Progetto Lavoratori Stagionali

Dudine Pietro VR456578
Rossignolo Davide VR456503

Anno 2021/2022

Indice

Sezione 1

Introduzione

i. Consegna

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, viticoltore, 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).

ii. Metodologia di lavoro adottata

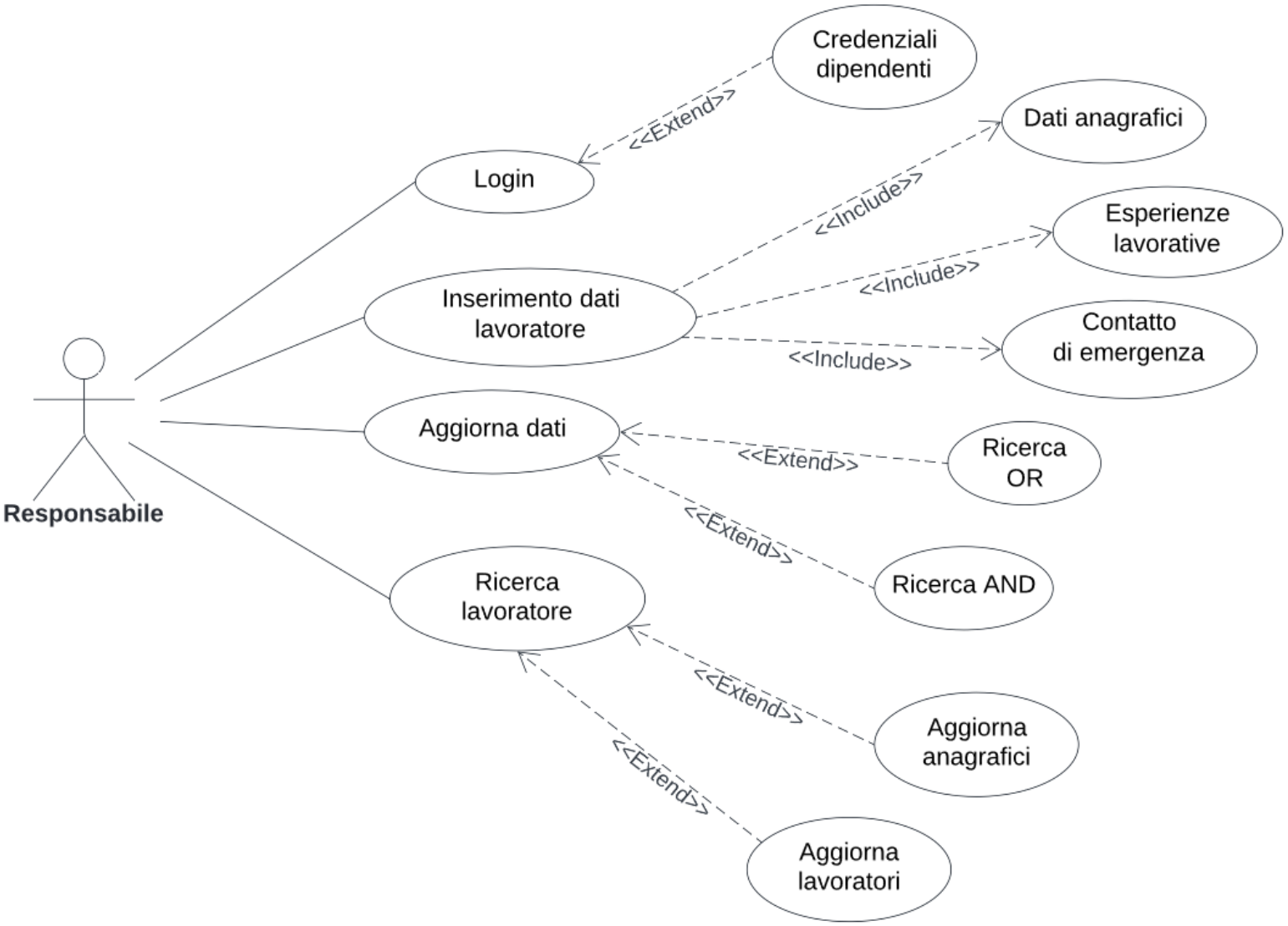
La metodologia di lavoro da noi adottata è stata di tipo Agile ed Incrementale. Tuttavia, si è cercato quanto più possibile di mantenere sequenziali le fasi di progettazione, implementazione e validazione; in modo da procedere sempre prima con la parte di progettazione e dopo con quella di implementazione. Queste fasi non sono sempre state lineari in quanto sono state anche incluse attività di refactoring sul materiale già costruito. Verso la fine di ogni ciclo si è proseguito con la documentazione e la fase di testing, in questa attività, si è raccolto il materiale UML generato dalla progettazione (principalmente diagrammi di classe) ed è stato aggiunto nel presente documento. Sono inoltre stati aggiunti ulteriori UML descrittivi (come ad esempio i sequence diagram). Prima di cominciare il ciclo principale, si è condotta la fase di analisi dei requisiti, generando i relativi use-cases e i diagrammi di attività, inoltre è stato prodotto lo sketch dell'interfaccia grafica in modo da avere quasi l'assoluta certezza di trovarsi in una situazione ben pianificata e concisa.
Per quanto riguarda l'implementazione, non sono state fatte particolari divisioni dei compiti o piani di sviluppo particolari, in quanto si è preferito avere sempre un feedback sul lavoro svolto. L'ordine di sviluppo che è stato seguito è stato di tipo prioritario andando ad aggiungere ciò che era ritenuto necessario man mano, inoltre sviluppo grafico e di back-end hanno sono stati svolti in parallelo, in quanto è stato deciso di seguito l'ordine in cui i vari frame si presentano all'utente

(quindi si è partiti dalla pagina di login per poi proseguire e suddividersi tutti gli altri frame della GUI).

Sezione 2

Struttura del progetto

In questa sezione sarà descritta la struttura generale del progetto e come esso si presenta all'utente finale. Verranno poi mostrati alcuni esempi di casi d'uso principali con relativi diagrammi di sequenza, per concludere sarà mostrato un diagramma riassuntivo delle attività generali.



i. Casi d’uso principali

Il diagramma dei casi d’uso sopra raffigurato rappresenta tutte le possibili operazioni che possono essere eseguite durante l’utilizzo del sistema. Di seguito vengono riportati alcuni casi d’uso tra i più importanti.

a. Caso d’uso: Login

Il dipendente per poter accedere deve essere registrato al portale, questa operazione verrà eseguita direttamente dal ufficio IT, che avrà il compito di inserire i dati necessari alla registrazione del dipendente direttamente nel database.

Precondizioni: il dipendente deve essere registrato al portale
Attore: Dipendente
Passi:

1. Il dipendente inserisce le credenziali fornitegli dall'ufficio IT
2. Se inserite correttamente entra nel portale.

Postcondizioni: il dipendente ora può usare il portale.

b. Caso d’uso: Inserimento dati lavoratore

Il dipendente accede alla pagina di inserimento dei dati di un lavoratore, dalla quale può inserire anche i contatti di emergenza, le disponibilità e le esperienze lavorative precedenti.

Precondizioni: il dipendente deve essere autenticato
Attore: Dipendente
Passi:

1. Il dipendente apre la pagina per inserire un nuovo lavoratore
2. Inserisce i dati anagrafici, vengono controllati ed inseriti nel sistema
3. Apre la pagina dei contatti di emergenza
4. Inserisce i contatti di emergenza, vengono controllati ed inseriti nel sistema
5. Apre la pagina delle esperienza lavorative
6. Inserisce le esperienze lavorative, vengono controllate ed inserite nel sistema
7. Apre la pagina delle disponibilità
8. Inserisce i vari periodi di disponibilità

Postcondizioni: un nuovo lavoratore è stato inserito

c. Caso d'uso: Modifica dati lavoratore

Il dipendente dopo aver cercato un lavoratore può modificare i dati legati al suo profilo.

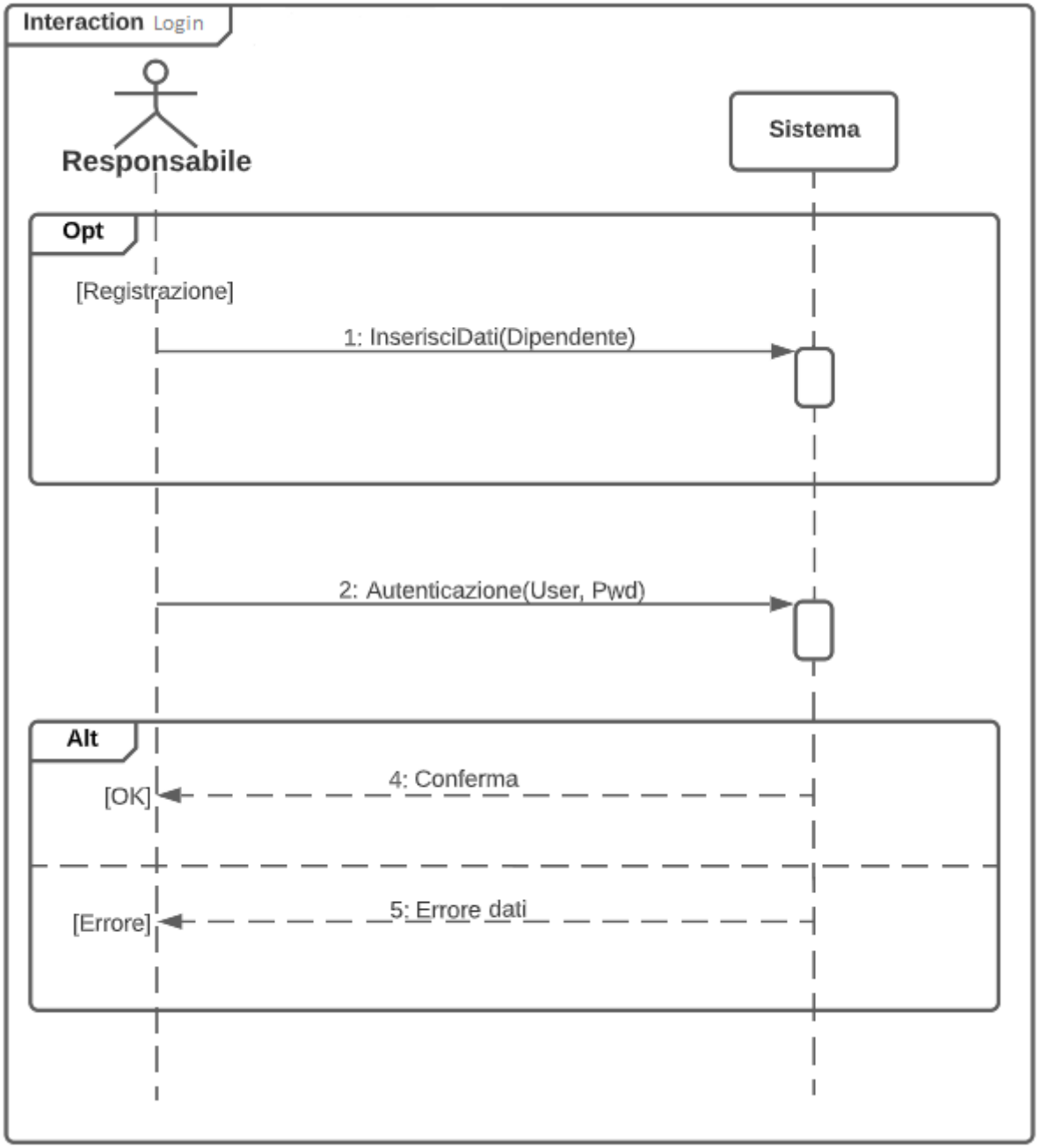
Precondizioni: il dipendente deve essere autenticato
Attore: Dipendente
Passi:

1. Il dipendente cerca il lavoratore
2. Apre la scheda del lavoratore
3. Modifica i dati anagrafici (se necessario), le modifiche sono controllate e sovrascritte alle precedenti nel sistema
4. Apre la pagina dei contatti di emergenza
5. Modifica i contatti di emergenza (se necessario), le modifiche sono controllate e sovrascritte alle precedenti nel sistema
6. Apre la pagina delle esperienza lavorative
7. Modifica le esperienze lavorative (se necessario), le modifiche sono controllate e sovrascritte alle precedenti nel sistema

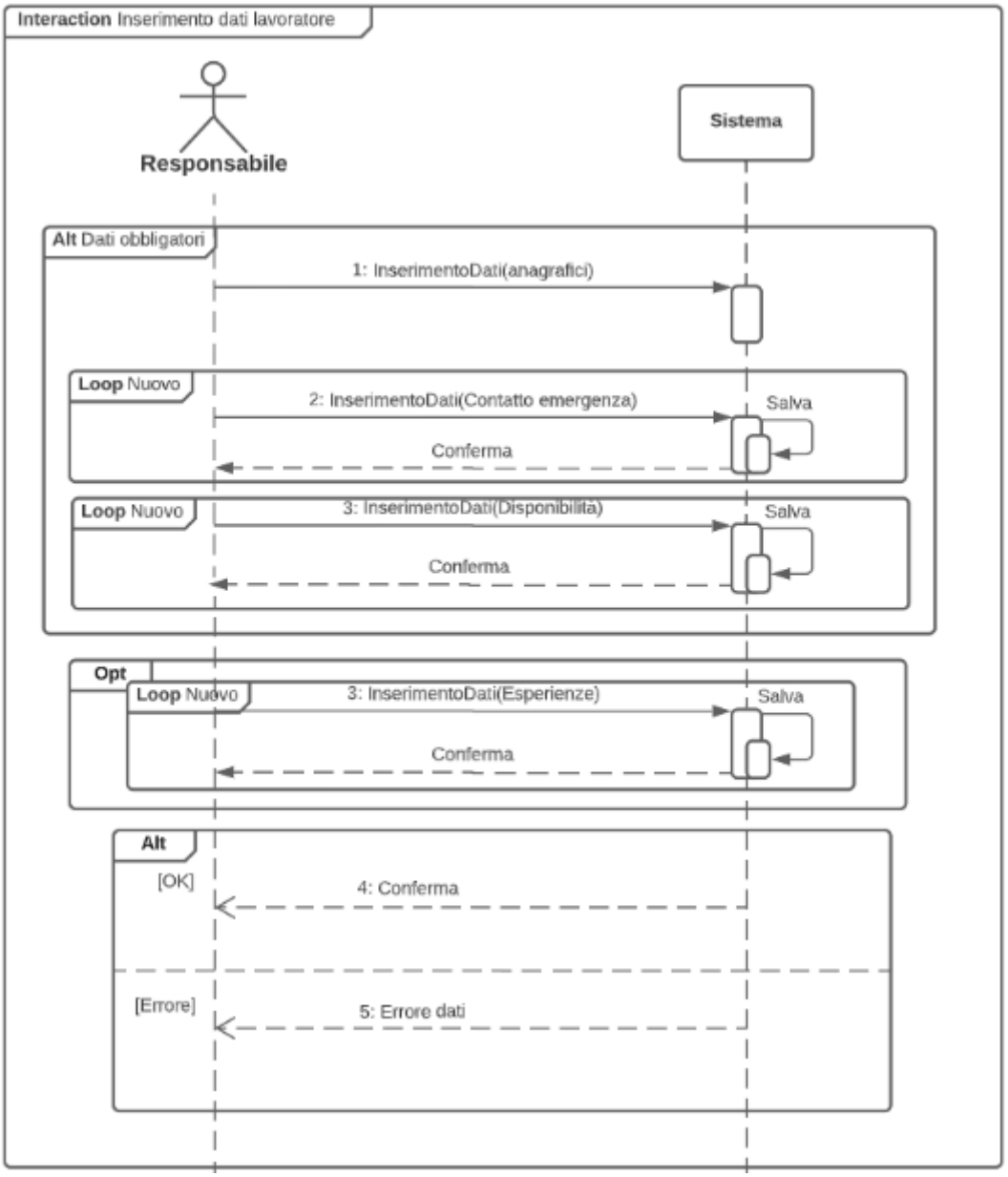
Postcondizioni: i dati lavoratore sono stati modificati

ii. Diagrammi di sequenza dei casi d'uso

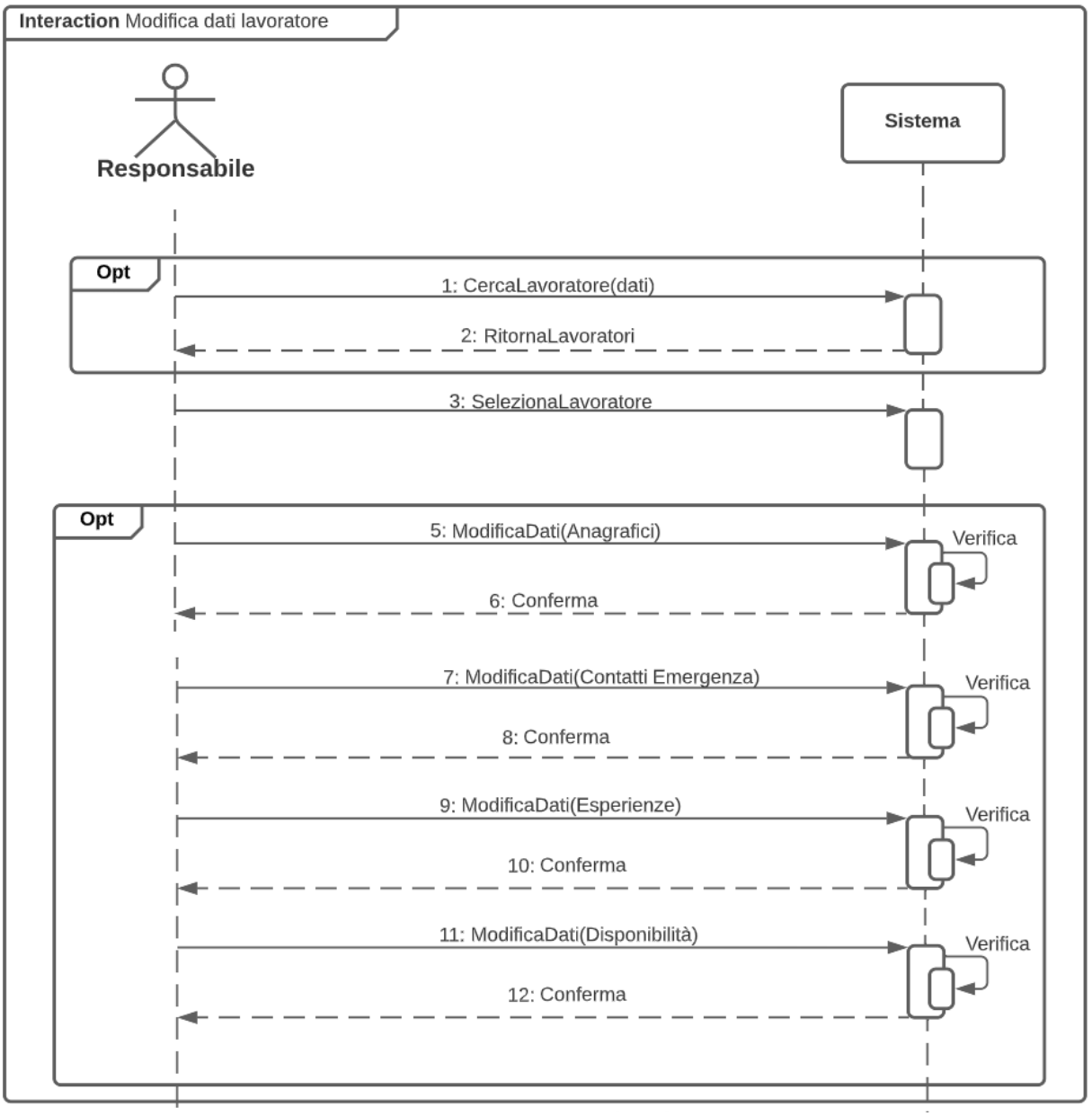
a. Diagramma di sequenza login



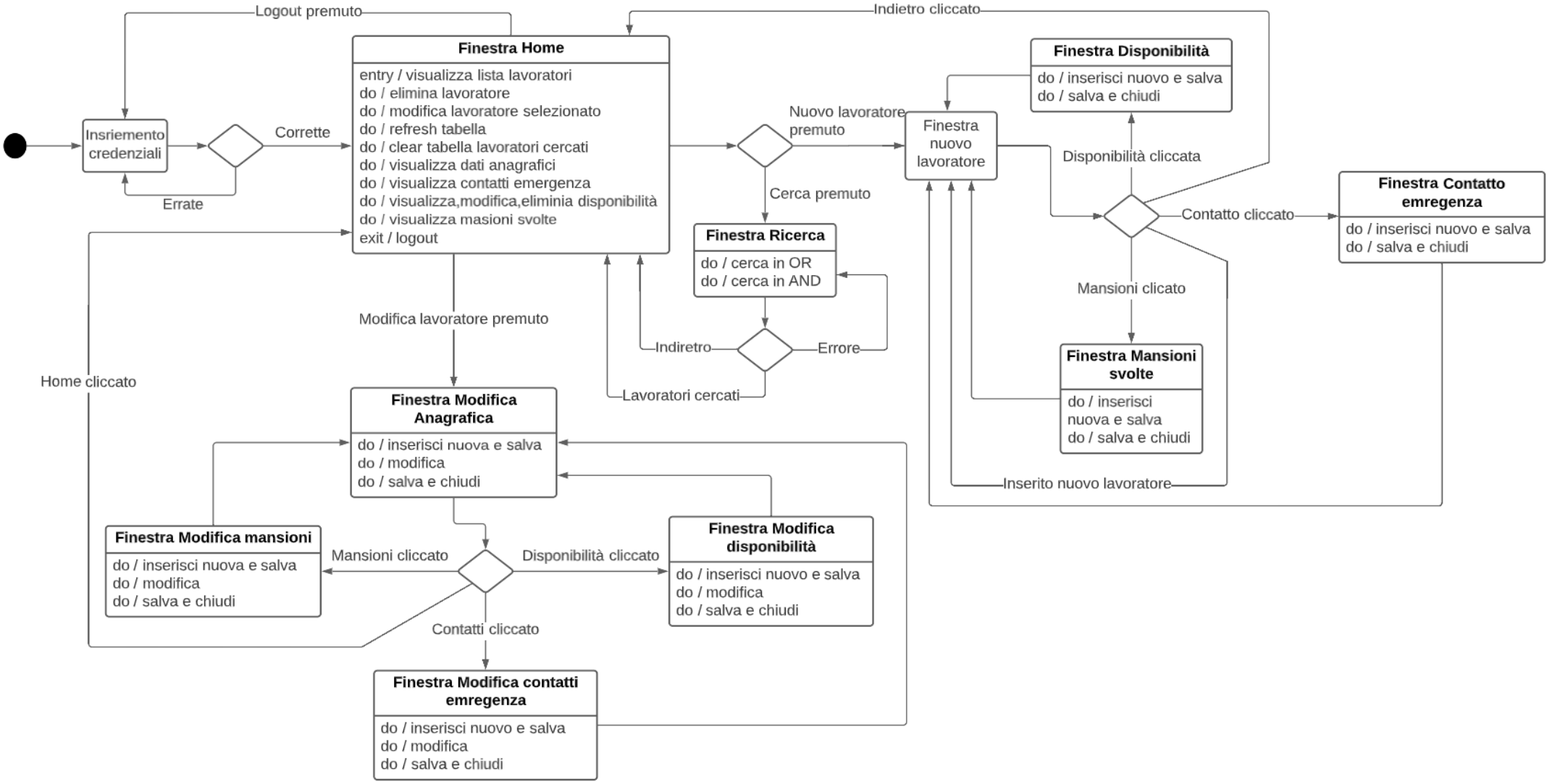
b. Diagramma di sequenza inserimento dati lavoratore



c. Diagramma di sequenza modifica dei dati



iii. Diagramma delle attività



Sezione 3

Implementazione

In questa sezione verranno riportati i metodi implementativi usati, i pattern architetturali ed i design usati per la realizzazione dell'interfaccia grafica.

i. Classi implementate

Classi per la gestione dei dati (Model)

- **Modello**: classe utilizzata per la rappresentazione, scrittura e modifica delle classi che rappresentano i dati

Classi frontend (View)

- **login**: pagina di accesso al sistema
- **home**: home del programma
- **search**: apre i filtri della ricerca

- Classi per l'aggiunta e modifica dei dati

- **addExperience**: gui che permette di aggiungere un'esperienza al lavoratore
- **addWorker:** view che permette di aggiungere un lavoratore al sistema
- **addAv**: permette di aggiungere un periodo di disponibilità a lavorare
- **addContact:** permette di aggiungere i contatti di emergenza
- **modify**: permette di modificare le informazioni anagrafiche di un utente
- **modifyAv**: permette di modificare le informazioni delle disponibilità di un lavoratore
- **modifyContact**: permette di modificare le informazioni di un contatto di emergenza selezionato di un lavoratore
- **modifyExperience**: permette di modificare le esperienze di un dato lavoratore

- Classi per la visualizzazione dei dati

- **viewContacts**: permette di visualizzare tutti i contatti di emergenza di un dato lavoratore
- **viewInfo**: permette di visualizzare le informazioni anagrafiche complete di un dato lavoratore
- **viewAv:** permette di visualizzare i periodi di disponibilità di un dato lavoratore
- **viewExperience**: permette di vedere tutte le esperienze lavorative precedenti di un dato lavoratore

Classi Controller

- **homeController**
- **loginController**
- **searchController**

- Controller per view di aggiunta e modifica dati

- **addExperienceController**: controller per la finestra per l'aggiunta delle esperienze
- **addWorkerController**: controller per la finestra di aggiunta dei lavoratori
- **addAvController:** controller per la finestra di aggiunta delle disponibilità

- **addContactController**: controller per la finestra di aggiunta dei contatti di emergenza
- **modifyController**: controller per view di modifica anagrafica

- Controller per view di visualizzazione dati

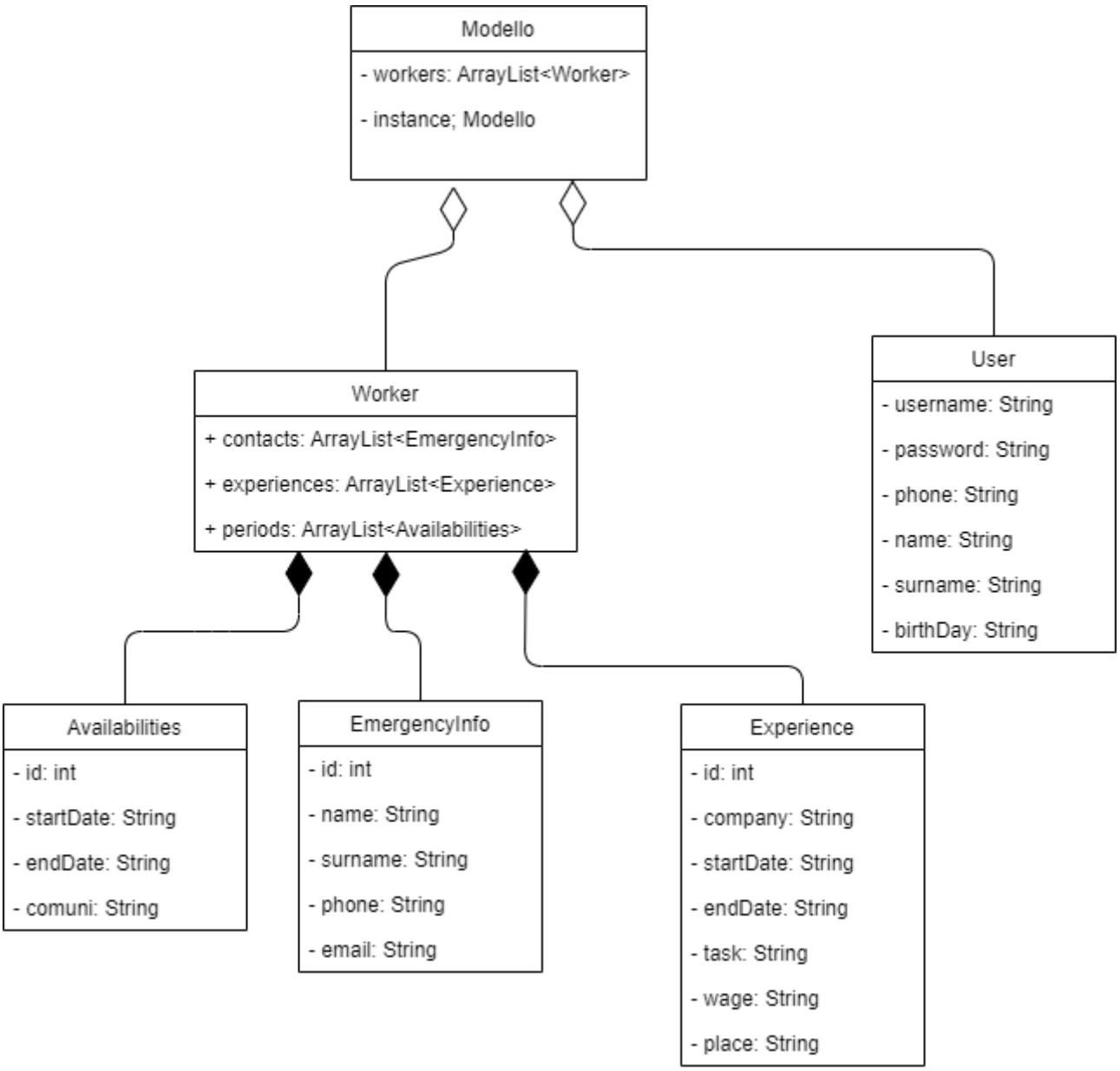
- **viewAvController:** controller per la finestra di visualizzazione delle disponibilità
- **viewContactController**: controller per la finestra di visualizzazione dei contatti
- **viewInfoController**: controller per la finestra di visualizzazione dell'anagrafica
- **viewExperienceController**: controller per la finestra di visualizzazione delle esperienze lavorative precedenti
- **modifyAvController**: controller per la finestra di modifica delle disponibilità di un lavoratore
- **modifyContactController**: controller per la finestra di modifica di un contatto di emergenza selezionato di un lavoratore
- **modifyExperienceController**: controller per la finestra di modifica delle precedenti esperienze lavorative di un lavoratore

Classi che rappresentano i dati da memorizzare

- **Availabilities**: questa classe rappresenta un periodo di disponibilità fornito dal lavoratore stagionale
- **EmergencyInfo**: rappresenta un contatto di emergenza del lavoratore
- **Experience**: rappresenta una esperienza lavorativa svolta negli anni precedenti
- **User**: rappresenta un dipendente dell'azienda (colui che usa il sistema)
- **Worker**: rappresenta un lavoratore stagionale

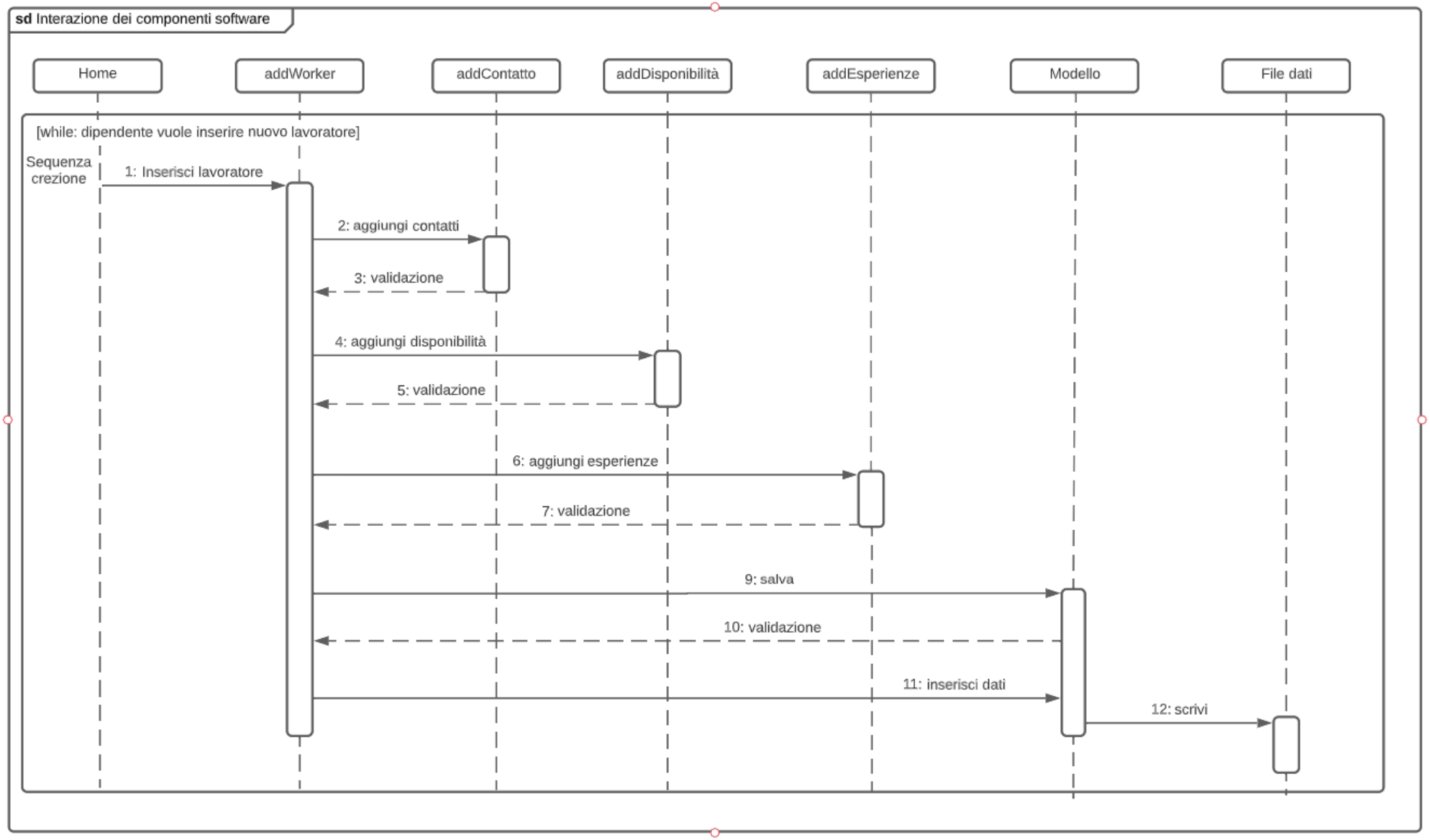
ii. Diagramma delle classi

Di seguito il diagramma delle classi del progetto, sono state escluse le classi della GUI poiché non sono indispensabili per capire la struttura del progetto.
Sono inoltre stati omessi i campi anagrafici nella classe worker per rendere il diagramma più comprensibile, i metodi delle varie classi sono stati omessi per lo stesso motivo.

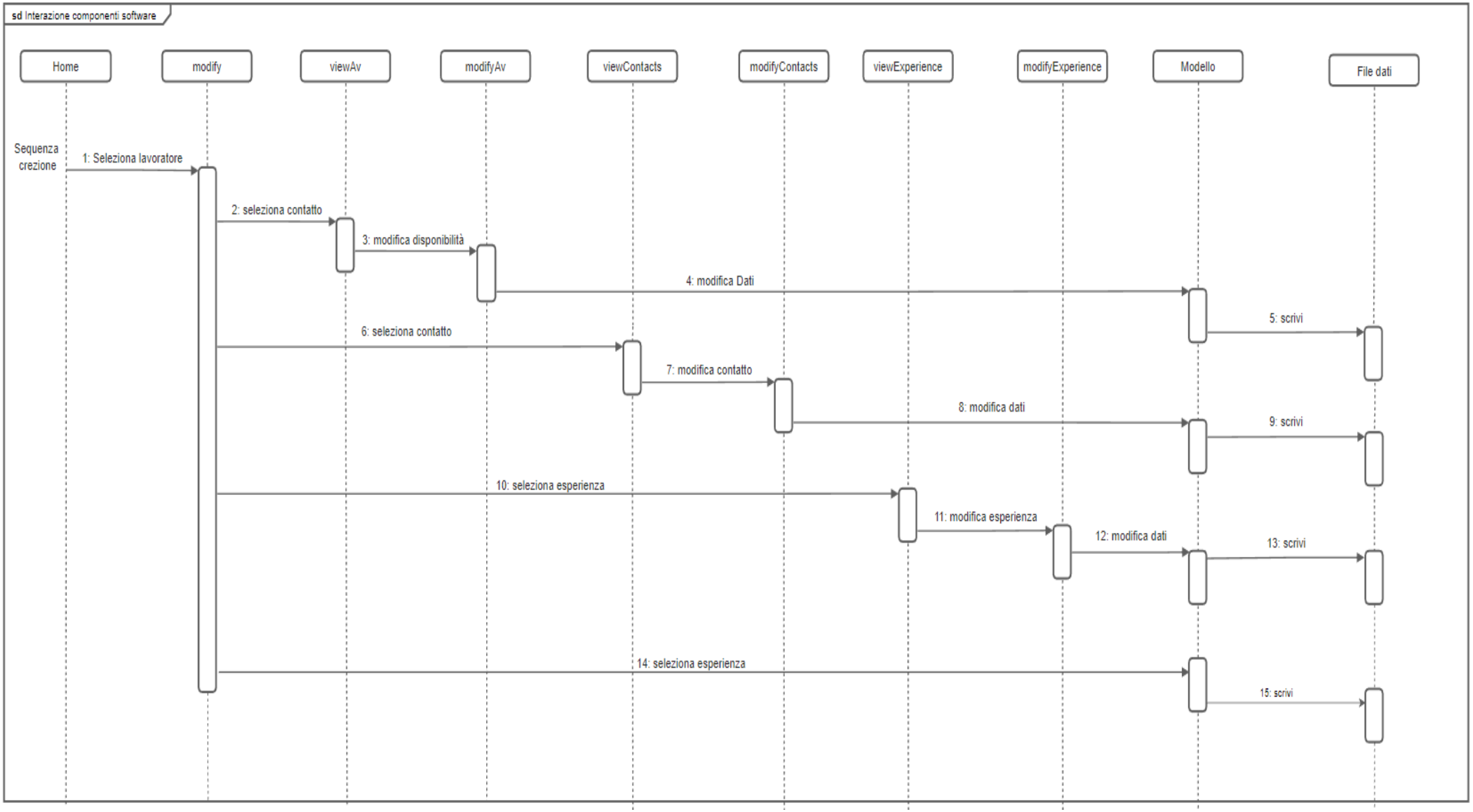


iii. Diagramma di sequenza del software

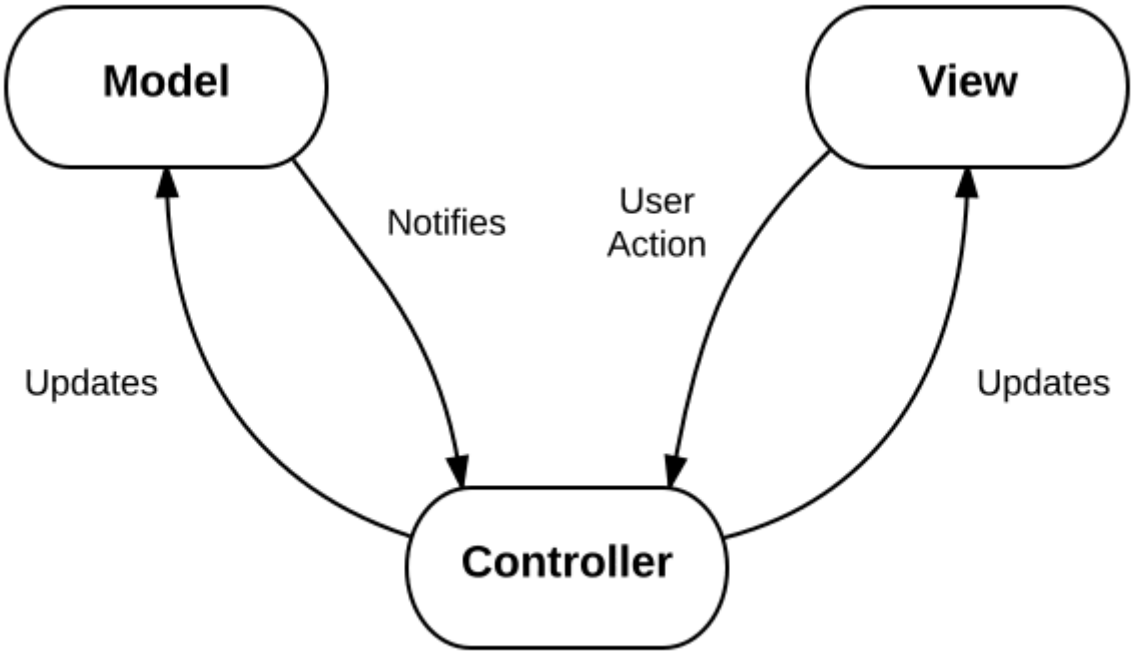
a. Inserimento nuovo lavoratore



b. Modifica lavoratore selezionato



iv. Pattern architetturale: MVC



Come pattern architetturale è stato deciso di usare il Model View Controller o MVC, questo perché permette di non avere dipendenza tra il front-end ed il back-end sfruttando un controllore il quale avrà il compito di unire le due parti. Questo permette di avere un codice più leggibile e più facile da mantenere.
Model View Controller si divide in:

- Model: ha il compito di gestire tutta la parte relativa ai dati, che va dalla lettura/scrittura fino alla manipolazione di essi
- View: ciò che l'utente vede e con cui interagisce, è stato deciso di sviluppare le View in FXML.
- Controller: colui che gestisce tutte le interazioni tra Model e View, ha il compito di controllare che i dati inseriti siano corretti.

NOTA: durante la progettazione è stato deciso di creare per ogni Modello una View ed un Controller in modo da massimizzare la leggibilità e la manutenzione del codice.

v. Pattern architetturale: Singleton

Per gestire creazione, elaborazione ed eliminazione dei dati e' stata creata una classe chiamata Modello che e' implementata seguendo il pattern Singleton. Il pattern Singleton garantisce che all'interno della applicazione venga creata una sola istanza di questa classe.
Quando la classe viene istanziata viene eseguita la lettura dei dati dai vari file csv e si crea quindi una lista dei lavoratori già presenti nel sistema.
Il vantaggio del pattern singleton è che permette di poter richiedere facilmente l'istanza all'interno di un qualsiasi controller garantendo che non saranno presenti stati indeterminati o multipli del sistema: è sempre presente un unico stato che evolve nel tempo dato che ad ogni modifica i dati vengono scritti sui file csv presenti nella memoria secondaria.
Il motivo per cui è stato scelto questo tipo di pattern è la possibilità di trattare la classe Modello come se fosse un database infatti quando viene richiesto un servizio andiamo direttamente a

controllare i dati presenti nel modello. Possiamo quindi paragonare i metodi di questa classe a delle query che all'occorrenza forniscono i dati richiesti.
Grazie a tale pattern è stato inoltre possibile implementare la visualizzazione in forma tabulare dei lavoratori e il rispettivo sistema di ricerca, tramite interfaccia grafica è infatti possibile eseguire ricerche in and e or.

Sezione 4

Test

In questa sezione verranno spiegate le principali attività che sono state svolte per testare il software durante lo sviluppo e il risultato della fase di testing con un utente generico.

i. Test del software

Durante lo sviluppo incrementale del progetto abbiamo svolto parallelamente anche le attività di testing. Grazie ad un file di dati campione abbiamo potuto testare ogni parte di codice che è stata prodotta; particolare attenzione è stata posta sulla fase di testing della classe modello in quanto questa è la classe principale con i metodi che gestiscono tutti i dati del progetto.

ii. Test con utente generico

Il software è stato poi testato con degli utenti generici ignari del funzionamento per testare l'intuitività del prodotto.
Da questa fase di testing abbiamo tratto delle conclusioni che ci hanno portato a fare delle modifiche ad alcune label e ad alcuni pulsanti in quanto non era molto chiaro il loro utilizzo.
Grazie alle osservazioni che ci sono state fatte durante il testing abbiamo modificato anche le pagine di modifica dei dati e di inserimento di un lavoratore.
Dopo aver reso il programma più intuitivo e user-friendly abbiamo riproposto all'utente generico il programma e abbiamo potuto notare che il tempo di utilizzo si è ridotto notevolmente.