernel);
Imgcodecs.imwrite("blackhat.jpg", dst);
```

Listing 4.16: Implementazione della conversione blackhat.

Filtro per gestire il contrasto (Listing 4.17)

Questo filtro gestisce la luminosità di un immagine, in particolare, ne aumenta la luminosità. Nella funzione viene applicato su una trasformazione di tipo BlackHat (Sezione 4.3.3).

```
//Aumento Contrasto
file ="blackhat.jpg";
src = Imgcodecs.imread(file, Imgcodecs.IMREAD_COLOR);
//Creating an empty matrix
Mat dest = new Mat(src.rows(), src.cols(), src.type());
//Increasing the contrast of the image
src.convertTo(dest, -1,1.8, 0);
// Displaying the image
Imgcodecs.imwrite("contrast.jpg", dest);
```

Listing 4.17: Filtro per aumento contrasto.

4.3.4 Classe di Utils

Per raccogliere tutti i metodi, utili alla gestione della lettura di un'immagine, viene creata una classe di Utils, in cui sono presenti i metodi utili alla decodifica del codice MRZ_G e della conversione delle immagini.
Nella classe di Utils sono presenti anche i seguenti metodi:

- **readMRZ** (Listing 4.18): il metodo setta la cartella che contiene i dizionari e le impostazioni di Tesseract tramite il comando *tesseract.setDatapath(sourcePath)* e il motore di Tesseract, tramite il comando *tesseract.setOcrEngineMode(1)*. Successivamente tramite il metodo *manipulateImage(src)* (Listing 4.14, Listing 4.15, Listing 4.16, Listing 4.17) viene effettuata la conversione delle immagini.
  Il metodo *readMRZ*, passando in input un dizionario e un immagine, tramite il metodo *readWithOCR()* (Listing 4.27), restituisce una lettura dell'immagine. Qualora il codice di controllo non sia corretto, il metodo prevede altri 4 scenari in cui passandogli dizionari e immagini elaborate con diversi filtri, va a fornire una lettura diversa del documento. Queste comprendono un alternanza tra un immagine in bianco e nero (Sezione 4.3.3), un tipo di dizionario denominato *ocrbint*[6] e un immagine in bianco e nero (Sezione 4.3.3), convertita con il filtro BlackHat (Sezione 4.3.3) aumentata di contrasto (Sezione 4.3.3) e un secondo dizionario denominato *mrz*[5];

```
public static HashMap<String,String> readMRZ(Mat src,String
    sourcePath, int scenario){
  IDInfo = new HashMap<String, String>();
  tesseract.setDatapath(sourcePath); // path to tessdata directory
  tesseract.setOcrEngineMode(1);
```

```
  switch(scenario) {
    case 0: // immagine contrasto , lettura con ocrbint
      manipulateImage(src);
      imageFile = new File("contrast.jpg");
      tesseract.setLanguage("ocrb_int");
      IDInfo=readWithOCR(tesseract,imageFile,src);
    break;
    case 1: // immagine bianco e nero, lettura con ocrbint
      imageFile = new File("colortogreyscale.jpg");
      tesseract.setLanguage("ocrb_int");
      IDInfo=readWithOCR(tesseract,imageFile,src);
    break;
    case 2:// immagine contrasto , lettura con mrz
      imageFile = new File("contrast.jpg");
      tesseract.setLanguage("mrz");
      IDInfo=readWithOCR(tesseract,imageFile,src);
    break;
    case 3: // immagine bianco e nero, lettura con mrz
      imageFile = new File("colortogreyscale.jpg");
      tesseract.setLanguage("mrz");
      IDInfo=readWithOCR(tesseract,imageFile,src);
    break;
    default:
      manipulateImage(src);
      imageFile = new File("contrast.jpg");
      tesseract.setLanguage("ocrb_int");
      IDInfo=readWithOCR(tesseract,imageFile,src);
    break;}
  if((IDInfo==null || badControl) && scenario<4 ) {
    scenario=scenario+1;
    badControl=false;
    readMRZ(src,sourcePath,scenario);}
  return IDInfo;}
```

Listing 4.18: readMRZ method.

* **readMRZFile** (Listing 4.19): il metodo si comporta in maniera analoga al metodo readMRz presente nel Listing 4.3.4, ma è pensato per la lettura dei file pdf. Inoltre, tramite il metodo *BufferedImage2Mat*, la BufferedImage viene convertita in una matrice Mat. Nel metodo *ReadMRZFile* è presente un'alternanza tra un immagine in bianco e nero, un tipo di dizionario denominato *ocrbint*[6] e un immagine in bianco e nero (Sezione 4.3.3), convertita con il Threshold semplice inv (Sezione 4.3.3) e un secondo dizionario denominato *mrz*[5];

```
public static HashMap<String, String> readMRZFile(BufferedImage
    image,String sourcePath,int scenario) throws IOException {
  IDInfo = new HashMap<String, String>();
  Mat src= BufferedImage2Mat(image);
  tesseract.setDatapath(sourcePath); // path to tessdata directory
  tesseract.setOcrEngineMode(1);
  switch(scenario) {
    case 0: // immagine threeshold , lettura con ocrbint
      manipulateImage(src);
      imageFile = new File("thres.jpg");
      tesseract.setLanguage("ocrb_int");
      IDInfo=readWithOCR(tesseract,imageFile,src);
    break;
    case 1: // immagine bianco e nero, lettura con ocrbint
      imageFile = new File("colortogreyscale.jpg");
      tesseract.setLanguage("ocrb_int");
```

```
        IDInfo=readWithOCR(tesseract,imageFile,src);
      break;
      case 2:// immagine threeshold , lettura con mrz
        imageFile = new File("thres.jpg");
        tesseract.setLanguage("mrz");
        IDInfo=readWithOCR(tesseract,imageFile,src);
      break;
      case 3: // immagine bianco e nero, lettura con mrz
        imageFile = new File("colortogreyscale.jpg");
        tesseract.setLanguage("mrz");
        IDInfo=readWithOCR(tesseract,imageFile,src);
      break;
      default:
        manipulateImage(src);
        imageFile = new File("thres.jpg");
        tesseract.setLanguage("ocrb_int");
        IDInfo=readWithOCR(tesseract,imageFile,src);
      break;
    }
    if((IDInfo==null || badControl) && scenario<4 ) {
      scenario=scenario+1;
      badControl=false;
      readMRZFile(image,sourcePath,scenario);
    }
    return IDInfo;
}
```

Listing 4.19: readMRZFile method.

* **mrzIdentify** (Listing 4.20): il metodo si occupa di creare una mappa di caratteri. Successivamente riconosce se il documento è una carta d'identità elettronica o un passaporto;

```
public static HashMap<String,String> mrzIdentify(String mrz) throws
    ParseException  {
  HashMap<String,String> IDInfo = new HashMap<String,String>();
  HashMap<Character,Integer> carachterMap = new HashMap<Character,
      Integer>();
  carachterMap.put('<',0);carachterMap.put('0',0);carachterMap.put('
      1',1);
  carachterMap.put('2',2);carachterMap.put('3',3);carachterMap.put('
      4',4);
  carachterMap.put('5',5);carachterMap.put('6',6);carachterMap.put('
      7',7);
  carachterMap.put('8',8);carachterMap.put('9',9);carachterMap.put('
      A',10);
  carachterMap.put('B',11);carachterMap.put('C',12);carachterMap.put
      ('D',13);
  carachterMap.put('E',14);carachterMap.put('F',15);carachterMap.put
      ('G',16);
  carachterMap.put('H',17);carachterMap.put('I',18);carachterMap.put
      ('J',19);
  carachterMap.put('K',20);carachterMap.put('L',21);carachterMap.put
      ('M',22);
  carachterMap.put('N',23);carachterMap.put('O',24);carachterMap.put
      ('P',25);
  carachterMap.put('Q',26);carachterMap.put('R',27);carachterMap.put
      ('S',28);
  carachterMap.put('T',29);carachterMap.put('U',30);carachterMap.put
      ('V',31);
  carachterMap.put('W',32);carachterMap.put('X',33);carachterMap.put
      ('Y',34);
```

```
  carachterMap.put('Z',35);

  if(mrz.substring(0,1).equals("C")  && IDInfo.get("tipoDocumento")
      ==null ) {
          IDInfo= ciParser(mrz,IDInfo,carachterMap);
        }
        else if(mrz.substring(0,1).equals("P") && IDInfo.get("
            tipoDocumento")==null ) {
          IDInfo= pParser(mrz,IDInfo,carachterMap);
        }
        return IDInfo;
}
```

Listing 4.20: mrzIdentify method.

* **ciParser** (Listing 4.21): il metodo serve per interpretare in modo corretto, grazie alla standardizzazione nel posizionamento dei caratteri, il codice MRZ_G di una CIE_G. Viene utilizzato anche il metodo chiamato *cognomeNome()* utile a leggere il nome e cognome in maniera corretta. Viene utilizzato anche il metodo chiamato *controlCode()* (Listing 4.3.4) che verifica vari codici di controllo;

```
public static HashMap<String,String> ciParser(String mrz,HashMap<
    String,String> IDInfo,HashMap<Character,Integer> carachterMap) {
  //Tipologia del documento: carta d'identita o passaporto
  IDInfo.put("tipoDocumento", "CARTA_IDENTITA");
  //Nazione emettitrice del documento
  if(mrz.substring(2, 5).equals("ITA")) IDInfo.put("nazioneDocumento
      ", "Italia");
    else IDInfo.put("nazioneDocumento", "Infomrazione non ben letta"
        );
  //Numero del documento + controllo codice controllo
  if(controlCode(mrz.substring(5,15),IDInfo,carachterMap)) IDInfo.
      put("numeroDocumento", mrz.substring(5,14));
    else IDInfo.put("numeroDocumento", "Numero documento mal letto."
        );
  //Data di nascita + controllo codice controllo
  if(controlCode(mrz.substring(31,38),IDInfo,carachterMap)) {
    String birthDate= mrz.substring(35,37).concat("/").concat(mrz.
        substring(33,35)).concat("/").concat(mrz.substring(31,33));
    SimpleDateFormat dateFormat = new SimpleDateFormat("dd/MM/yy");
    try {
      Date date = dateFormat.parse(birthDate);
      Format formatter = new SimpleDateFormat("dd/M/yyyy");
      birthDate = formatter.format(date);
    } catch (ParseException e) {
      e.printStackTrace();
    }
    IDInfo.put("dataNascita",birthDate);
  }
  else IDInfo.put("dataNascita","Data di nascita mal letta");
  //Sesso
    IDInfo.put("sesso",mrz.substring(38,39));
    //Data scadenza della carta + controllo codice controllo
    if(controlCode(mrz.substring(39,46),IDInfo,carachterMap)) {
      String expiredDate= mrz.substring(43,45).concat("/").concat(
          mrz.substring(41,43)).concat("/").concat(mrz.substring
          (39,41));
      SimpleDateFormat dateFormat = new SimpleDateFormat("dd/MM/yy")
          ;
      try {
        Date date = dateFormat.parse(expiredDate);
```

```
        Format formatter = new SimpleDateFormat("dd/M/yyyy");
        expiredDate = formatter.format(date);
      } catch (ParseException e) {
        e.printStackTrace();
      }
      IDInfo.put("dataScadenzaDocumento",expiredDate);
  }
    else IDInfo.put("dataScadenzaDocumento","Data di scadenza della
        carta mal letta");
      //Nazionalita
      if(mrz.substring(46,49).equals("ITA") || mrz.substring(46,49).
          equals("1TA")) IDInfo.put("nazionalita", "Italiana");
        else IDInfo.put("nazionalita", "Infomrazione non ben letta");
  //Controllo su tutte le cifre di controllo
      String bigControl= mrz.substring(5,15).concat(mrz.substring
          (31,38)).concat(mrz.substring(39,46)).concat(mrz.substring
          (60,61));
    if(!controlCode(bigControl,IDInfo,carachterMap)) badControl=true
        ;
    //Stampo nome e cognome, anche multipli cognome multiplo
        separato da - , nome da spazio
    cognomeNome(mrz,IDInfo);
  //Ritorno la mappa
    return IDInfo;
}
```

Listing 4.21: ciParser method.

* **pParser** (Listing 4.22): il metodo serve per interpretare in modo corretto, grazie alla standardizzazione nel posizionamento dei caratteri, il codice MRZ di un passaporto. Il metodo *pParser* è analogo al metodo *ciParser* (Listing 4.21), con l'unica eccezione che viene usato per la lettura del passaporto;

```
public static HashMap<String,String> pParser(String mrz,HashMap<
    String,String> IDInfo,HashMap<Character,Integer> carachterMap) {
  //Tipologia del documento: cartd d'identita o passaporto
  IDInfo.put("tipoDocumento", "PASSAPORTO");

  //Nazione emettitrice del documento
  if(mrz.substring(2, 5).equals("ITA")) IDInfo.put("nazioneDocumento
      ", "Italia");
  else IDInfo.put("nazioneDocumento", "Infomrazione non ben letta");

  //Stampo nome e cognome, anche multipli cognome multiplo separato
      da - , nome da spazio
  cognomeNomeP(mrz,IDInfo);

  //Numero del documento + controllo codice controllo
    if(controlCode(mrz.substring(45,55),IDInfo,carachterMap)) IDInfo
        .put("numeroDocumento", mrz.substring(45,54));
    else IDInfo.put("numeroDocumento", "Numero documento mal letto."
        );

    //Nazionalita
    if(mrz.substring(55,58).equals("ITA") || mrz.substring(55,58).
        equals("1TA")) IDInfo.put("nazionalita", "Italiana");
    else IDInfo.put("nazionalita", "Infomrazione mal letta");


    //Data di nascita + controllo codice controllo
    if(controlCode(mrz.substring(58,65),IDInfo,carachterMap)) {
```

```
    String birthDate= mrz.substring(62,64).concat("/").concat(mrz.
        substring(60,62)).concat("/").concat(mrz.substring(58,60));
    SimpleDateFormat dateFormat = new SimpleDateFormat("dd/MM/yy");
    try {
      Date date = dateFormat.parse(birthDate);
        Format formatter = new SimpleDateFormat("dd/M/yyyy");
        birthDate = formatter.format(date);
      } catch (ParseException e) {
        e.printStackTrace();
      }
      IDInfo.put("dataNascita",birthDate);
    }
    else IDInfo.put("dataNascita","Data di nascita mal letta");

    //Sesso
      IDInfo.put("sesso",mrz.substring(65,66));

     //Data scadenza documento + controllo codice controllo
     if(controlCode(mrz.substring(66,73),IDInfo,carachterMap)) {
      String expiredDate= mrz.substring(70,72).concat("/").concat(
          mrz.substring(68,70)).concat("/").concat(mrz.substring
          (66,68));
        SimpleDateFormat dateFormat = new SimpleDateFormat("dd/MM/yy
            ");
      try {
        Date date = dateFormat.parse(expiredDate);
        Format formatter = new SimpleDateFormat("dd/M/yyyy");
        expiredDate = formatter.format(date);
      } catch (ParseException e) {
        e.printStackTrace();
      }
        IDInfo.put("dataScadenzaDocumento",expiredDate);
      }
      else IDInfo.put("dataScadenzaDocumento","Data di scadenza
          della carta mal letta");

      //Controllo su tutte le cifre di controllo
      String bigControl= mrz.substring(45,55).concat(mrz.substring
          (58,65)).concat(mrz.substring(66,73)).concat(mrz.substring
          (mrz.length()-2,mrz.length()-1));
      if(!controlCode(bigControl,IDInfo,carachterMap)) badControl=
          true;

   //Ritorno la mappa
     return IDInfo;
}
```

Listing 4.22: pParser method.

* **controlCode** (Listing 4.23): metodo che implementa l'algoritmo di controllo descritto nella Sezione 4.3.1;

```
public static boolean controlCode(String mrz, HashMap<String,String>
     IDInfo, HashMap<Character,Integer> carachterMap) {
  char controlChar;
  int[] checkprod=new int[]{7,3,1};
  int[] trasformedLetterToNumber= new int[mrz.length()-1];
  int[] multiplyResult= new int[mrz.length()-1];
  for (int j=0 ;j < mrz.length()-1;j++) {
    controlChar = mrz.charAt(j);
      trasformedLetterToNumber[j]=carachterMap.get(controlChar);
  }
  int j=0,sum=0;
  while(j< mrz.length()-1) {
    for(int k=0; k<3; k++) {
      multiplyResult[j] = trasformedLetterToNumber[j] * checkprod[k
          ];
        j++;
      }
  }
  for(int k=0;k<mrz.length()-1;k++) sum+=multiplyResult[k];
  int controlDigit= sum\%10;
  if(controlDigit == Integer.parseInt(mrz.substring(mrz.length()-1,
      mrz.length()))) return true;
  else return false;
}
```

Listing 4.23: controlCode method.

* **cognomeNome** (Listing 4.24): metodo che si occupa di leggere e definire Nome e Cognome per la carta d'identità. Essendo nome e cognome gli unici dati che hanno una lunghezza variabile, si è scelto di gestire il nome e cognome con un metodo specifico. La gestione del nome e del cognome è definita sempre nel documento $ICAO_G$[1] 9303: Il nome del titolare è generalmente rappresentato in due parti: l'identificatore primario e l'identificatore secondario. Lo Stato o l'organizzazione di emissione, stabilisce quale parte del nome è l'identificatore primario. Questo potrebbe essere il cognome di famiglia, il cognome da nubile o da sposata, ecc. Nel documento si consiglia di utilizzare caratteri maiuscoli, tranne nel caso di un prefisso, ad es. "von", "Mc" o "de la", per questi casi è appropriato un misto di maiuscolo e minuscolo. Le parti rimanenti del nome sono l'identificatore secondario. Questi possono essere i nomi, i nomi familiari, iniziali o altri nomi secondari. Questi nomi devono essere scritti nel campo dell'identificatore secondario. Se viene utilizzato un unico campo per il nome, l'identificatore secondario deve essere separato dall'identificatore primario da una virgola singola (,). Se i caratteri nazionali non sono di origine latina, deve essere prevista la traslitterazione in caratteri latini;

```
public static void cognomeNome(String mrz,HashMap<String,String>
    IDInfo) {
  String nome="", cognome="";
  int i=62;
  while(i<mrz.length()-1) {
    if(mrz.charAt(i)!= '<' && i!=mrz.length()-1) {
      if(mrz.charAt(i)!=' ' && mrz.charAt(i)<=90 && mrz.charAt(i)
          >=65) {
          cognome+=mrz.charAt(i);
          i++;
```

```
        }
      }
      else {
        if(mrz.charAt(i+1)!= '<') {
          cognome+="-";
          i++;
        }
        if(mrz.substring(i,i+2).equals("<<") && mrz.substring(i+2,i
            +3)!="<") {
          i+=2;
          while(i<mrz.length()-1) {
            if(mrz.charAt(i)!= '<' && i!=mrz.length()-1 && mrz.
                charAt(i)!= ' ' &&
            mrz.charAt(i)<=90 && mrz.charAt(i)>=65) {
              if(mrz.charAt(i)!=' ') {
                nome+=mrz.charAt(i);
                  i++;}
            }
              else {
                if(mrz.charAt(i+1)!= '<' && i!=mrz.length()-2 && mrz
                    .charAt(i+1)!= ' ' &&
                mrz.charAt(i)<=90 && mrz.charAt(i)>=65) {
                  nome+=" ";
                  i++;}
                i++;  }}}}}
  IDInfo.put("cognome",cognome.substring(0,1)+cognome.substring(1).
      toLowerCase());
  IDInfo.put("nome",nome.substring(0,1)+nome.substring(1).
      toLowerCase());}
```

Listing 4.24: cognomeNome method.

* **cognomeNomeP** (Listing 4.25): il metodo è analogo al metodo *cognomeNome* presente nel Listing 4.3.4, ma si occupa di leggere e definire Nome e Cognome per il passaporto;

```
public static void cognomeNomeP(String mrz,HashMap<String,String>
    IDInfo) {
  String nome="", cognome="";
  int i=5;
  while(i<mrz.length()/2) {
    if(mrz.charAt(i)!= '<' && i!=mrz.length()/2) {
      if(mrz.charAt(i)!=' ' && mrz.charAt(i)<=90 && mrz.charAt(i)
          >=65) {
        cognome+=mrz.charAt(i);
        i++;
      }
    }
    else {
      if(mrz.charAt(i+1)!= '<') {
        cognome+="-";
        i++;
      }
      if(mrz.substring(i,i+2).equals("<<") && mrz.substring(i+2,i+3)
          !="<") {
        i+=2;
        while(i<mrz.length()/2) {
          if(mrz.charAt(i)!= '<' && i!=mrz.length()/2 && mrz.charAt(
              i)!= ' ' &&
          mrz.charAt(i)<=90 && mrz.charAt(i)>=65) {
            if(mrz.charAt(i)!=' ') {
              nome+=mrz.charAt(i);
```

```
                    i++;
                  }
                }
                else {
                  if(mrz.charAt(i+1)!= '<' && i!=mrz.length()/2 && mrz
                      .charAt(i+1)!= ' ' &&
                mrz.charAt(i)<=90 && mrz.charAt(i)>=65) {
                    nome=nome+" "+mrz.charAt(i+1);
                    i++;
                  }
                  i++;
                }}}}}
  IDInfo.put("cognome",cognome.substring(0,1)+cognome.substring(1).
      toLowerCase());
  IDInfo.put("nome",nome.substring(0,1)+nome.substring(1).
      toLowerCase());}
```

Listing 4.25: cognomeNomeP method.

∗ **compressByte e decompressByte** (Listing 4.26): metodi deputati alla compressione e decompressione dei file in ingresso, utili per una risposta dell'applicativo più veloce;

```
public static byte[] compressBytes(byte[] data) {
  Deflater deflater = new Deflater();
  deflater.setInput(data);
  deflater.finish();
  ByteArrayOutputStream outputStream = new ByteArrayOutputStream(
      data.length);
  byte[] buffer = new byte[1024];
  while (!deflater.finished()) {
    int count = deflater.deflate(buffer);
    outputStream.write(buffer, 0, count);}
  try {
    outputStream.close();
  } catch (IOException e) {}
  return outputStream.toByteArray();}

public static byte[] decompressBytes(byte[] data) {
  Inflater inflater = new Inflater(); inflater.setInput(data);
  ByteArrayOutputStream outputStream = new ByteArrayOutputStream(
      data.length);
  byte[] buffer = new byte[1024];
  try {
    while (!inflater.finished()) {
      int count = inflater.inflate(buffer);
      outputStream.write(buffer, 0, count);}
    outputStream.close();
  } catch (IOException ioe) {} catch (DataFormatException e) {}
  return outputStream.toByteArray();}
```

Listing 4.26: compressByte e decompressByte method.

∗ **readWithOCR** (Listing 4.27): il metodo si occupa di fare la lettura di un file tramite il comando *ImageIO.read(imageFile)*. Una volta eseguita la lettura attraverso il metodo *tesseract.doOCR(image)*, viene restituita una stringa di caratteri contenente l'elaborazione dell'immagine. Questa, in base al tipo di documento (CIE$_G$ o passaporto), farà partire l'identificazione dell'MRZ$_G$, tramite *mrzIdentify()*;

```
public static HashMap<String,String> readWithOCR(ITesseract
    tesseract, File imageFile,Mat src){
          try{
  IDInfo=null;  BufferedImage image = ImageIO.read(imageFile);
  String result = tesseract.doOCR(image);
  for(int m=0,l=2; l<result.length(); m++,l++) {
    if(result.substring(m, l).equals("C<")) { IDInfo = mrzIdentify(
        result.substring(m,m+93));}
    if(result.substring(m, l).equals("P<")) { IDInfo = mrzIdentify(
        result.substring(m,result.length()));}
    if(result.substring(m, l+1).equals("CII") && l+1 < result.length
        ()) {
      IDInfo = mrzIdentify(result.substring(m,m+93));}}
  }catch (Exception e) {return IDInfo=null;}return IDInfo;}
```

Listing 4.27: readWithOCR method.

* **BufferedImageToMat** (Listing 4.28): metodo per creare una Mat (matrice di OpenCV) da una BufferedImage.

```
public static Mat BufferedImageToMat(BufferedImage image) throws
    IOException {nu.pattern.OpenCV.loadLocally();
ByteArrayOutputStream byteArrayOutputStream = new
    ByteArrayOutputStream();
ImageIO.write(image, "jpg", byteArrayOutputStream);
    byteArrayOutputStream.flush();
return Imgcodecs.imdecode(new MatOfByte(byteArrayOutputStream.
    toByteArray()), Imgcodecs.IMREAD_ANYCOLOR);}
```

Listing 4.28: BufferedImageToMat method.

4.3.5 Codifica front-end

Verrà mostrata la codifica e le maschere del front-end$_G$ utili alla registrazione di una nuova company nel sistema. Il front-end$_G$ è stato scritto utilizzando il framework$_G$ Angular$_G$. Di seguito verrà mostrata una registrazione di una nuova company, come descritto nello scopo dello stage (Sezione 2.2.1). Si porrà particolare attenzione a descrivere l'implementazione della funzione creata e non delle altre parti che non sono oggetto dello stage.
Quando si vuole registrare una nuova company, comparirà la seguente maschera (Figura 4.6 non compilata, Figura 4.7 compilata). In tutte le form sono presenti dei validatori che controllano la correttezza delle informazioni inserite. Per esempio controlla che la mail sia in un formato appropriato e che tutti i campi obbligatori sono inseriti.

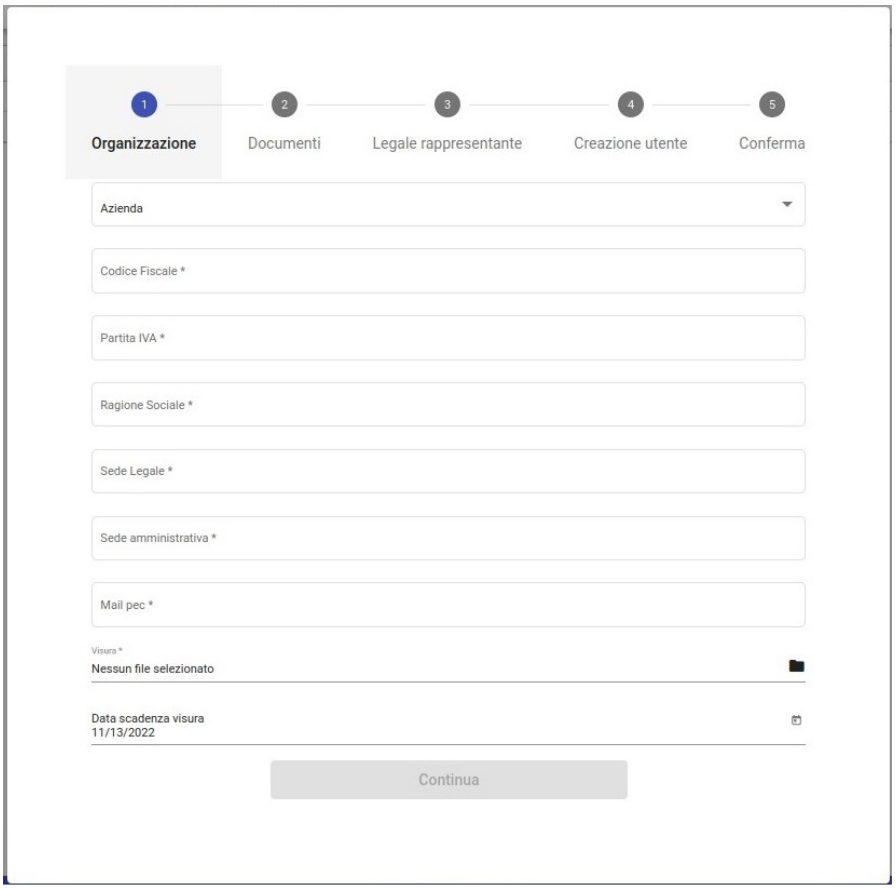


Figura 4.6: Prima schermata registrazione nuova company non compilata.

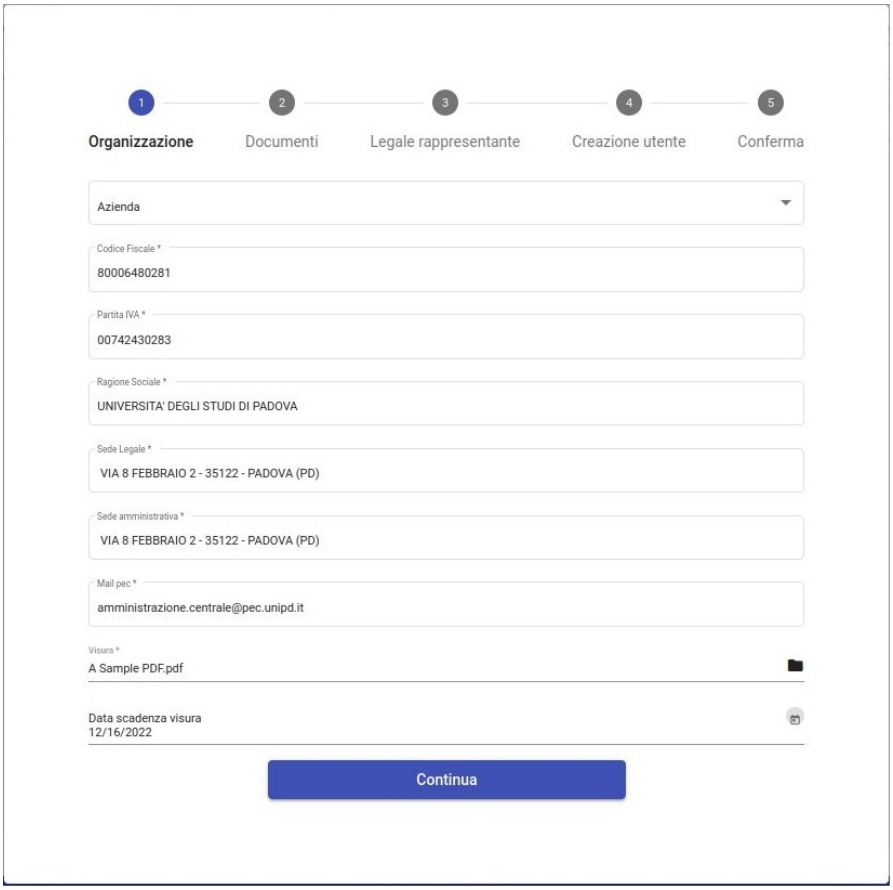


Figura 4.7: Prima schermata registrazione nuova company compilata.

Le prossime immagini (Figura 4.8, Figura 4.9, Figura 4.10), mostrano le maschere relative all'inserimento e alla lettura di un documento di riconoscimento. Nella Figura 4.8, viene chiesto di caricare un documento di identità, in formato immagine o pdf. Successivamente, nel momento stesso in cui ne viene confermata la selezione, viene fatta la chiamata al back-end$_G$ per leggere, estrarre e disporre le informazioni dal documento caricato.
Durante l'elaborazione, è presente uno spinner (Figura 4.9), che blocca ogni operazione fintantoché la decodifica del documento non ha dato un esito. Nel caso in cui qualche dato del documento venga letto o interpretato in modo anomalo, viene segnalato all'utente, richiedendo di inserire manualmente i dati necessari alla registrazione.
In seguito a un elaborazione esatta del documento vengono popolati i campi della form (Figura 4.10).

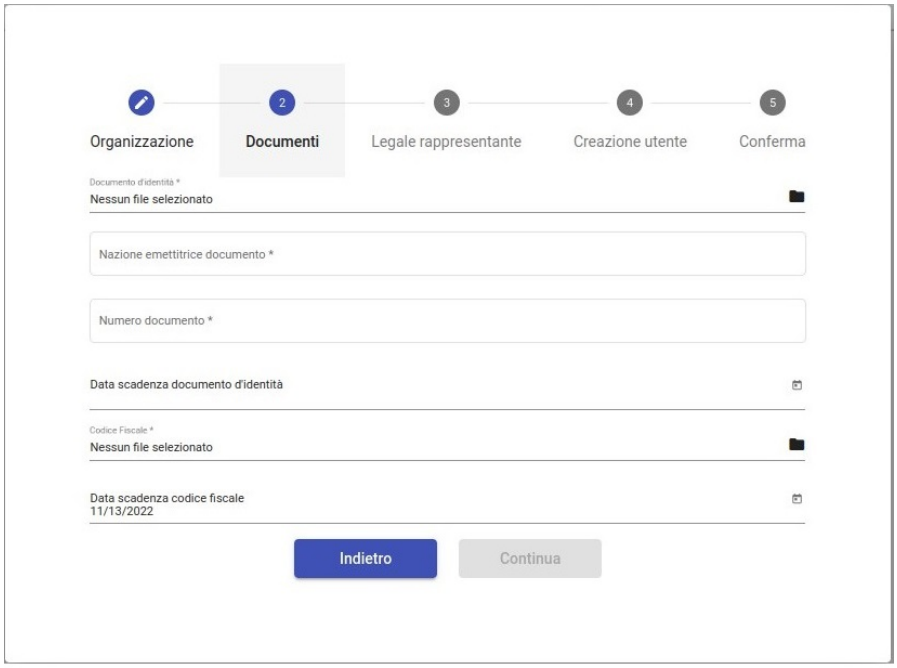


Figura 4.8: Schermata di inserimento documento.

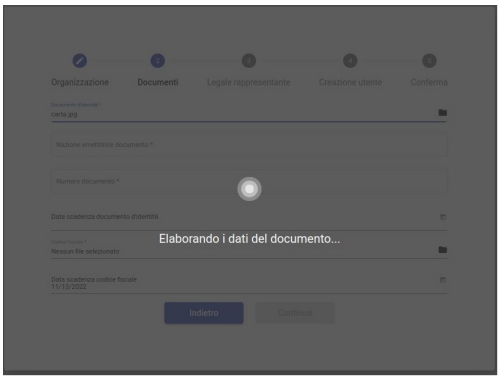


Figura 4.9: Spinner di elaborazione: è attivo finché l'elaborazione non termina.

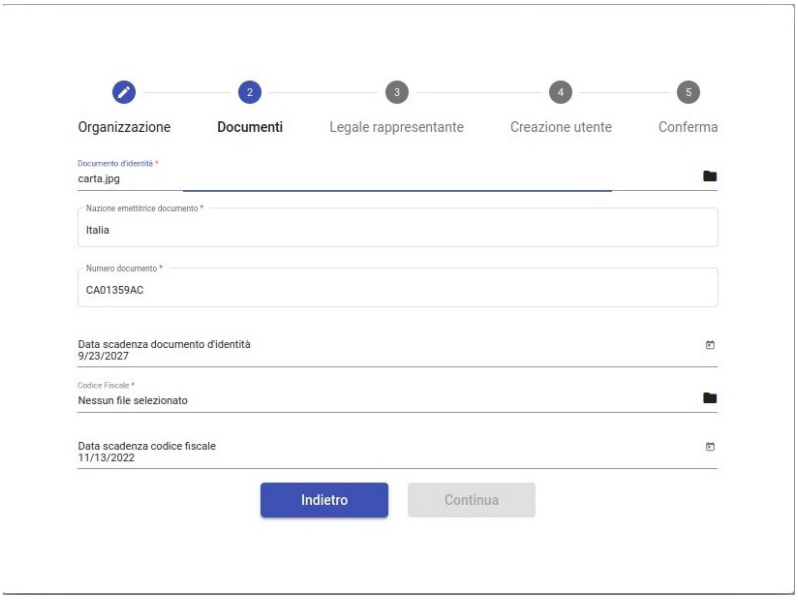


Figura 4.10: Form popolata con i risultati dell'elaborazione.

Di seguito i listing (Listing 4.29, Listing 4.30, Listing 4.31) contenenti il codice realizzato.
upload (Listing 4.29) è il metodo che contiene la chiamata al back-end$_G$. Passa in ingresso, tramite una richiesta POST, un url e un body contenente il file in input. Il metodo restituisce una risposta in formato Json.

```
1 public upload(file:File): Observable<any>{
    const uploadImageData = new FormData();
    uploadImageData.append('imageFile', file, file.name);
    return this.http.post('/attachment/decode-image', uploadImageData, {
        observe: 'response', responseType:'json'});}
```

Listing 4.29: Chiamata al controller del back-end$_G$.

Il metodo *onFileChanged()* (Listing 4.30) prende il file in input, lo salva in una variabile denominata *selectedFile* e infine chiama il metodo *onUpload()* (Listing 4.31).

```
onFileChanged(event:any){
    this.selectedFile = event.target.files[0];
    this.onUpload();}
```

Listing 4.30: Metodo che viene invocato alla selezione di un file.

Il metodo onUpload() (Listing 4.31), si occupa di chiamare il metodo *upload* (Listing 4.29). *upload* farà la chiamata al back-end$_G$ e restituirà la risposta. Questa risposta viene gestita dal *.subscribe()*. Il subscribe prende la risposta e ne inserisce i vari valori nei campi della form. Successivamente, una volta terminata l'elaborazione viene settato il timeout dello spinner, che finisce con l'elaborazione del documento o dopo 10 secondi. Nel caso l'elaborazione del documento impieghi più tempo lo spinner scomparirà ma il programma continuerà l'elaborazione.

```
onUpload(){
  this.spinner.show();
  this.companyService.upload(this.selectedFile).subscribe((response)=>{
    this.docsLegaleRappresentanteForm.get('tipoDocumentoIdentita').
        setValue(response.body.tipoDocumento);
    this.docsLegaleRappresentanteForm.get('nazioneDocumento').setValue(
        response.body.nazioneDocumento);
    this.docsLegaleRappresentanteForm.get('numeroDocumento').setValue(
        response.body.numeroDocumento);
    this.legaleRappresentanteForm.get('nomeLegaleRappresentante').
        setValue(response.body.nome);
    this.legaleRappresentanteForm.get('cognomeLegaleRappresentante').
        setValue(response.body.cognome);
    const [day, month, year] = response.body.dataNascita.split('/');
    this.legaleRappresentanteForm.get('dataNascitaLegaleRappresentante')
        .setValue(new Date(+year,+month -1, +day ));
    this.legaleRappresentanteForm.get('sessoLegaleRappresentante').
        setValue(response.body.sesso);
    this.legaleRappresentanteForm.get('nazionalitaLegaleRappresentante')
        .setValue(response.body.nazionalita);
    const [eday, emonth, eyear] = response.body.dataScadenzaDocumento.
        split('/');
    this.docsLegaleRappresentanteForm.get('scadenzaDocumentoIdentita').
        setValue(new Date(+eyear,+emonth -1, +eday ));
    this.spinner.hide();});
  setTimeout(() => {this.spinner.hide();}, 10000);}
```

Listing 4.31: onUpload().

In questa finestra (Figura 4.11) si può vedere come i dati letti precedentemente abbiano popolato anche questo form. I campi non valorizzati sono la provincia di nascita, il luogo di nascita e il codice fiscale, dati non recuperabili dal codice MRZ$_G$. Nella Figura 4.12, si può vedere che è presente un pulsante per il calcolo automatico del codice fiscale, previa compilazione manuale dei campi non presenti nel codice MRZ$_G$.

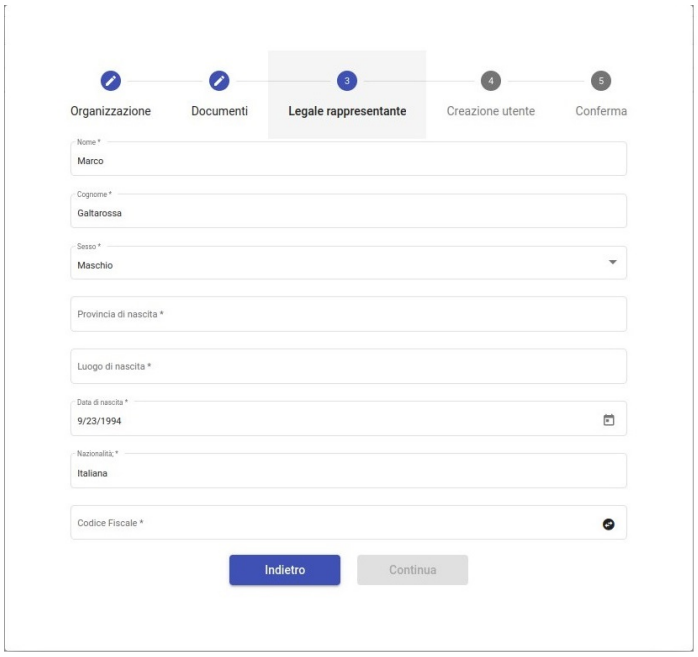


Figura 4.11: Form con altri dati letti dall'MRZ.

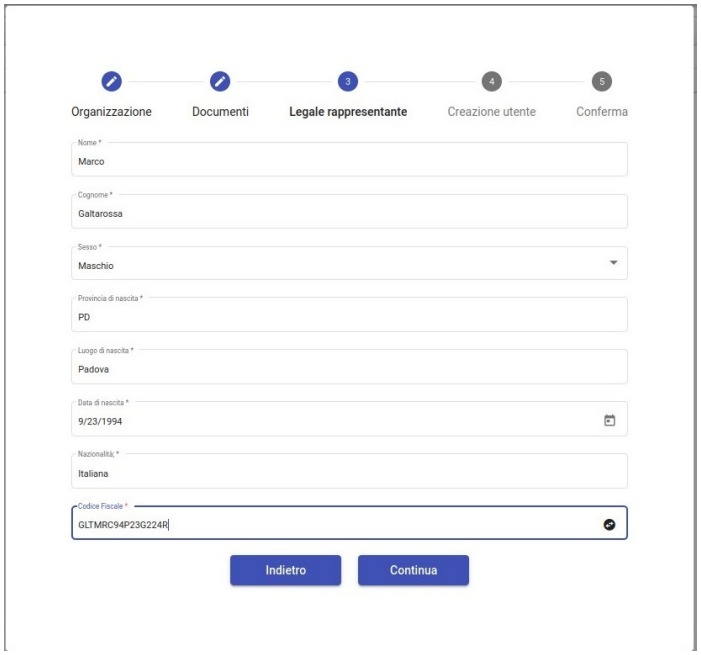


Figura 4.12: Form con altri dati letti dall'MRZ più gli altri dati non leggibili dal codice.

Si prosegue inserendo la mail e confermandola (Figura 4.13, Figura 4.14).

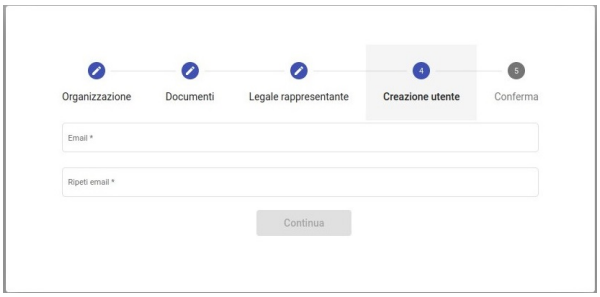


Figura 4.13: Form di inserimento e conferma mail

Figura 4.14: Mail inserita e confermata

L'ultima operazione è la conferma di tutti i dati inseriti (Figura 4.15).

Figura 4.15: Conferma della registrazione.

4.4 Parte seconda: registrazione e accesso ai servizi tramite CIE

All'inizio dello stage si prevedeva di integrare il processo di autenticazione "Entra con CIE", un servizio di autenticazione, la cui attivazione va richiesta allo Stato. L'azienda ne ha richiesto l'attivazione ma per le tempistiche lunghe non è stato possibile sviluppare appieno questa parte dello stage. Avendo però a disposizione un lettore di carte (CyberJack RFID Basic) abilitato alla lettura delle CIE_G, è stato chiesto dal tutor aziendale, di capire quali informazioni si possono ricavare dalla lettura tramite lettore di carte.
Lo Stato mette a disposizione degli SDK_G utili alla lettura di codici presenti nelle CIE_G[14]. In particolare, un SDK_G denominato ***cie-nis-java-sdk***, si occupa di estrarre il NIS_G (Numero Identificativo Servizi) e l'$hash_G$ della CIE_G .

4.4.1 Servizi Pinless CIE: cie-nis-java-sdk

Cie-Nis-java-sdk[3] (esempio implementazione nel Listing 4.32) è una SDK_G sviluppata in linguaggio Java che permette di avviare un protocollo di verifica del NIS_G, associato ad ogni CIE_G. Il NIS_G è univoco per ogni CIE_G, è a lettura libera e non è riconducibile direttamente al titolare della stessa. La SDK_G consente di controllare l'autenticità e l'originalità della CIE_G e di convalidare il NIS_G. È possibile così utilizzare la CIE_G per un servizio pinless, ovvero un servizio che non necessita del PIN per la lettura della carta.
Un servizio pinless tramite CIE_G permette di utilizzare la carta come strumento unico di accesso a molteplici servizi che richiedono un'abilitazione alla fruizione degli stessi, come accesso a mezzi di trasporto, luoghi di lavoro, luoghi pubblici, etc. L'abilitazione per la fruizione del servizio si compone di due fasi: una di Enrollment e una di Accesso. La fase di Enrollment, sviluppata nella SDK_G, verificata l'autenticità e l'originalità della CIE_G, restituisce il NIS_G e l'$hash_G$ della chiave pubblica dei servizi H(KPUB). L'integratore del servizio potrà così abilitare la CIE_G all'uso del proprio servizio, associando l'output restituito dalla SDK_G all'utente.
La fase di accesso consente all'utente, precedentemente registratosi nella fase di Enrollment, di utilizzare la CIE_G per accedere al servizio pinless. L'accesso avviene, ad esempio, su un tornello che, verificata l'originalità della CIE_G mediante i dati salvati in fase di Enrollment e ne autorizza l'accesso.
CieNis-java-sdk è strutturata per un'applicazione Desktop e richiede l'installazione di Java SE 15 o superiore. La SDK_G richiede la dipendenza esterna della libreria BouncyCastle[7].

```
NisSdk nisSdk = new TestNisAuthenticated().initSdk();
  if(nisSdk.isReady()){
      //metodo per eseguire la registrazione
      nisSdk.enroll();
      //metodo che mostra le operazioni di accesso da eseguire al
          tornello
      nisSdk.access();
}

public NisSdk initSdk() {
  return new NisSdk(new NfcTerminalImpl(), this, true);
}
```

Listing 4.32: Esempio di uso di cie-nis-java-sdk.

Una volta scaricato l'SDK$_G$ si copiano le cartelle core e ias nel progetto (cartelle core e ias presenti nell'SDK$_G$). Successivamente. da una carta di identità collegata a un lettore NFC$_G$ si effettua la lettura del NIS$_G$ e dell'hash$_G$.
Infine si implementano le API$_G$ per l'interazione con il database e il front-end$_G$. In particolare un API$_G$, si occupa di leggere il NIS$_G$ dalla carta, confrontarlo con i dati salvati in database e se presente restituire true, altrimenti restituirà false (Listing 4.8).
Un'altra API$_G$ si occupa di recuperare in NIS$_G$ e l'hash$_G$ code. (Listing 4.11)
Per l'utilizzo di un lettore di carte su sistema operativo Linux, è obbligatoria l'installazione delle seguenti librerie[9]: *libccid*, *libpcslite1*, *libpcslite-dev*, *pcsc-tools*.

Capitolo 5

Valutazione retrospettiva

In questo capitolo sono presenti il soddisfacimento degli obiettivi dello stage e le mie conclusioni

5.1 Consuntivo attività e obiettivi

Nella Tabella 5.1 sono esplicitati i tempi che sono stati consuntivati al termine dello stage.

Tabella 5.1: Tabella dei tempi effettivi

Attività	**Tempo previsto**	**Tempo effettivo**
Analisi requisiti e stesura documentazione	92h	60h
Studio documentazione tecnica	80h	77h
Formazione su procedure aziendali	4h	4h
Formazione linguaggi di sviluppo	20h	24h
Formazione ambienti di sviluppo	4h	24h
Implementazione parte back-end$_G$	36h	85h
Implementazione parte front-end$_G$	68h	30h
Totale ore	304h	304h

Come si può facilmente vedere dalla Tabella 5.1, tra le ore preventivate e le ore consuntivate nelle singole voci, si evidenziano delle differenze.
Le cause che hanno determinato gli scostamenti rispetto ai tempi previsti sono:

- per l'analisi dei requisiti e la stesura della documentazione, il tempo impiegato è stato inferiore di 32 ore. Dovendo integrare due funzioni su un programma già esistente, è stata fatta l'analisi dei requisiti solo nei task del progetto, questo ha portato a una riduzione del tempo speso per l'analisi e la stesura della documentazione;
- lo studio della documentazione tecnica ha richiesto 3 ore in meno rispetto i tempi preventivati;
- la formazione sui linguaggi di sviluppo è stata aumentata di 4 ore, dovute a un webinar$_G$ interno di 3 giorni lavorativi (24 ore) sul linguaggio Angular$_G$;

* la formazione sugli ambienti di sviluppo ha richiesto 20 ore più del preventivato, in quanto sono stati considerati IDE$_G$, come Visual Studio Code, Eclipse, ecc., che non sono mai stati utilizzati durante il percorso di studio;

* la codifica del back-end$_G$ ha richiesto molto tempo in più (49 ore) della stima preventivata. L'incremento del tempo è dovuto principalmente alle innumerevoli prove effettuate sulla lettura di diversi documenti. Trovare la miglior combinazione di filtri e dizionario è stato il processo che ha maggiormente influito sullo scostamento;

* una volta appreso il funzionamento di Angular$_G$, l'implementazione della parte front-end$_G$ ha richiesto 38 ore meno del previsto, in quanto si è trattato di aggiungere alcuni parametri o metodi al codice già in parte esistente. Essendo un nuovo prodotto la parte di front-end$_G$ potrà essere soggetta a ulteriori modifiche.

Nella Tabella 5.2 è esplicitato il soddisfacimento degli obiettivi, descritti nella Tabella 2.1, divisi per importanza e con il numero di quelli soddisfatti e non.

Tabella 5.2: Tabella soddisfacimento obiettivi

ID	Descrizione	Soddisfatto sì/no
	Obiettivi obbligatori	
O01	Analisi documentazione tecnica	**si**
O02	Analisi requisiti applicativi e tecnici e diagramma di flusso	**si**
O03	Mockup del front-end$_G$	**si**
O04	Implementazione delle parti back-end$_G$ e front-end$_G$ per la lettura delle informazioni dalla foto del documento	**si**
O05	Implementazione della parte front-end$_G$ per l'upload dell'immagine del documento e la visualizzazione delle informazioni in esso contenute	**si**
	Obiettivi desiderabili	
D01	Analisi documentazione tecnica Middleware CIE$_G$	**si**
D02	Analisi requisiti applicativi e tecnici e diagramma di flusso	**si**
D03	Implementazione front-end$_G$ per l'interfacciamento con il lettore NFC$_G$	**no**
D04	Implementazione front-end$_G$ e back-end$_G$ per il login tramite CIE$_G$	**si**

L'obiettivo facoltativo D03, presente nella Tabella 5.2, non è stato soddisfatto, non essendo stato possibile ottenere in tempi utili l'accesso al servizio fornito dallo Stato *"Entra con CIE"*. D'accordo con il tutor aziendale, sono stati utilizzati degli altri metodi, sempre forniti dallo stato, per estrarre dati dalla CIE$_G$.
In particolare, il codice NIS$_G$, è stato associato all'utente proprietario del documento, riuscendo cosi a fare ugualmente un "login con CIE" come previsto dall'obiettivo D04 presente nella Tabella 5.2. Sanmarco ha ritenuto comunque sufficiente anche questo metodo di login. Sono stati soddisfatti tutti gli obiettivi obbligatori.

5.2 Conoscenze acquisite

Di seguito vengono descritte le conoscenze acquisite grazie a questo percorso di stage.

Lavoro individuale Durante questo percorso di stage, ho dovuto studiare nuove tecnologie in modo totalmente autonomo. Il mio tutor e l'intero team, mi hanno fornito tutto il materiale, il supporto e l'aiuto di cui necessitavo per poter raggiungere gli obiettivi prefissati. Mi hanno lasciato libero su come fare il PoC$_G$, e questo mi ha permesso di apprendere e riflettere a fondo, sulle tecnologie da me utilizzate. Ogni giorno ho superato i miei limiti riuscendo sempre ad imparare nozioni che fino a prima di allora erano a me sconosciute.

Lavoro in team Nonostante abbia impiegato la maggior parte del mio tempo nel lavoro individuale, non è mancata l'occasione di lavorare in team. Questo mi ha permesso di capire che per svolgere un buon lavoro di gruppo, è necessario definire delle regole ben precise in modo da poter raggiungere obiettivi comuni a tutti. Lavorando in team, ho compreso che l'aiutarsi nelle difficoltà e una buona coesione del gruppo, sono il segreto per una buona riuscita del progetto.

Tecnologie Durante il percorso di stage ho avuto la possibilità di imparare il framework$_G$ Angular$_G$. Inoltre, ho affinato le mie conoscenze nell'uso del linguaggio Java e in particolare, sull'uso di Spring, nella gestione dei database e nella gestione delle chiamate rest effettuate con Postman.
Inoltre, ho visto la gestione dei task e delle repository tramite programmi della suite Atlassian. Lo stage mi ha portato a conoscere cose nuove e approfondire argomenti che avevo visto durate gli studi. Quindi non posso che essere soddisfatto di queste nuove conoscenze acquisite.

5.3 Università a confronto con il mondo lavorativo

Durante lo stage mi sono ritrovato ad utilizzare nozioni e tecnologie che ho studiato durante il percorso di laurea, sopratutto per tematiche affrontate nei corsi di *Ingegneria del Software*, di *Programmazioni ad oggetti* e di *Basi di dati*.
Tuttavia penso che la maggior parte delle conoscenze acquisite durante questi due mesi siano state del tutto nuove o non affrontate approfonditamente durante gli anni accademici. Questo probabilmente è dovuto al fatto che un corso di laurea deve garantire conoscenze di base, che poi lo studente affinerà nel mondo del lavoro. Nonostante ciò, reputo che il nostro corso di laurea fornisca le conoscenze fondamentali nel formare persone che poi andranno a ricoprire diverse posizioni nel mondo del lavoro in ambito IT. Questo perché i vari corsi garantiscono una buona infarinatura generale dei concetti che poi verranno utilizzati nelle attività lavorative. Grazie ai numerosi progetti viene permesso agli studenti di farsi un'idea del lavoro in team che è richiesto nella maggior parte delle aziende.
Se dovessi trovare un aspetto che secondo me può essere migliorato nel corso di laurea, è l'esperienza concreta che si può fare lavorando sul campo. Lo stage è essenziale per questo, tuttavia se fosse possibile spendere ulteriore tempo per interfacciarsi con il mondo del lavoro, certamente ogni studente che porterà a termine il percorso di studi, avrà una visione più chiara di ciò che potrà fare.

5.4 Valutazione personale

Mi sento molto soddisfatto del percorso di stage che ho intrapreso e concluso.
Ho avuto la possibilità di lavorare per una azienda che rispecchia le mie aspettative e sono stato inserito all'interno di un team di alto livello con cui siamo entrati subito in sintonia. Sono inoltre soddisfatto di me stesso, in quanto concluso il periodo di stage l'azienda ha deciso di assumermi. Sanmarco mi ha permesso di conoscere il mondo della programmazione in maniera appassionante e appassionata, facendomi cosi scegliere questa via come ambito in cui voglio concentrarmi nell'immediato futuro.
Dal mio punto di vista, uno stage è essenziale per qualsiasi corso di laurea, perché ti permette di confrontare i concetti teorici del corso di studio, alla praticità del mondo del lavoro. Infine consiglio a tutte le persone a intraprende un percorso di studi nell'informatica, per poter entrare in questo mondo che è sempre in espansione dove non ci sono limiti alle cose che si possono apprendere, scoprire, fare, creare.

Glossario

Angular framework sviluppato da Google per lo sviluppo delle applicazioni web. 4, 20, 22, 23, 27, 49, 58

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. 25, 56, 63

back-end parte del programma non visibile all'utente che permette il funzionamento delle interazioni con il front-end. 3, 6–8, 20, 21, 27, 50, 52, 57, 58

CIE carta d'identità elettronica emessa dallo stato italiano, è caratterizzata da un codice MRZ. v, 5, 6, 8, 19, 43, 48, 55, 58, 63

CRUD in infromatica CRUD, acronimo di create read update delete, sono le quattro operazini di un databaase persistente. Nel contesto dello sviluppo di applicazioni web corrispondono alle quattro operazioni HTTP: put, get, post e delete. 24

DTO è un design pattern usato per trasferire dati tra sottosistemi di un'applicazione software. 28

framework architettura logica di supporto sulla quale un software può essere progettato e sviluppato, facilitandone lo sviluppo da parte del programmatore. 4, 9, 20, 22–24, 27, 49, 58

front-end parte del programma visibile all'utente con cui egli può interagire, tipicamente è un'interfaccia utente. 3, 6–8, 20, 21, 49, 56–58

hash è la chiave pubblica dei servizi, chiave che è associata a ciascuna CIE. 31, 32, 55, 56

ICAO è un'agenzia autonoma delle Nazioni Unite incaricata di sviluppare i principi e le tecniche della navigazione aerea internazionale, delle rotte e degli aeroporti e promuovere la progettazione e lo sviluppo del trasporto aereo internazionale rendendolo più sicuro e ordinato. 34, 46, 63

IDE è un software che, in fase di programmazione, supporta i programmatori nello sviluppo e debugging del codice sorgente di un programma. 3, 21, 22, 58, 63

MRZ codice alfanumerico, usato per l'identificazione dei dati del proprietario di una carta di identità elettronica o un passaporto. Gestito dall'ICAO. v, 7, 34–36, 40, 43, 48, 53, 63

NFC è una tecnologia di ricetrasmissione che fornisce connettività senza fili bidirezionale a corto raggio. 6, 8, 56, 63

NIS è il numero identificativo dei servizi, un codice che è associato a ciascuna CIE. 10, 11, 19, 31–33, 55, 56, 58, 63

OCR sono programmi dedicati al rilevamento dei caratteri contenuti in un documento e al loro trasferimento in testo digitale leggibile da una macchina. 25, 26, 34, 35, 63

PoC è una bozza di un determinato progetto o metodo, allo scopo di provarne la fattibilità o dimostrare la fondatezza di alcuni principi o concetti costituenti. v, 7, 58, 63

Scrum è un framework agile utilizzato per la gestione del ciclo di vita del software, iterativo ed incrementale, con lo scopo di gestire progetti, prodotti software o applicazioni di sviluppo. 2, 7

SDK in informatica, indica genericamente un insieme di strumenti per lo sviluppo e la documentazione di software. 55, 56

SQL è un linguaggio standardizzato per database basati sul modello relazionale (RDBMS), progettato per diverse operazioni come l'inserimento, la modifica, l'eliminazione, ecc, di dati in un database. 23, 63

stakeholder in ingegneria del software, uno stakeholder è una persona a vario titolo coinvolta nel ciclo di vita di un software, che ha influenza sul prodotto o sul processo. 2, 3

UML in ingegneria del software, *UML, Unified Modeling Language* (ing. linguaggio di modellazione unificato) è un linguaggio di modellazione e specifica basato sul paradigma object-oriented. L'*UML* svolge un'importantissima funzione di "lingua franca" nella comunità della progettazione e programmazione a oggetti. Gran parte della letteratura di settore usa tale linguaggio per descrivere soluzioni analitiche e progettuali in modo sintetico e comprensibile a un vasto pubblico. 9, 63

webinar è una sessione educativa o informativa la cui partecipazione avviene in forma remota tramite una connessione a internet. 4, 58

Acronimi

API Application Program Interface. 61

CIE Carta d'Identità Elettronica. 61

DTO Data Transfer Object. 61

ICAO International Civil Aviation Organization. 61

IDE Integrated Development Environment. 61

MRZ Machine Readable Zone. 62

NFC Near Field Communication. 62

NIS Numero Identificativo dei Servizi. 62

OCR Optical Character Recognition. 62

PoC Proof of Concept. 62

SDK Software Development Kit. 62

SQL Structured Query Language. 62

UML Unified Modeling Language. 62

Bibliografia

Siti web consultati

[1] *PDF ICAO 9303*. URL: https://www.icao.int/publications/Documents/9303_p3_cons_en.pdf (cit. alle pp. 33, 44).

[2] *Repository della libreria wrapper di OpenCV*. URL: https://mvnrepository.com/artifact/org.openpnp/opencv (cit. a p. 36).

[3] *Repository di cie-nis-java-sdk*. URL: https://github.com/italia/cie-nis-java-sdk (cit. a p. 54).

[4] *Repository di Tesseract-OCR*. URL: https://github.com/tesseract-ocr/tesseract (cit. alle pp. 25, 34).

[5] *Repository dove è presente il dizionario mrz*. URL: https://github.com/DoubangoTelecom/tesseractMRZ (cit. alle pp. 34, 38, 39).

[6] *Repository dove è presente il dizionario ocrb*. URL: https://github.com/Shreeshrii/tessdata_ocrb (cit. alle pp. 34, 38, 39).

[7] *Sito da cui è stata presa la libreria utile alla lettura del codice NIS del documento*. URL: https://mvnrepository.com/artifact/org.bouncycastle/bcprov-jdk15on/1.64 (cit. a p. 54).

[8] *Sito da cui scaricare la libreria Tess4j*. URL: http://mvnrepository.com/artifact/net.sourceforge.tess4j (cit. a p. 25).

[9] *Sito da cui scaricare tutte le librerie per linux,per lettura di un documento tramite un lettore di carte*. URL: https://people.debian.org/~rousseau/smartcard.html (cit. a p. 55).

[10] *Sito da cui si può scaricare una cartella tessdata utile alla lettura di un documento di identità*. URL: https://sourceforge.net/projects/tess4j/.

[11] *Sito da cui sono state prese le immagini e parte del codice delle conversioni*. URL: https://www.tutorialspoint.com/index.htm (cit. a p. 36).

[12] *Sito ufficiale Angular*. URL: https://angular.io/ (cit. alle pp. 3, 21).

[13] *Sito ufficiale Atlassian*. URL: https://www.atlassian.com/it (cit. a p. 3).

[14] *Sito ufficiale CIE developers*. URL: https://developers.italia.it/it/cie/ (cit. a p. 54).

[15] *Sito ufficiale DBeaver*. URL: https://dbeaver.io/ (cit. a p. 22).

[16] *Sito ufficiale di StarUML, usato per disegnare i casi d'uso.* URL: https://staruml.io/ (cit. a p. 9).

[17] *Sito ufficiale Eclipse.* URL: https://www.eclipse.org/ide/ (cit. alle pp. 3, 21).

[18] *Sito ufficiale OpenCV.* URL: https://opencv.org/ (cit. alle pp. 25, 34, 36).

[19] *Sito ufficiale Tess4j.* URL: https://tess4j.sourceforge.net/ (cit. a p. 25).

[20] *Sito ufficiale Visual Studio Code.* URL: https://code.visualstudio.com/ (cit. a p. 3).

[21] *Sito ufficilae di Spring.* URL: https://spring.io/ (cit. a p. 23).

[22] *Wikipedia.* URL: https://it.wikipedia.org/ (cit. alle pp. 2, 3, 21, 23, 25–27).