Lavoratori Stagionali

Ingegneria del software

Segala Federico & Zamboni Davide

VR457144 - VR455880

Università degli Studi di Verona

Settembre 2022

Indice

Contesto **3**

Analisi dei requisiti **4**

Casi d'uso **4**

Inserimento dati lavoratore 5

Visualizza e Modifica dati 7

Diagrammi di Attività **9**

Implementazione e sviluppo **11**

Note sullo sviluppo **11**

Diagramma delle classi **12**

Classi implementate 13

Diagrammi di Sequenza del Software **14**

Inserimento di un lavoratore 14

Visualizzazione e Modifica di un Lavoratore 15

Pattern Architetturali usati **16**

MVC 16

Singleton Pattern 17

Observer Pattern 17

Test e Validazione **17**

Ispezione del codice e confronto con documentazione 17

Test degli sviluppatori 18

Test utente generico 20

Scelte Progettuali **20**

Contesto

Si vuole progettare un sistema informatico di una agenzia che fornisce servizi di supporto alla ricerca di lavoro stagionale. I lavoratori interessati possono iscriversi al servizio, rivolgendosi agli sportelli dell'agenzia. Il sistema deve permettere la gestione delle anagrafiche e la ricerca di lavoratori stagionali, nei settori dell'agricoltura e del turismo.

I responsabili del servizio, dipendenti dell'agenzia, inseriscono i dati dei lavoratori. Per ogni lavoratore vengono memorizzati i dati anagrafici (nome, cognome, luogo e data di nascita, nazionalità), indirizzo, recapito telefonico personale (se presente), email, le eventuali specializzazioni/esperienze precedenti (bagnino, barman, istruttore di nuoto, viticultore, floricultore), lingue parlate, il tipo di patente di guida e se automunito. Sono inoltre memorizzati i periodi e le zone (comuni), per i quali il lavoratore è disponibile. Di ogni lavoratore si memorizzano anche le informazioni di almeno una persona da avvisare in caso di urgenza: nome, cognome, telefono, indirizzo email.

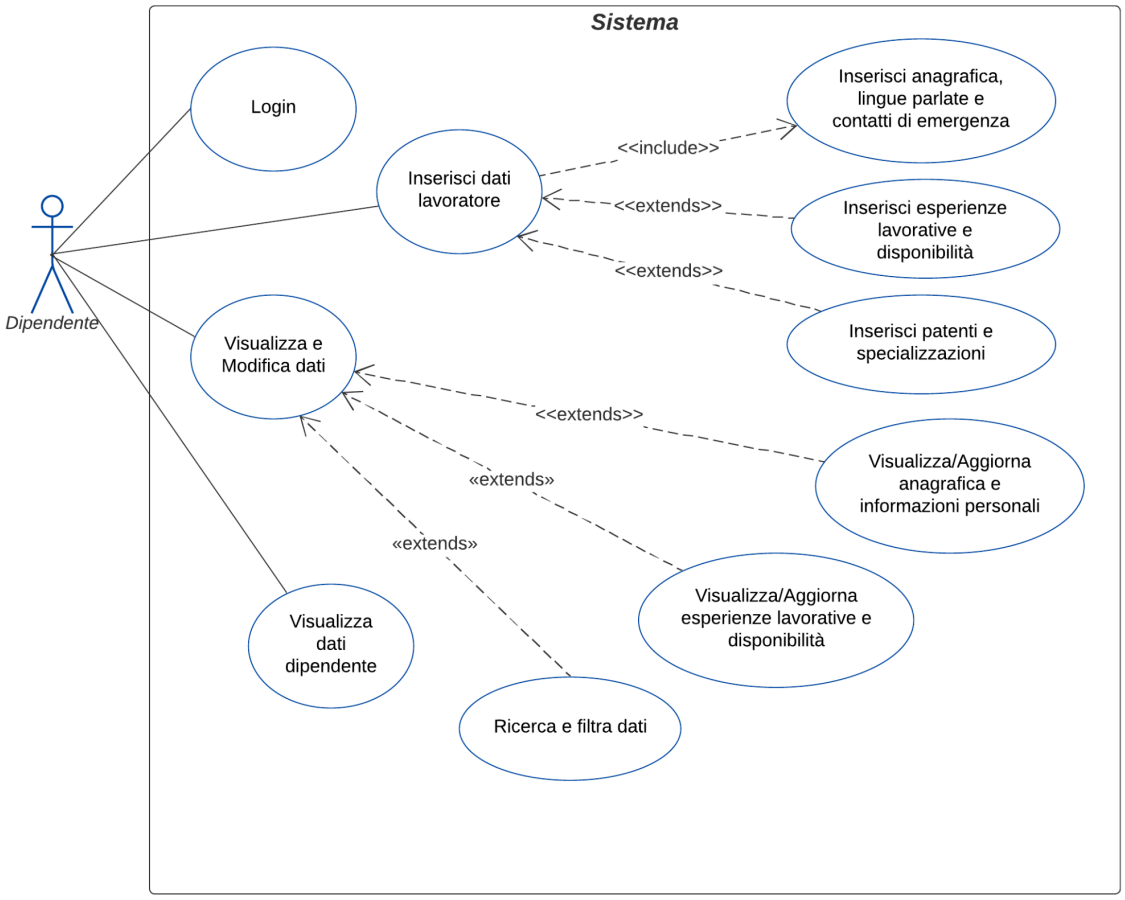
I dipendenti dell'agenzia devono autenticarsi per poter accedere al sistema e inserire i dati dei lavoratori. Il sistema permette ai dipendenti dell'agenzia di aggiornare le anagrafiche con tutti i lavori che i lavoratori stagionali hanno svolto negli ultimi 5 anni. Per ogni lavoro svolto vanno registrati: periodo, nome dell'azienda, mansioni svolte, luogo di lavoro, retribuzione lorda giornaliera. Per i dipendenti dell'agenzia si memorizzano i dati anagrafici, l'indirizzo email, il telefono e le credenziali di accesso (login e password).

Una volta registrate le informazioni sui lavoratori, il personale dell'agenzia può effettuare ricerche rispetto a possibili profili richiesti.In particolare, il sistema deve permettere ai dipendenti di effettuare ricerche per lavoratore, per lingue parlate, periodo di disponibilità, mansioni indicate, luogo di residenza, disponibilità di auto/patente di guida. Il sistema deve permettere di effettuare ricerche complesse, attraverso la specifica di differenti condizioni di ricerca (sia in AND che in OR).

Analisi dei requisiti

Casi d'uso

Gli attori principali del sistema sono i responsabili del servizio, i quali devono inserire i dati dei lavoratori. I dipendenti dell'agenzia per utilizzare il sistema devono prima accedere con le proprie credenziali.



Dopo l'opportuna autenticazione il dipendente accede all'interfaccia che permette la visualizzazione dei dati e, qualora siano presenti, la loro relativa modifica. È possibile inoltre accedere alla schermata di inserimento di un nuovo lavoratore e alla visualizzazione dati del dipendente oppure uscire dal programma. L'utente, una volta fatto l'accesso, ha a disposizione i diversi casi d'uso in qualsiasi momento.

Inserimento dati lavoratore

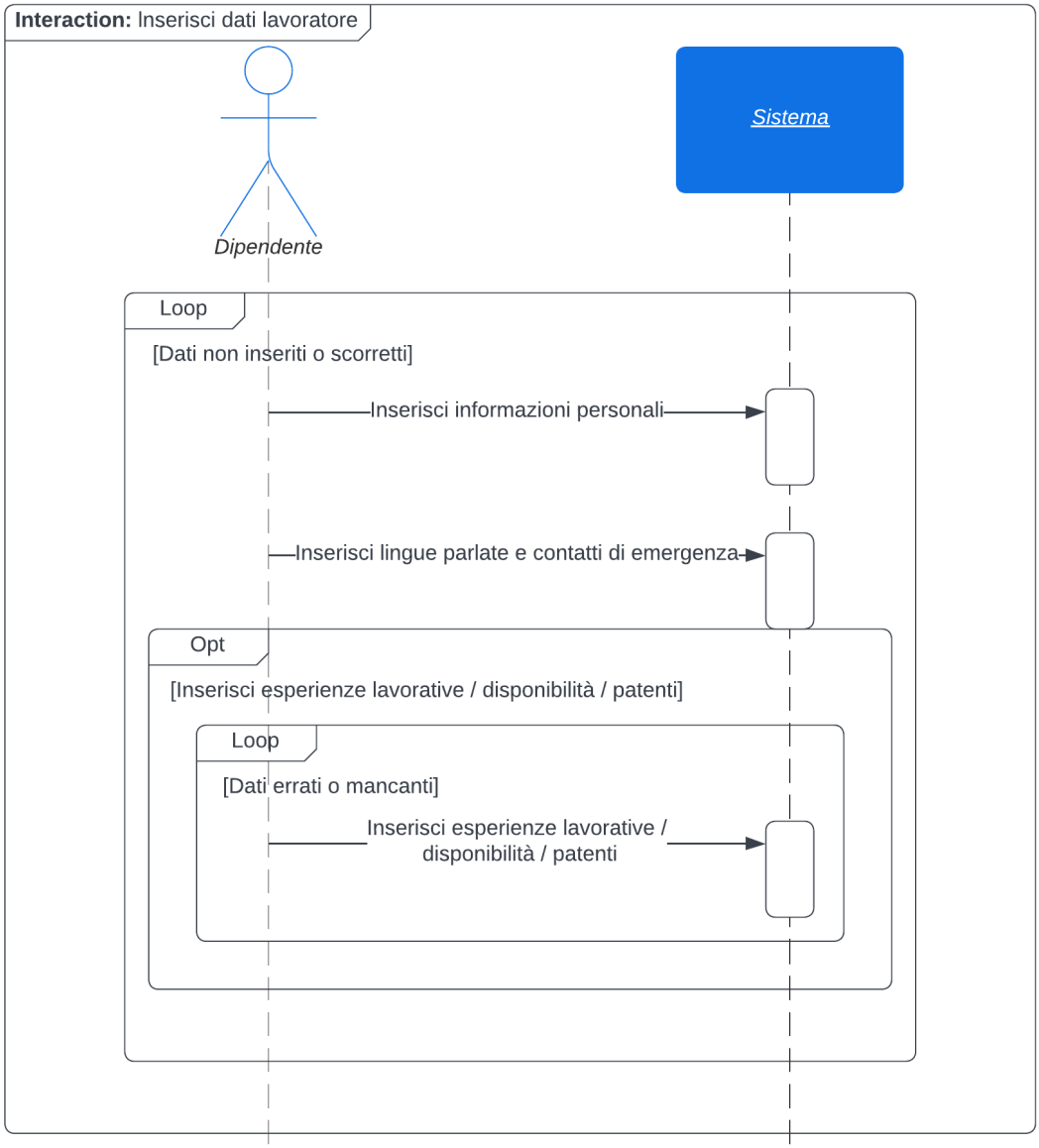
Il dipendente deve essere in grado di inserire i dati di un lavoratore. L'inserimento dei dati di un lavoratore comporta il contestuale inserimento dei dati anagrafici (nome, cognome, luogo e data di nascita, nazionalità), indirizzo, recapito telefonico personale, email, se automunito, ed eventuali specializzazioni/esperienze precedenti. In questa schermata è inoltre possibile inserire le esperienze lavorative, i periodi di disponibilità, le lingue parlate, patenti ed i contatti di emergenza.

A livello progettuale è stato scelto di rendere facoltativo l'inserimento delle esperienze lavorative, periodi di disponibilità, patenti e specializzazioni in quanto è possibile integrare questi dati anche in fasi successive come nella visualizzazione dei dati.

Attori	Dipendente
Precondizioni	Il dipendente deve già essersi autenticato
Passi	1. Il dipendente accede all'interfaccia base del sistema; 2. Il dipendente accede al servizio di inserimento dati del lavoratore; 3. Il dipendente inserisce le informazioni personali del lavoratore; 4. Il dipendente inserisce le lingue parlate e i contatti di emergenza del lavoratore stagionale; 5. Il dipendente può inserire le esperienze lavorative / disponibilità / patenti del lavoratore: a. I dati vengono inseriti in modo errato, torno al punto 5; b. Le esperienze / disponibilità / patenti sono corrette, vado avanti; 6. Premo sul pulsante salva: a. Le informazioni sono errate o mancanti. Torno al punto 3; b. Le informazioni sono corrette. Concludo.
Postcondizioni	I dati del lavoratore sono inseriti

Quando il dipendente si trova nella schermata di inserimento delle informazioni ha la possibilità di inserire le informazioni personali del lavoratore.

Ai fini di avere più chiarezza nel sequence diagram viene considerato accorpato l'inserimento delle lingue parlate e dei contatti di emergenza, in quanto il loro inserimento avviene nel medesimo modo. Anche le esperienze lavorative, i periodi di disponibilità e le patenti sono state accorpate per lo stesso motivo appena citato.



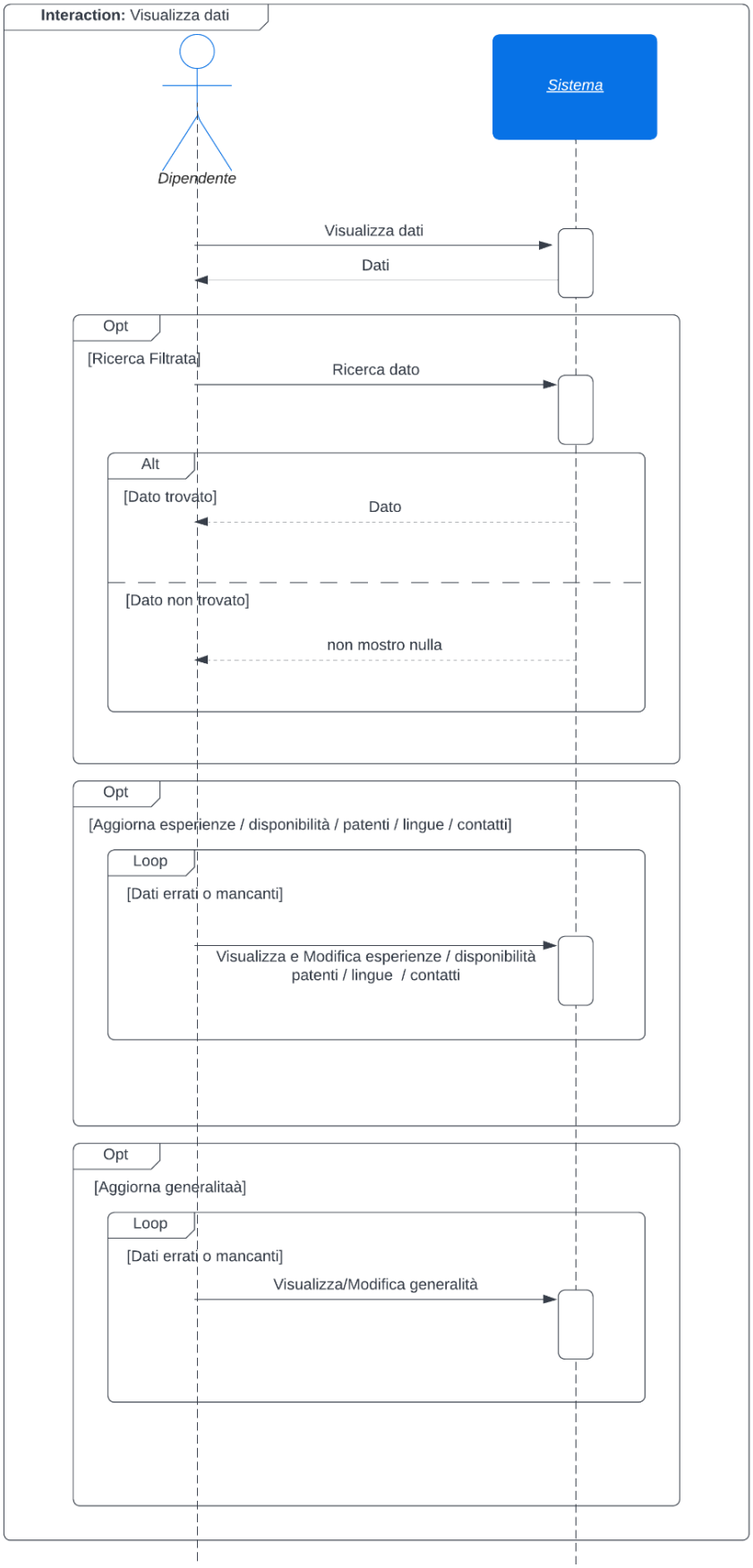
Nel caso in cui le singole informazioni siano errate o mancanti l'utente si ritroverà a reinserirle correttamente con contestuale avviso.

Durante l'inserimento delle informazioni personali, il dipendente può al contempo passare all'inserimento delle patenti, se presenti, dei periodi e luoghi di disponibilità e delle esperienze lavorative che i lavoratori stagionali hanno svolto negli ultimi 5 anni. Per ogni lavoro svolto vengono inseriti: periodo, nome dell'azienda, mansioni svolte, luogo di lavoro e retribuzione lorda giornaliera. Nel caso in cui le singole informazioni siano errate l'utente si ritroverà a reinserirle correttamente con contestuale avviso.

Visualizza e Modifica dati

All'interno del sistema è possibile visualizzare i dati precedentemente inseriti ed integrarli con informazioni aggiuntive o modificarli. Nella medesima schermata è possibile inoltre filtrare i dati attraverso una ricerca (in and oppure in or). Nel caso in cui la ricerca non produca risultati, verrà riportata la tabella dei lavoratori vuota.

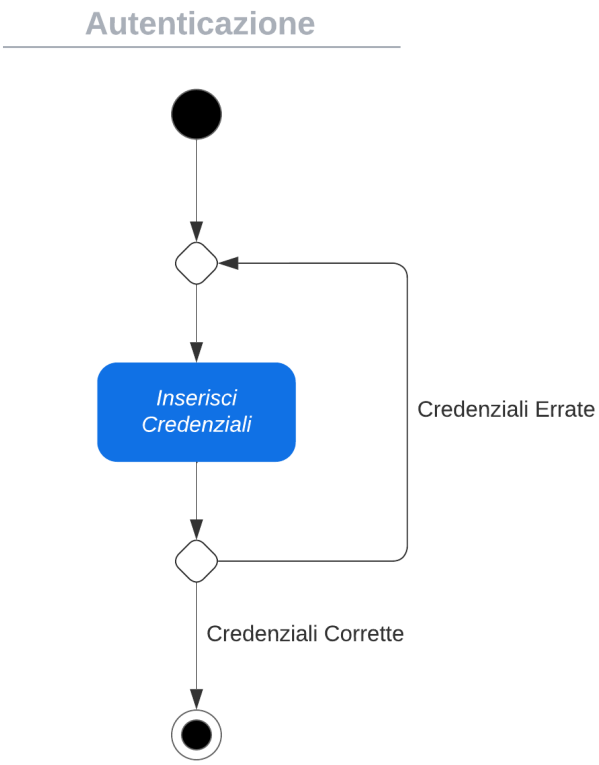
Attori	Dipendente
Precondizioni	L'utente deve già essere autenticato
Passi	1. Il dipendente accede all'interfaccia base del sistema; 2. All'interno dell'interfaccia di base l'utente avrà davanti a sé tutti i record relativi ai lavoratori stagionali; 3. L'utente avrà la possibilità di ottenere un elenco ristretto dei record tramite una ricerca filtrata; 4. Trovato il record di interesse sarà possibile, tramite un click, accedere alle schermate di visualizzazione e/o modifica delle generalità, lingue parlate, patenti, contatto di emergenza, disponibilità e delle esperienze lavorative; 5. L'utente accedendo ad una schermata di visualizzazione/ modifica avrà la possibilità di visualizzare e/o modificare l'oggetto in questione, attraverso la replica della vista che ha utilizzato in fase di inserimento. Per ogni modifica effettuata viene effettuato un controllo: a. Le informazioni sono errate o mancanti. Torno al punto 5. b. Le informazioni sono corrette. Torno al punto 2.
Postcondizioni	Al termine di questo caso d'uso l'utente avrà a propria disposizione i dati del lavoratore stagionale richiesto e eventualmente i dati modificati

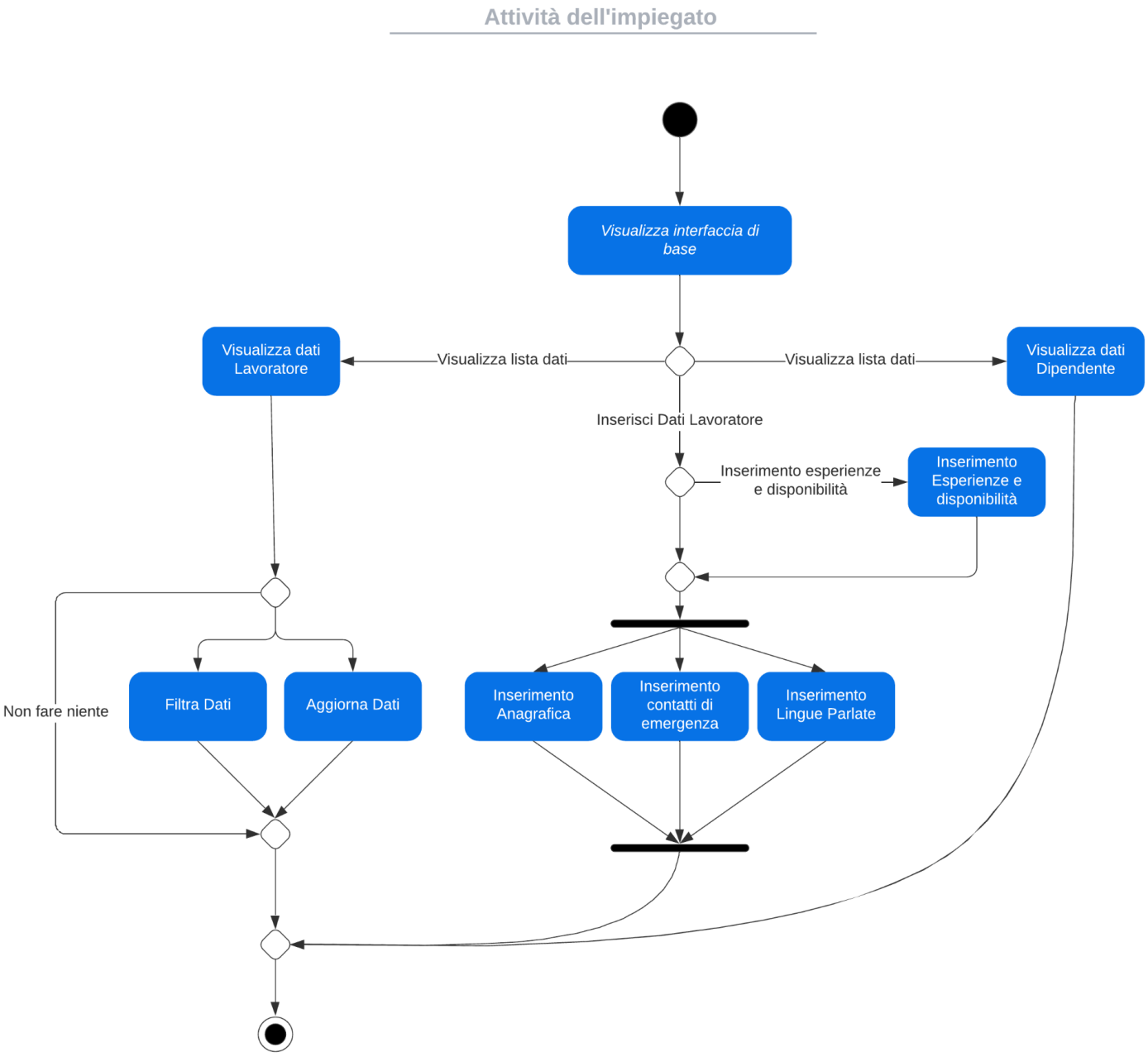

Interaction: Visualizza dati
Sistema
Dipendente
Visualizza dati
Dati
Opt
[Ricerca Filtrata]
Ricerca dato
Alt
[Dato trovato]
Dato
[Dato non trovato]
non mostro nulla
Opt
[Aggiorna esperienze / disponibilità / patenti / lingue / contatti]
Loop
[Dati errati o mancanti]
Visualizza e Modifica esperienze / disponibilità
patenti / lingue / contatti
Opt
[Aggiorna generalitaà]
Loop
[Dati errati o mancanti]
Visualizza/Modifica generalità

Diagrammi di Attività

Di seguito vengono illustrati i diagrammi di attività dell'operazione di autenticazione del dipendente e di tutte le possibili attività dell'impiegato una volta effettuato l'accesso.

Nota: i diagrammi che seguono andranno a descrivere una singola attività dell'utente rispetto al sistema. Non è presente all'interno dei diagrammi la possibilità di ripetere più volte la stessa operazione. Questo è stato scelto per semplicità grafica.




Attività dell'impiegato
Visualizza interfaccia di base
Visualizza dati Lavoratore
Visualizza lista dati
Visualizza lista dati
Visualizza dati Dipendente
Inserisci Dati Lavoratore
Inserimento esperienze e disponibilità
Inserimento Esperienze e disponibilità
Non fare niente
Filtra Dati
Aggiorna Dati
Inserimento Anagrafica
Inserimento contatti di emergenza
Inserimento Lingue Parlate

Implementazione e sviluppo

Note sullo sviluppo

Il processo di sviluppo adottato è stato principalmente di tipo agile ed incrementale. Tuttavia, soprattutto durante le fasi iniziali di sviluppo è stata dedicata maggior attenzione alla fase di pianificazione, analisi dei requisiti e progettazione software ad alto livello, avvicinandosi ad un modello Plan Driven.

Successivamente quando si è passati alla fase di realizzazione del progetto tramite linguaggio di programmazione *Java*, si è alternato ciclicamente tra progettazione (talvolta modificando ed ampliando la progettazione iniziale), implementazione e validazione.

Tale sviluppo di tipo agile è stato affiancato ad una pianificazione incrementale, ovvero ogni nuova funzionalità implementata veniva prima progettata, costituendo una base sulla quale veniva poi fatta l'implementazione nel codice e infine si è proceduto con una fase di test.

Assieme allo sviluppo è avvenuta la stesura della relativa documentazione. Durante la fase iniziale di analisi dei requisiti, prima dell'implementazione, è stato fatto uno studio riguardo la progettazione architetturale e sono stati generati i relativi use cases e diagrammi di attività, in modo da avere già specificata un'impalcatura solida sulla quale andare a stendere il codice. In seguito, durante l'alternanza ciclica delle varie fasi, la documentazione è stata modificata ed integrata, principalmente generando i class diagram e sequence diagram.

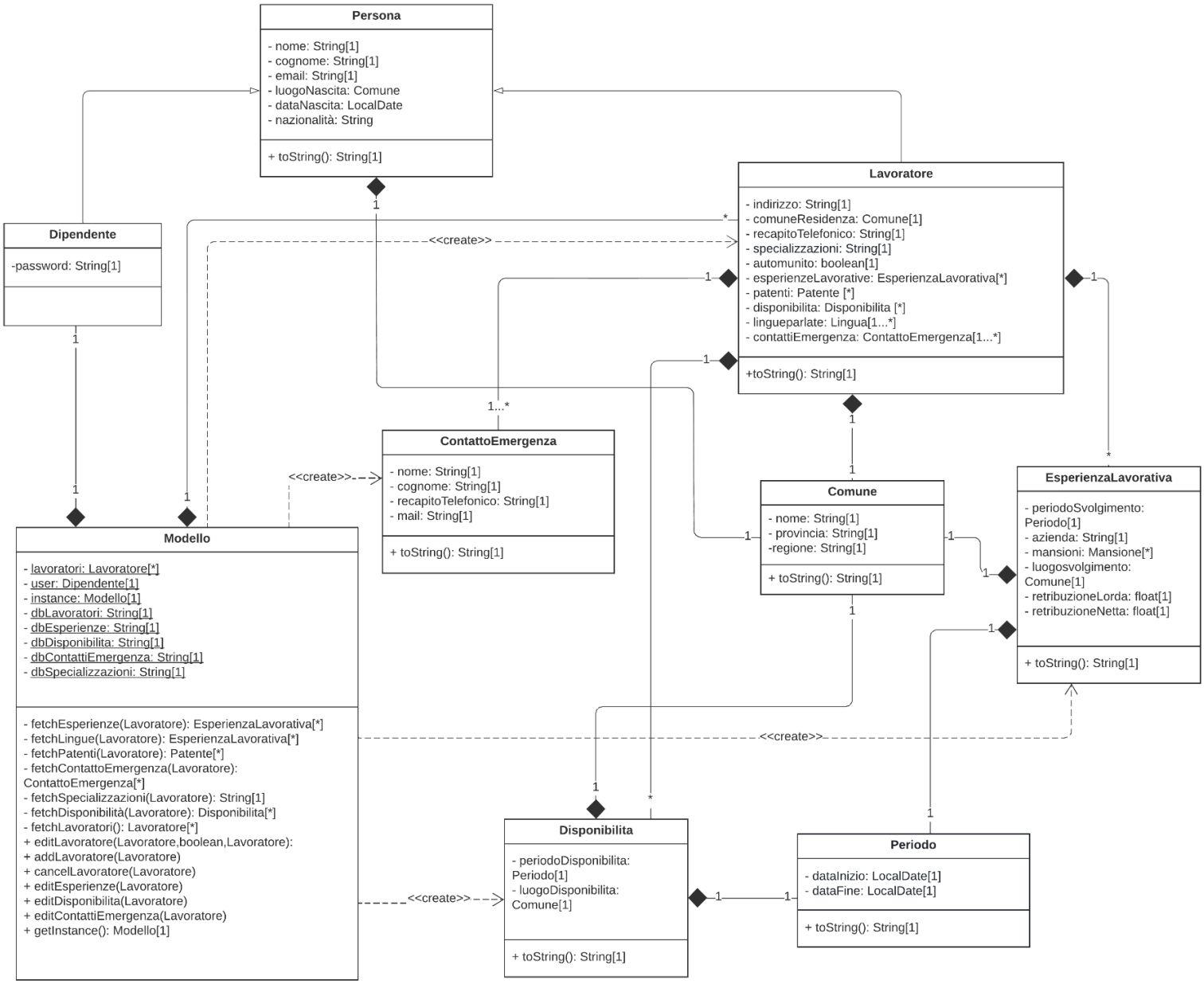
Per quanto riguarda l'implementazione non c'è stata una suddivisione a priori dei task ma si è seguito l'ordine di sviluppo in base alle priorità di funzionamento, aggiungendo gradualmente funzioni in base alla necessità. Tutto il progetto è stato sviluppato utilizzando il sistema di versioning *GitHub* in modo da permettere a tutti i componenti del gruppo di lavorare in maniera asincrona al progetto e avere un sistema che permettesse di avere una tracciabilità sulle varie fasi dello sviluppo. A tali fasi di sviluppo separate sono state affiancate diverse sessioni di Pair-Programming in modo da avere una visione chiara ed univoca soprattutto nelle sezioni più critiche del codice.

Per tracciare lo stato dell'implementazione è stato utilizzato inoltre un sistema di notazione condivisa in modo tale che chiunque potesse aggiungere parti da dover implementare in futuro o casi particolari da coprire.

Diagramma delle classi

Le azioni dell'utente vengono catturate dai *listener*, i quali hanno dei metodi definiti in modo tale che il sistema reagisca, andando a modificare le informazioni contenute nel modello e quindi aggiornare la vista. Di seguito è riportato il diagramma UML delle classi del modello per comprendere la struttura principale del progetto.

Diagramma delle classi del modello



Classi implementate

Per gestire la comunicazione tra Modello, Vista e controllore (vedasi *modello MVC* nei capitoli seguenti) sono state adottate le seguenti scelte:

Per la comunicazione Vista-Controllore si è utilizzato il meccanismo basato su *ActionListeners* la quale implementazione è standard.

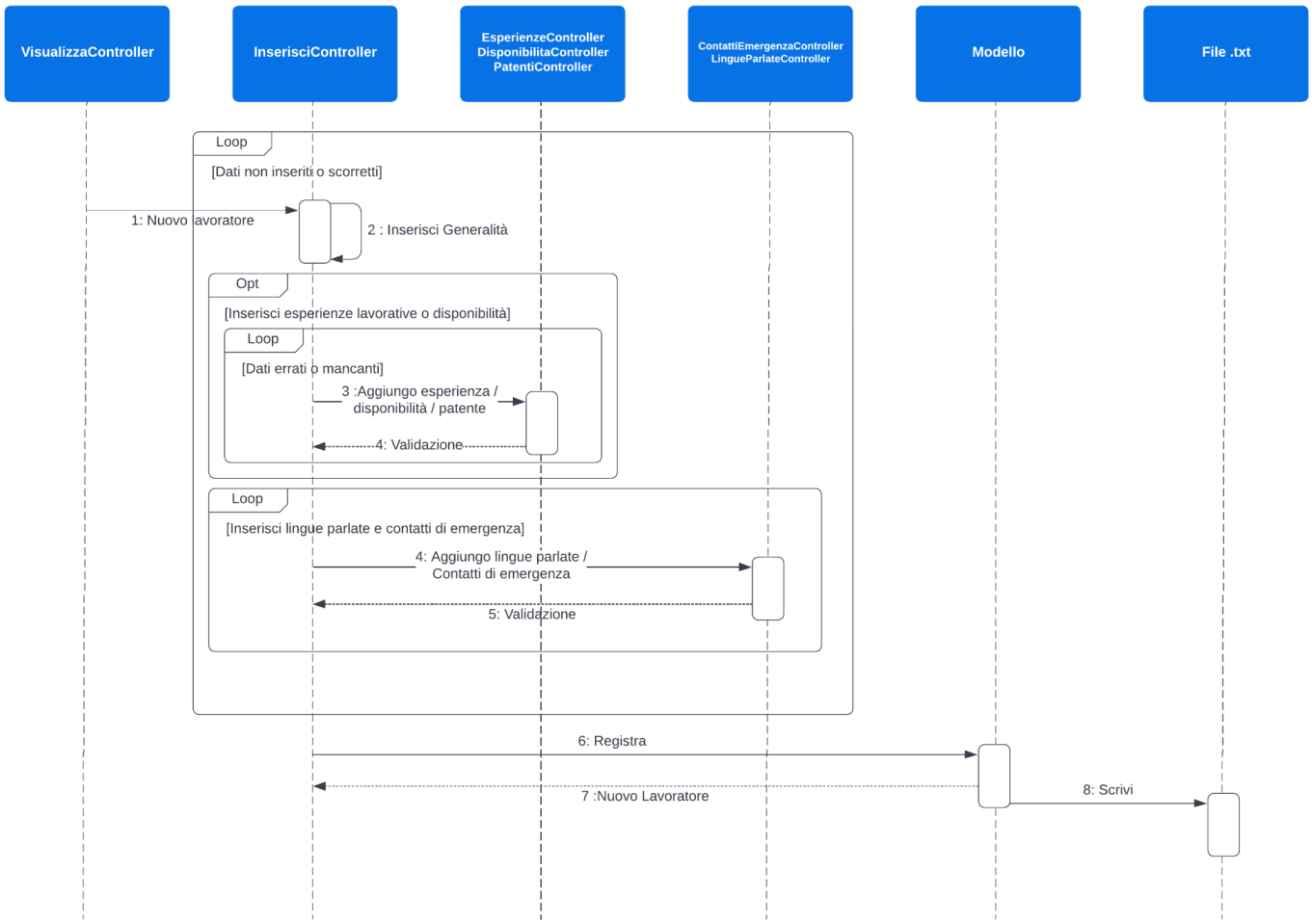
Per la comunicazione Modello-Vista e Modello-Controllore si è utilizzata come punto di accesso ai dati la classe *Model* che immagazzina e fornisce i dati. Di seguito i metodi utilizzati dalla classe *Model* per gestire l'accesso alle collezioni di dati.

- *getInstance( )*: ritorna l'oggetto univoco *Model;*
- *fetchLavoratori( )*: ritorna la lista di lavoratori;
- *fetchEsperienze(Lavoratore);*
- *fetchLingue(Lavoratore);*
- *fetchPatenti(Lavoratore);*
- *fetchContattoEmergenza(Lavoratore);*
- *fetchSpecializzazioni(Lavoratore);*
- *fetchDisponibilità(Lavoratore);*
- *addLavoratore(Lavoratore):* scrive su disco un nuovo lavoratore;
- *cancelLavoratore(Lavoratore):* cancella da disco un lavoratore ;
- *editLavoratore(Lavoratore,boolean,Lavoratore):* sovrascrive su disco un lavoratore;
- *editEsperienze(Lavoratore);*
- *editDisponibilita(Lavoratore);*
- *editContattiEmergenza(Lavoratore);*

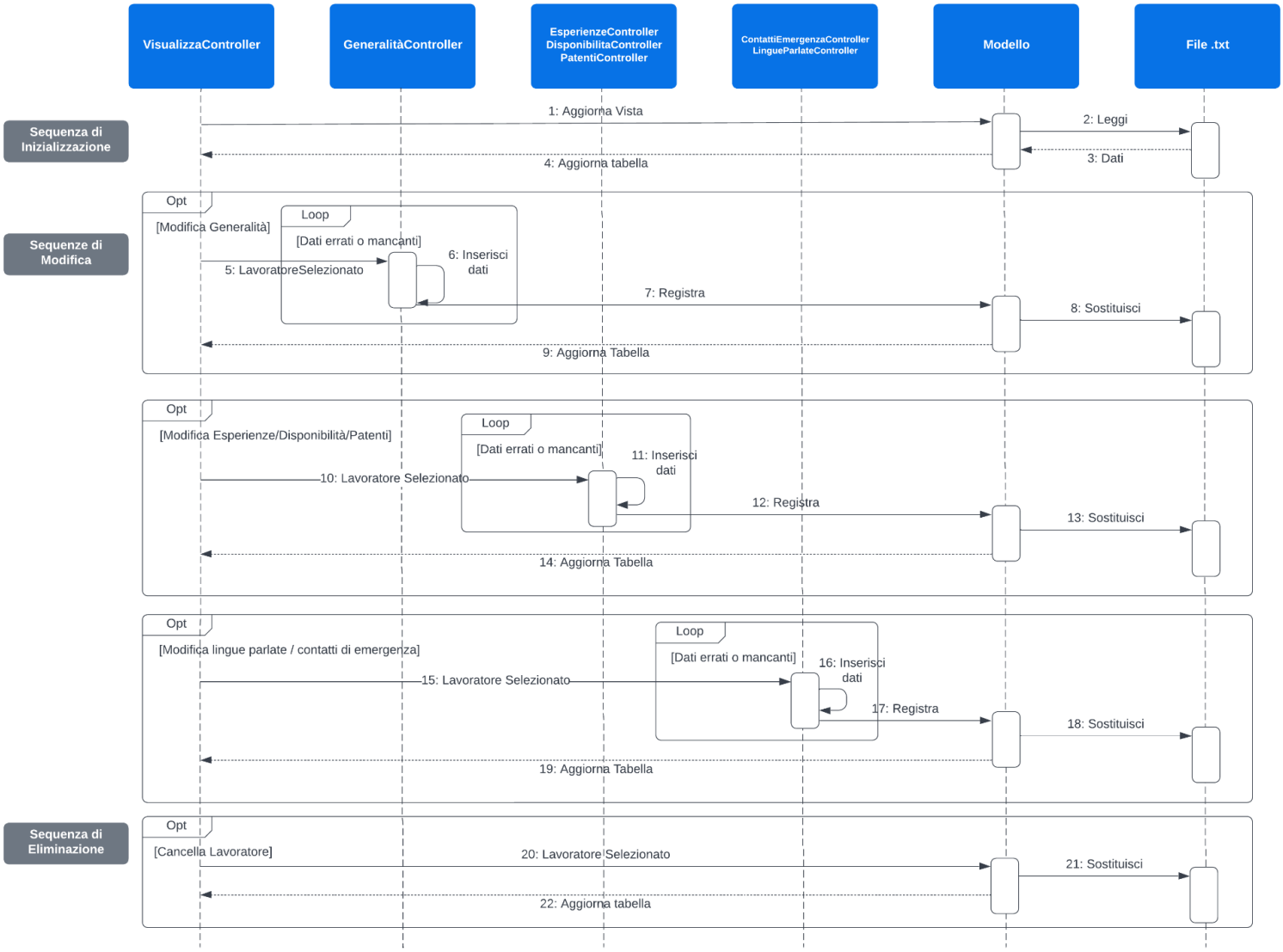
Diagrammi di Sequenza del Software

In questa sezione verranno mostrati i diagrammi di sequenza che descrivono le dinamiche di alcune interazioni tra classi particolarmente complesse. In particolare viene descritta la comunicazione tra Controllore, Modello e file su disco, sia nella fase di inserimento di un nuovo lavoratore che durante la visualizzazione e modifica di un lavoratore. Ai fini di garantire una migliore leggibilità sia le classi EsperienzeController, DisponibilitaController, PatentiController che le classi ContattiEmergenzaController e LinguaParlateController sono state accorpate.

Inserimento di un lavoratore



Visualizzazione e Modifica di un Lavoratore



Pattern Architetturali usati

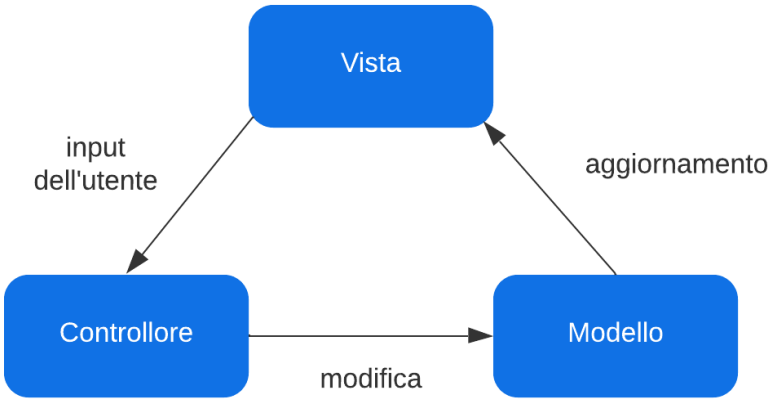
MVC

Il sistema è stato progettato utilizzando le tecniche di modellazione ad oggetti. Dal punto di vista architetturale si è scelto di utilizzare il pattern *Model View Controller (MVC)*. Poiché per produrre l'interfaccia grafica sono state utilizzate le librerie *JavaFX*, la scelta del pattern *MVC* è stata naturale in quanto quest'ultimo è nativamente implementato nelle librerie scelte.

Il pattern *MVC* permette il vantaggio di avere una separazione netta tra le seguenti componenti:

- **Model**: Il modello rappresenta la logica interna dell'applicazione. Questa parte gestisce la logica per il login, l'aggiunta, modifica o eliminazione dei dati in modo permanente su disco;
- **View**: la vista rappresenta la componente grafica e gestisce ciò che è visibile all'utente. La vista è rappresentata attraverso file *FXML* che ci restituiscono una composizione grafica attraverso pulsanti,tabelle,label, immagini ed elenchi;
- **Controller**: il controller permette la comunicazione tra vista e modello. Esso definisce infatti la logica applicativa, ovvero il comportamento del sistema a fronte degli stimoli esterni, che vengono rilevati attraverso dei listener che ascoltano la vista.

Di seguito una rappresentazione schematica dell'architettura del sistema



Singleton Pattern

Come detto prima, per gestire la creazione, elaborazione ed eliminazione dei dati è stata creata una classe *Model*. Considerando quindi il fatto che *Model* si troverà a gestire le collezioni di dati, è stato ritenuto opportuno permettere la creazione di una singola istanza di questa classe. Per realizzare tale comportamento è stato utilizzato il pattern *Singleton*, il quale appunto garantisce che all'interno dell'applicazione venga creata una sola istanza di questa classe.

Quando la classe viene istanziata avviene una lettura dei file *.txt* e ad ogni inserimento o modifica dei dati, tali aggiunte o modifiche vengono riportate sui file. Il vantaggio di poter applicare il pattern *Singleton* risiede nel poter richiedere facilmente l'istanza all'interno di un qualsiasi controller. Questo garantisce la presenza di un unico stato dell'istanza che evolve nel tempo e che riporta una sua copia memorizzata su disco tramite i file *txt*.

L'applicazione di tale pattern permette inoltre di trattare *Model* come un database, ovvero tutte le volte che viene richiesto un servizio quest'ultimo viene interrogato. Si può vedere dunque ogni metodo di *Model* come una sorta di query che all'occorrenza fornisce i dati richiesti.

Observer Pattern

Per l'implementazione dell'interfaccia grafica, come detto prima è stata usata la libreria *JavaFX*. Tale libreria per il suo funzionamento si basa sul meccanismo di *ActionListeners*, il quale può essere visto come un'implementazione del pattern Observer. I listeners infatti, come secondo il pattern Observer, sono dei meccanismi per notificare gli osservatori del cambio di stato dell'oggetto preso in considerazione (*subject*).

Test e Validazione

Per verificare la solidità del software prodotto sono state effettuate diverse attività di test come definito di seguito.

Ispezione del codice e confronto con documentazione

Questa fase consiste nella revisione del documento delle specifiche, confrontandolo con il codice prodotto ai fini di verificare la consistenza tra codice e specifiche. Tale fase consiste poi nell'analisi statica del codice, in modo tale da andare a cercare eventuali ottimizzazioni o cattivi usi dei pattern. Tale tipo di verifica è stata fatta in maniera continuativa durante tutto il processo di sviluppo.

Test degli sviluppatori

Questo tipo di validazione consiste nell'inserimento da parte dello sviluppatore di diversi tipi di input, sia corretti che errati, affinché si verifichi il corretto funzionamento del software.

In generale i test svolti possono essere divisi nelle seguenti macro-aree:

- Verifica del corretto passaggio di dati tra le viste e della verifica del loro corretto salvataggio in memoria;
- Inserimento di dati errati, input vuoti o stringhe malformate;
- Verifica della corretta reazione della vista, anche nel caso in cui l'utente anticipi il naturale flusso di esecuzione andando ad interagire con aree utili in seguito.

In particolare seguono la maggior parte dei test effettuati dagli sviluppatori:

Autenticazione

- Verifica del corretto funzionamento dell'autenticazione**:** i dati errati vengono respinti ed a fronte dei dati corretti viene mostrata la schermata iniziale.

Inserimento di un nuovo lavoratore

- Verifica che in caso di campi vuoti venga ritornato un errore;
- Controllo l'inserimento di lavoratori duplicati;
- Verifica del corretto aggiornamento della vista in relazione alle azioni dell'utente;
- Verifica che nome e cognome non contengano numeri;
- Verifica dell'inserimento di una mail corretta;
- Verifica che la data di nascita sia almeno 16 anni fa;
- Verifica che l'indirizzo sia formulato correttamente;
- Verifica che il numero di telefono abbia la lunghezza corretta (assumiamo esistano solo numeri italiani);
- Verifica di campi estranei nelle specializzazioni;
- Verifica che ci sia almeno una lingua e un contatto di emergenza.

Inserimento di un'esperienza lavorativa

- Verifica che in caso di campi vuoti venga ritornato un errore;
- Controlli sulle date (viene testato l'inserimento di esperienze più vecchie di 5 anni);
- Controllo che inizio fine ed azienda non esistano già;
- Controllo retribuzione lorda e netta;
- Controllo del corretto funzionamento della vista nel caso dell'aggiunta e rimozione delle mansioni;

- Verifica della reazione della vista quando si seleziona un'esperienza.

Inserimento delle Lingue parlate

- Verifica che ci sia almeno una lingua parlata;
- Verifica della reazione della vista ad ogni interazione.

Inserimento delle patenti

- Verifica della reazione della vista ad ogni interazione.

Inserimento di un contatto d'emergenza

- Controllo di valori errati nei campi;
- Controllo che non esista un contatto d'emergenza con la stessa mail;
- Verifica della reazione della vista ad ogni interazione.

Inserimento disponibilità

- Verifica che non esistano disponibilità nello stesso comune con date sovrapponibili;
- Controllo del corretto funzionamento della vista nel caso dell'aggiunta e rimozione delle disponibilità;
- Verifica della reazione della vista quando si seleziona una disponibilità.

Visualizza Esperienze

- Verifica della corretta reazione dell'interfaccia alla selezione di un'esperienza;
- Verifica che la ricerca veloce restituisca i risultati corretti;
- Verifica del corretto passaggio di parametri del lavoratore selezionato alle altre viste.

Ricerca

- Creazione di più lavoratori con specifiche caratteristiche in modo che, applicando specifici filtri di ricerca, i risultati mostrassero esattamente i lavoratori creati;
- Verifica la ricerca in and e or dei periodi di disponibilità;
- Applicazione di un filtro di ricerca e successiva modifica del lavoratore contenente quel determinato filtro di ricerca;
- Test del mantenimento dei filtri una volta riaperta la schermata di ricerca.

Test utente generico

In conclusione il software è stato sottoposto ad un test da parte di alcuni individui con limitata dimestichezza informatica ed assoluto distacco dallo sviluppo. In questa fase non si è cercato in nessun modo di guidare l'esperienza, in modo tale da non influenzare il risultato, lasciando navigare l'utente liberamente nel sistema. Non è stata data nessuna spiegazione sull'utilizzo del software, se non una generale indicazione dei suoi fini. Tale test ha avuto come scopo quello da una parte di rilevare errori non visibili dallo sviluppatore, dovuti da un suo bias nei confronti del programma, dall'altra di verificare l'intuitività dell'interfaccia grafica.

Scelte Progettuali

In questo capitolo vengono definite le scelte di progettazione attuate dagli sviluppatori nei casi in cui si presentassero più alternative con le quali implementare un determinato comportamento del programma, nel caso in cui le specifiche date non avessero definito come trattare quest'ultimo.

Innanzitutto si è deciso di utilizzare come identificativo per ogni lavoratore la propria mail, di conseguenza non è stato reso possibile l'inserimento di più lavoratori con la medesima mail.

Durante l'inserimento di un nuovo lavoratore è stato reso obbligatorio l'inserimento di tutti i dati personali, di almeno una lingua parlata, e di almeno un contatto di emergenza. Le altre informazioni è stato deciso di poterle inserire anche in un momento successivo.

Nel momento dell'inserimento delle esperienze lavorative è stato deciso di aggiungere un campo retribuzione netta giornaliera al fine di rendere più complete le informazioni fornite sull'esperienza.

Per quanto riguarda la fase di ricerca è stato scelto di aggiungere una ricerca veloce, che accetta come input una stringa e va ad effettuare istantaneamente la ricerca di essa all'interno dei lavoratori. Tale tipo di ricerca è qualcosa di facoltativo e che va eventualmente ad integrarsi alla vera e propria ricerca, chiamata filtraggio nel nostro caso. A tal proposito nella schermata di filtraggio dati si è scelto di non mescolare la ricerca in and e or, ma bensì di inserire un pulsante per poter scegliere il tipo di ricerca in quanto riteniamo che un eventuale utente possa trovare difficoltoso effettuare una ricerca dove gli and e or sono mescolati tra di loro.

Infine assumiamo che le credenziali del dipendente, così come i suoi dati siano già registrati a priori, ed eventualmente modificati o registrati direttamente nel database.