



Progetto di Ingegneria del Software e Sicurezza Informatica
Relazione



2018/2019

Roberto Broccoletti, Paolo Di Massimo, Luca Luzi
Davide Manco, Iacopo Pacifici

1. Introduzione

Il progetto in questione ha come scopo principale riuscire a realizzare un'applicazione Web che permetta di digitalizzare ed automatizzare il processo di gestione di appalti pubblici. Infatti, ancora oggi la gestione e la contabilizzazione di un appalto edilizio vengono fatte per la maggioranza in modo manuale, grazie alla compilazione di diversi documenti che, molto spesso, richiedono molto spreco di tempo e personale. Il nostro primo scopo è appunto quello di automatizzare il tutto, cercando di velocizzare i vari step grazie all'aiuto di nuove tecnologie. In particolare, abbiamo ritenuto essere di rilevante importanza per il nostro obiettivo l'utilizzo di smart contract, ovvero protocolli di transazione in grado di auto eseguirsi e auto mutarsi in base al verificarsi di certe condizioni. Infatti, in questo modo, i diversi attori coinvolti nel contratto d'appalto vedranno diminuire in modo sostanziale le loro attività.

Anche se gli smart contract sono separati dal concetto di Blockchain, trovano proprio con le Blockchain un perfetto utilizzo, in quanto, quest'ultime riescono a garantire diverse proprietà (quali affidabilità, trasparenza, non ripudio, confidenzialità e integrità) che rafforzano maggiormente la potenza dei contratti intelligenti.

Per questo motivo abbiamo scelto di realizzare una Dapp (Decentralized Application) basandosi proprio sulla tecnologia della Blockchain. Nel nostro caso la scelta è ricaduta su una Blockchain privata, Quorum, in quanto riesce a garantire una maggior sicurezza e privatezza dei dati al suo interno.

2. Matrice delle responsabilità

Attività	Roberto Broccoletti	Paolo Di Massimo	Luca Luzi	Davide Manco	Iacopo Pacifici
Prima intervista	X	X	X	X	X
Analisi requisiti e fonti	X	X	X	X	X
Creazione story card	X	X	X	X	X
Convalida story card	X	X	X	X	X
Acquisizione conoscenze blockchain	X	X	X	X	X
Acquisizione conoscenze Quorum	X	X	X	X	X
Acquisizione conoscenze smart contract	X	X	X	X	X

Acquisizione conoscenze Solidity	X	X	X	X	X
Acquisizione conoscenze Truffle	X	X	X	X	X
Acquisizione conoscenze NodeJS	X	X	X	X	X
Acquisizione conoscenze Express	X	X	X	X	X
Acquisizione conoscenze Angular	X	X	X	X	X
Acquisizione conoscenze Bootstrap	X	X	X	X	X
Analisi e progettazione Story 1	X	X			
Analisi e progettazione Story 2				X	X
Analisi e progettazione Story 3	X	X			
Analisi e progettazione Story 4				X	X
Analisi e progettazione Story 5	X	X			
Analisi e progettazione Story 6				X	X
Analisi e progettazione Story 7	X	X			
Analisi e progettazione Story 8				X	X
Analisi e progettazione Story 9	X	X			
Analisi e progettazione Story 10				X	X
Implementazione Blockchain privata	X				X
Implementazione DBMS		X		X	
Progettazione storie lato frontend	X	X			
Progettazione storie lato server				X	X
Progettazione storie lato smart contract		X			X
Implementazione meccanismo token	X			X	
Test	X	X		X	X
Documentazione finale	X	X		X	X
Manuale d'installazione	X	X		X	X

3. Definizione delle fonti

Le fonti alle quali ci siamo attenuti sono:

- 1) Stakeholder: il prof. Luca Spalazzi, il prof. Francesco Spegni, il prof. Berardo Naticchia
- 2) Documenti preesistenti: Sito “biblus.acca.it” dal quale abbiamo attinto il dettaglio dei documenti da realizzare

Durante lo svolgimento ci siamo basati su documentazioni trovate sul web per l'utilizzo delle tecnologie necessarie per il progetto quali Quorum, Solidity, Truffle, NodeJS, Express, Angular, Bootstrap.

4. Raccolta dei requisiti

I requisiti utente sono:

- **Utente:** sono tutti i soggetti che utilizzeranno il software, ovvero:
 - 1) Direttore dei Lavori: è la figura professionale necessaria per seguire l'andamento regolare delle lavorazioni in cantiere
 - 2) RUP (Responsabile Unico del Procedimento): è un incaricato della ditta appaltante a seguire i lavori
 - 3) Ditta Appaltatrice: è la ditta incaricata per la realizzazione dei lavori
- **Ambiente:** il sistema può essere utilizzato in un qualsiasi momento e luogo da ognuno degli utenti sopra specificati, ovviamente con una necessaria connessione internet
- **Obiettivo:** l'utente può svolgere in maniera elettronica le operazioni di gestione e monitoraggio di un cantiere, sfruttando, inoltre, i vantaggi derivanti dalla tecnologia Blockchain

I requisiti di sistema sono le funzioni che deve avere il nostro sistema e i vincoli che deve rispettare. Questi sono classificati in:

- **Requisiti di dominio:** si deve progettare e sviluppare un'applicazione web che permetta agli utenti di memorizzare, in modo sicuro e persistente, informazioni all'interno della Blockchain.

- **Requisiti funzionali:**
 - In seguito ad un login, la nostra applicazione web si deve connettere alla Blockchain in modo da permettere ai vari attori di poter svolgere le loro attività designate. In particolare:
 - Il Direttore dei lavori potrà:
 - Inserire, modificare e visualizzare i dati relativi al giornale dei lavori
 - Inserire, modificare e visualizzare i dati relativi al libretto delle misure
 - Sciogliere la riserva presentata dalla ditta riguardo un rigo di un libretto delle misure
 - Inserire e visualizzare i dati relativi allo stato di avanzamento dei lavori
 - Inserire i dati riguardanti un certificato di pagamento all'interno di un registro di pagamento
 - Visualizzare i dati relativi al registro di contabilità
 - Visualizzare i dati relativi al certificato di pagamento
 - Stampare i resoconti dei vari documenti
 - La ditta potrà:
 - Visualizzare i dati relativi al giornale dei lavori
 - Visualizzare i dati relativi al libretto delle misure
 - Apporre la firma o la firma con riserva sui vari righi del libretto delle misure
 - Visualizzare i dati relativi al registro di contabilità
 - Visualizzare i dati relativi allo stato di avanzamento dei lavori
 - Visualizzare i dati relativi al certificato di pagamento
 - Stampare i resoconti dei vari documenti
 - Il RUP potrà:
 - Inserire, modificare e visualizzare nuovi lavori all'istanziazione del contratto
 - Inserire, modificare e visualizzare la rata di acconto all'istanziazione del contratto
 - Inserire e visualizzare i dati relativi al registro di contabilità
 - Inserire e visualizzare i dati relativi ad un certificato di pagamento
 - Visualizzare i dati relativi al giornale dei lavori
 - Visualizzare i dati relativi al libretto delle misure
 - Visualizzare i dati relativi allo stato di avanzamento dei lavori
 - Inserire e visualizzare i dati relativi al certificato di pagamento
 - Stampare i resoconti dei vari documenti
 - L'applicazione utilizza un database locale in modo da memorizzare dati che non sarebbero memorizzabili all'interno di una Blockchain, ad esempio foto o video

- **Requisiti non funzionali:**

- Il sistema è di tipo client - server
- Il server adotta un'architettura orientata alle risorse (servizi REST)
- Il server per memorizzare i dati riguardanti la messaggistica utilizzerà un database relazionale
- Il server per memorizzare i dati riguardanti il cantiere utilizzerà una Blockchain
- I dati inviati e ricevuti sono tutti in formato JSON
- Per accedere all'applicazione è necessario un browser ed una connessione internet

5. Glossario

Termine	Descrizione	Sinonimi	Collegamenti
Web App	Consiste nell'applicazione di gestione dei cantieri che stiamo realizzando	App, Applicazione	
Utente	Utilizzatori della web app	Direttore dei lavori Ditta appaltatrice Responsabile unico del procedimento	Login, Credenziali di accesso
Direttore dei lavori	Utente facente parte della gestione del cantiere	DL, direttore	
Responsabile unico del procedimento	Utente facente parte della gestione del cantiere	RUP	
Ditta appaltatrice	Utente facente parte della realizzazione del progetto in cantiere	Ditta	
Server	Sistema informatico di gestione dati provenienti dalla web app		Utente, Login, Database, Blockchain

Blockchain	Database distribuito utilizzato per elaborare e memorizzare i dati di gestione del cantiere	Quorum, Blockchain privata	Server
Database	Sistema informatico utilizzato per la memorizzazione di dati non memorizzabili all'interno di una blockchain (quali fotografie e video) e per dati utili alla messaggistica della web app	DB, DBMS	Server, Blockchain
Login	Procedura di accesso alla web app	Accesso	Utente, Server, Blockchain
Credenziali di accesso	Nome utente e password per accedere alla web app	Credenziali	
Lavorazione	Indica un lavoro contenuto all'interno di un contratto d'appalto che dovrà essere completato dalla ditta appaltatrice	Lavoro, misura	Giornale dei lavori, Libretto delle misure, Registro di contabilità, Stato di avanzamento dei lavori, Utente
Giornale dei lavori	Documento di gestione contenente dati riguardanti le lavorazioni effettuate giornalmente in cantiere	Giornale	Utente, Server, Blockchain
Libretto delle misure	Documento di gestione contenente dati relativi a precise lavorazioni	Libretto	Utente, Server, Blockchain

Registro di contabilità	Documento di contabilità contenente dati relativi a precise lavorazioni	Registro	Utente, Server, Blockchain
Stato di avanzamento dei lavori	Documento di gestione contenente dati relativi all'avanzamento delle lavorazioni	SAL	Utente, Server, Blockchain
Certificato di pagamento	Documento che certifica il dovuto pagamento di un certo importo da parte del committente dei lavori verso la ditta appaltatrice	Certificato	Utente, Server, Blockchain
Rata di acconto	È un valore monetario che, una volta superato, sancisce la creazione di un nuovo SAL	Soglia	Stato di avanzamento dei lavori

6. Analisi dei requisiti

Per quanto riguarda l'analisi dei requisiti siamo andati ad utilizzare il modello XP (eXtreme Programming) che prevede innanzitutto la creazione di storie utente e poi la suddivisione di queste in task card.

Per pura semplicità di rappresentazione abbiamo riportato le varie story card, seguite immediatamente dalle task card che abbiamo realizzato.

In totale sono state definite 10 storie utente, suddivise in 2 differenti release. Per questioni di tempistica e di non primaria necessità, abbiamo escluso da queste 2 release la story numero 7, infatti, in seguito ad un colloquio con alcuni degli stakeholder si è rivelata essere marginale e non fondamentale per i nostri scopi. Abbiamo però deciso di riportarla comunque all'interno della presente relazione per poter essere magari inserita in una successiva release.

5.1. Story Card 1

Release: 1	Story ID: 1
Story Tag: Inserimento lavorazioni e rata di acconto da parte del RUP	
Descrizione: il RUP ha come primo compito inserire le lavorazioni che saranno realizzate nel corso del progetto architettonico ed inoltre la rata di acconto.	Dettaglio: In seguito al primo login, il RUP dovrà inserire le lavorazioni necessarie al progetto. Queste saranno formate da: <ul style="list-style-type: none">• Codice identificativo della lavorazione• Nome della lavorazione• Costo della lavorazione Al termine di questo, dovrà anche inserire un valore monetario che sancisce la rata di acconto da utilizzare durante la gestione del progetto. Dopo aver inserito tutti i dati, si avrà un resoconto dei dati inseriti e la possibilità di poter terminare questa fase, con il conseguente inizio del contratto d'appalto.

5.1.1. Task Card 1

Story ID: 1	Task ID: 1.1
Task tag: Studio Blockchain	Software Engineer: Tutti
Descrizione: <ul style="list-style-type: none">• Studio della tecnologia Blockchain in generale• Studio Blockchain Quorum• Studio della Blockchain 7nodes fornitaci• Studio della creazione dell'infrastruttura della Blockchain• Installazione Vagrant• Installazione Virtual Box• Avvio prima Blockchain	

Story ID: 1	Task ID: 1.2
Task tag: Studio Smart Contract	Software Engineer: Tutti
Descrizione:	
<ul style="list-style-type: none"> • Studio di Solidity • Studio degli Smart Contract forniti 	

Story ID: 1	Task ID: 1.3
Task tag: Schede CRC	Software Engineer: Roberto, Paolo
Descrizione:	
<ul style="list-style-type: none"> • Preparazione delle schede CRC • Operazioni di selezione e disambiguazione delle schede CRC 	

Story ID: 1	Task ID: 1.4
Task tag: Analisi orientata alle interfacce	Software Engineer: Roberto, Paolo
Descrizione:	
<ul style="list-style-type: none"> • Selezione schede CRC degli utenti • Controllo delle responsabilità che devono essere tramutate in interfacce 	

Story ID: 1	Task ID: 1.5
Task tag: Mockup e modelli	Software Engineer: Roberto, Paolo
Descrizione:	
<ul style="list-style-type: none"> • Creazione mockup • Creazione dei modelli di presentazione, navigazione e dialogo 	

Story ID: 1	Task ID: 1.6
Task tag: Analisi orientata alle classi	Software Engineer: Roberto, Paolo
Descrizione:	
<ul style="list-style-type: none"> • Classificare schede CRC in classi e attributi • Definire le relazioni tra classi e attributi • Creazione diagramma delle classi di analisi • Realizzare il diagramma dei package di analisi 	

Story ID: 1	Task ID: 1.7
Task tag: Analisi orientata ai comportamenti	Software Engineer: Roberto, Paolo
Descrizione:	
<ul style="list-style-type: none"> • Creazione diagramma delle sequenze e di stato del flusso di lavoro presentato 	

Story ID: 1	Task ID: 1.8
Task tag: Implementazione smart contract per funzionalità per le lavorazioni	Software Engineer: Davide, Iacopo
Descrizione:	
<ul style="list-style-type: none"> • Creazione smart contract Misure.sol • Implementazione struct in grado di contenere lavorazioni • Implementazione di funzioni di aggiunta di una lavorazione (con controllo che non venga inserita una lavorazione identica ad una già presente) • Implementazione funzione di modifica (con sovrascrittura) di una lavorazione • Implementazione funzione di visualizzazione delle lavorazioni 	
Test:	
<ul style="list-style-type: none"> • Inserimento di dati corretti -> Inserimento del dato • Inserimento di dati errati (scambio tipi di dato) -> Messaggio di errore • Inserimento di dati parziali -> Messaggio di errore • Inserimento di una misura con codice già inserito -> Messaggio di errore • Modifica di un dato -> Modifica del dato • Terminazione inserimento delle misurazioni -> Schermata di inserimento della soglia 	

Story ID: 1	Task ID: 1.9
Task tag: Implementazione smart contract per funzionalità per la rata di acconto	Software Engineer: Davide, Iacopo
Descrizione:	
<ul style="list-style-type: none"> • Implementazione funzione di aggiunta della rata di acconto • Implementazione funzione di modifica della rata di acconto • Implementazione funzione di visualizzazione della rata di acconto 	
Test:	
<ul style="list-style-type: none"> • Inserimento della rata -> Inserimento avvenuto con successo e visualizzazione del valore • Inserimento della rata anche se già inserita in precedenza -> Messaggio di errore • Modifica della rata -> Modifica della rata e visualizzazione del nuovo valore 	

Story ID: 1	Task ID: 1.10
Task tag: Creazione progetto Truffle	Software Engineer: Davide, Iacopo
Descrizione:	
creazione dell'ambiente Truffle per la migrazione dei contratti all'interno della blockchain	

Story ID: 1	Task ID: 1.11
Task tag: Implementazione file di migrazione del contratto	Software Engineer: Roberto, Paolo
Descrizione:	
per poter migrare il contratto all'interno della blockchain si necessita del file di migrazione	
<ul style="list-style-type: none"> • Creazione file di migrazione per il contratto Misure.sol 	
Test:	
<ul style="list-style-type: none"> • Migrazione Misure.sol -> Controllo su Blockchain di avvenuta migrazione 	

Story ID: 1	Task ID: 1.12
Task tag: Studio server	Software Engineer: Tutti
Descrizione:	
<ul style="list-style-type: none"> • Studio Node.js • Studio Express • Studio Web3.js • Installazione Node.js 	

Story ID: 1	Task ID: 1.13
Task tag: Implementazione lato server inserimento, modifica, visualizzazione dati	Software Engineer: Roberto, Paolo
Descrizione:	
<ul style="list-style-type: none"> • Creazione funzione di POST per richiamare la funzione contenuta nello smart contract per l'inserimento di una lavorazione • Creazione funzione di POST per richiamare la funzione contenuta nello smart contract per la modifica di una lavorazione • Creazione funzione di GET per richiamare la funzione contenuta nello smart contract per la visualizzazione di una lavorazione • Creazione funzione di POST per richiamare la funzione contenuta nello smart contract per l'inserimento della rata di acconto • Creazione funzione di POST per richiamare la funzione contenuta nello smart contract per la modifica della rata di acconto • Creazione funzione di GET per richiamare la funzione contenuta nello smart contract per la visualizzazione della rata di acconto 	
Test:	
<ul style="list-style-type: none"> • Chiamata ad una funzione del contratto -> Comportamento desiderato della funzione • Chiamata errata ad una funzione -> Messaggio di errore • Dati inseriti non corretti -> Messaggio di errore 	

Story ID: 1	Task ID: 1.14
Task tag: Implementazione lato server funzione che avvii la gestione del contratto d'appalto	Software Engineer: Roberto, Paolo
Descrizione:	
in seguito all'inserimento da parte del RUP dei dati, il contratto d'appalto può essere definito come iniziato, quindi dovrà essere creata una funzione che al termine dell'inserimento vada a migrare tutti i contratti rimanenti all'interno della Blockchain.	
Test:	
<ul style="list-style-type: none"> Terminazione inserimento della rata di acconto -> Instanziazione contratti e schermata principale 	

Story ID: 1	Task ID: 1.15
Task tag: Studio frontend	Software Engineer: Tutti
Descrizione:	
<ul style="list-style-type: none"> Studio Angular Studio Bootstrap Installazione Angular 	

Story ID: 1	Task ID: 1.16
Task tag: Implementazione lato frontend interfaccia per gli inserimenti	Software Engineer: Davide, Iacopo
Descrizione:	
dovranno essere implementate delle interfacce per:	
<ul style="list-style-type: none"> Form inserimento lavorazione Form modifica lavorazione Visualizzazione lavorazioni inserite Form inserimento rata di acconto Form modifica rata di acconto Visualizzazione rata di acconto 	

5.2. Story Card 2

Release: 1	Story ID: 2
Story Tag: Inserimento e modifica giornale dei lavori	
Descrizione: il giornale dei lavori è lo strumento per riportare di giorno in giorno gli avvenimenti del cantiere. Solamente il direttore dei lavori può aggiungere un nuovo rigo del giornale o modificarne uno esistente. Può essere consultato da tutti gli attori.	Dettaglio: solamente il direttore dei lavori potrà accedere all'interfaccia utente per poter aggiungere un nuovo rigo sul giornale, per poter modificare un rigo già presente (si dovrà comunque tener traccia del rigo sovrascritto) e per poter terminare la compilazione di un giornale. In seguito ad ogni inserimento o modifica, dovranno essere immediatamente visibili sull'interfaccia utente, sia per il direttore dei lavori, ma anche per gli altri attori. Nel caso di terminazione della compilazione di un giornale, dovrà essere possibile poter compilare un nuovo giornale. Inoltre, i giornali passati dovranno essere comunque consultabili da qualsiasi utente. Non sarà però possibile poter compilare in alcun modo un giornale dei lavori passato, quindi né inserire un nuovo rigo, né modificarne uno.

5.2.1. Task Card 2

Story ID: 2	Task ID: 2.1
Task tag: Studio giornale dei Lavori	Software Engineer: Tutti
Descrizione: per poter capire i dati da dover inserire nel giornale dei lavori bisogna: <ul style="list-style-type: none">• Documentarsi su come è fatto un giornale dei lavori• Trovare dei modelli di giornale dei lavori	

Story ID: 2	Task ID: 2.2
Task tag: Schede CRC	Software Engineer: Davide, Iacopo
Descrizione:	
<ul style="list-style-type: none"> Preparazione delle schede CRC Operazioni di selezione e disambiguazione delle schede CRC 	

Story ID: 2	Task ID: 2.3
Task tag: Analisi orientata alle interfacce	Software Engineer: Davide, Iacopo
Descrizione:	
<ul style="list-style-type: none"> Selezione schede CRC degli utenti Controllo delle responsabilità che devono essere tramutate in interfacce 	

Story ID: 2	Task ID: 2.4
Task tag: Mockup e modelli	Software Engineer: Davide, Iacopo
Descrizione:	
<ul style="list-style-type: none"> Creazione mockup Creazione dei modelli di presentazione, navigazione e dialogo 	

Story ID: 2	Task ID: 2.5
Task tag: Analisi orientata alle classi	Software Engineer: Davide, Iacopo
Descrizione:	
<ul style="list-style-type: none"> Classificare schede CRC in classi e attributi Definire le relazioni tra classi e attributi Creazione diagramma delle classi 	

Story ID: 2	Task ID: 2.6
Task tag: Analisi orientata ai comportamenti	Software Engineer: Davide, Iacopo
Descrizione:	
<ul style="list-style-type: none"> • Creazione diagramma delle sequenze e di stato 	

Story ID: 2	Task ID: 2.7
Task tag: Implementazione smart contract per inserimento, modifica, visualizzazione rigo del giornale dei lavori	Software Engineer: Roberto, Paolo
Descrizione:	
<ul style="list-style-type: none"> • Creazione smart contract Giornale.sol • Implementazione struct in grado di contenere righi di giornali • Implementazioni funzione di inserimento di un rigo del giornale • Implementazione funzione di modifica (senza sovrascrittura) di un rigo del giornale • Implementazione funzione di visualizzazione di un giornale • Implementazione funzione di terminazione del giornale dei lavori corrente 	
Test:	
<ul style="list-style-type: none"> • Inserimento di dati corretti -> Inserimento del dato • Inserimento di dati errati (scambio tipi di dato) -> Messaggio di errore • Inserimento di dati parziali -> Messaggio di errore • Modifica di un dato -> Creazione nuovo rigo che sovrascriva il rigo modificato • Visualizzazione di un preciso giornale -> Corretta visualizzazione del giornale richiesto • Terminazione giornale corrente -> Passaggio alla compilazione del nuovo giornale 	

Story ID: 2	Task ID: 2.8
Task tag: Implementazione funzione di migrazione del contratto	Software Engineer: Roberto, Paolo
Descrizione:	
<ul style="list-style-type: none"> • Implementazione funzione di migrazione del contratto Giornale.sol 	
Test:	
<ul style="list-style-type: none"> • Migrazione Giornale.sol -> Controllo su Blockchain di avvenuta migrazione 	

Story ID: 2	Task ID: 2.9
Task tag: Implementazione lato frontend interfaccia per inserimenti modifiche e visualizzazione giornale	Software Engineer: Roberto, Paolo
Descrizione:	
<ul style="list-style-type: none"> • Visualizzazione giornale dei lavori • Form inserimento rigo giornale • Form modifica rigo giornale 	

Story ID: 2	Task ID: 2.10
Task tag: Implementazione lato server per l'inserimento, modifica, visualizzazione del rigo del giornale dei lavori	Software Engineer: Davide, Iacopo
Descrizione:	
<ul style="list-style-type: none"> • Creazione funzione di POST per richiamare la funzione contenuta nello smart contract per l'inserimento di un rigo di un giornale • Creazione funzione di POST per richiamare la funzione contenuta nello smart contract per la modifica di un rigo di un giornale • Creazione funzione di GET per richiamare la funzione contenuta nello smart contract per la visualizzazione di un giornale • Creazione funzione di POST per richiamare la funzione contenuta nello smart contract per la terminazione di un giornale 	
Test:	
<ul style="list-style-type: none"> • Chiamata corretta di una funzione -> chiamata alla corretta funzione dello smart contract • Chiamata non corretta di una funzione -> Messaggio di errore 	

5.3. Story Card 3

Release: 1	Story ID: 3
Story Tag: Inserimento e modifica libretto delle misure	
Descrizione: il libretto delle misure è il documento attraverso il quale si accerta la quantità di lavoro eseguita di tutte le lavorazioni. Viene redatto dal direttore dei lavori e viene firmato dalla ditta appaltatrice, eventualmente con riserva. Può essere consultato da tutti gli attori.	Dettaglio: solamente il direttore dei lavori potrà accedere all'interfaccia utente per l'inserimento o la modifica (anche in questo caso il rigo sovrascritto dovrà essere visibile) di un rigo del libretto delle misure. Al termine di ogni inserimento o modifica, questi dovranno essere immediatamente visibili ad ogni attore. Nel caso di terminazione della compilazione di un libretto delle misure, dovrà essere possibile poter compilare un nuovo rigo di un nuovo libretto delle misure, senza la possibilità di poter inserire un nuovo rigo o modificarne uno nel libretto che è stato terminato. Deve essere però possibile ad ogni utente poter consultare qualsiasi libretto passato.

5.3.1. Task Card 3

Story ID: 3	Task ID: 3.1
Task tag: Studio libretto delle misure	Software Engineer: Tutti
Descrizione: al fine di poter comprendere i dati che dovranno essere memorizzati in un libretto, è necessario: <ul style="list-style-type: none">• Documentarsi su come è fatto un libretto delle misure• Trovare dei modelli di libretto delle misure	

Story ID: 3	Task ID: 3.2
Task tag: Schede CRC	Software Engineer: Davide, Iacopo
Descrizione:	
<ul style="list-style-type: none"> Preparazione delle schede CRC Operazioni di selezione e disambiguazione delle schede CRC 	

Story ID: 3	Task ID: 3.3
Task tag: Analisi orientata alle interfacce	Software Engineer: Davide, Iacopo
Descrizione:	
<ul style="list-style-type: none"> Selezione schede CRC degli utenti Controllo delle responsabilità che devono essere tramutate in interfacce 	

Story ID: 3	Task ID: 3.4
Task tag: Mockup e modelli	Software Engineer: Davide, Iacopo
Descrizione:	
<ul style="list-style-type: none"> Creazione mockup Creazione dei modelli di presentazione, navigazione e dialogo 	

Story ID: 3	Task ID: 3.5
Task tag: Analisi orientata alle classi	Software Engineer: Davide, Iacopo
Descrizione:	
<ul style="list-style-type: none"> Classificare schede CRC in classi e attributi Definire le relazioni tra classi e attributi Creazione diagramma delle classi di analisi Realizzare il diagramma dei package di analisi 	

Story ID: 3	Task ID: 3.6
Task tag: Analisi orientata ai comportamenti	Software Engineer: Davide, Iacopo
Descrizione:	
<ul style="list-style-type: none"> • Creazione diagramma delle sequenze e di stato 	

Story ID: 3	Task ID: 3.7
Task tag: Implementazione smart contract per l'inserimento, modifica e visualizzazione dei righi del libretto delle misure	Software Engineer: Roberto, Paolo
Descrizione:	
<ul style="list-style-type: none"> • Creazione smart contract Libretto.sol • Implementazione struct in grado di contenere righi di libretti • Implementazione funzioni che permettano il collegamento di questo smart contract allo smart contract Misure.sol che contiene le lavorazioni da fare • Implementazione funzione di inserimento di un rigo del libretto (aggiungere controllo che il codice della lavorazione inserita in un rigo sia uno tra quelli già inseriti precedentemente dal RUP) • Implementazione funzione di modifica di un rigo del libretto • Implementazione funzione di visualizzazione di un libretto • Implementazione funzione per la terminazione del libretto corrente 	
Test:	
<ul style="list-style-type: none"> • Inserimento di dati corretti -> Inserimento del dato • Inserimento di un dato con un codice di lavorazione che non è stato precedentemente inserito dal RUP -> Messaggio di errore • Inserimento di dati errati (scambio tipi di dato) -> Messaggio di errore • Inserimento di dati parziali -> Messaggio di errore • Modifica di un dato -> Creazione nuovo rigo che sovrascriva il rigo modificato • Visualizzazione di un preciso libretto -> Corretta visualizzazione del libretto richiesto • Terminazione libretto corrente -> Passaggio alla compilazione del nuovo libretto 	

Story ID: 3	Task ID: 3.8
Task tag: Implementazione funzione migrazione contratto	Software Engineer: Roberto, Paolo
Descrizione:	
<ul style="list-style-type: none"> • Implementazione funzione di migrazione del contratto Libretto.sol 	
Test:	
<ul style="list-style-type: none"> • Migrazione Libretto.sol -> Controllo su Blockchain di avvenuta migrazione 	

Story ID: 3	Task ID: 3.9
Task tag: Implementazione lato frontend interfaccia per inserimenti modifiche e visualizzazione libretto	Software Engineer: Davide, Iacopo
Descrizione:	
<ul style="list-style-type: none"> • Visualizzazione libretto • Form inserimento rigo libretto • Form modifica rigo libretto 	

Story ID: 3	Task ID: 3.10
Task tag: Implementazione lato server dell'inserimento, modifica, visualizzazione dei righi di un libretto delle misure	Software Engineer: Roberto, Paolo
Descrizione:	
<ul style="list-style-type: none"> • Creazione della funzione di POST per richiamare la funzione per l'inserimento di un rigo del libretto • Creazione della funzione di POST per richiamare la funzione contenuta nello smart contract per la modifica di un rigo del libretto • Creazione della funzione di GET per richiamare la funzione contenuta nello smart contract per restituire un libretto • Creazione della funzione di POST per richiamare la funzione contenuta nello smart contract per terminare un libretto 	
Test:	
<ul style="list-style-type: none"> • Chiamata corretta di una funzione -> Chiamata alla corretta funzione dello smart contract • Chiamata non corretta di una funzione -> Messaggio di errore 	

5.4. Story Card 4

Release: 1	Story ID: 4
Story Tag: Firma o riserva al libretto delle misure	
Descrizione: ogni libretto delle misure dovrà essere firmato dalla ditta appaltatrice che potrà scegliere se firmare il libretto, ovvero attestare che il libretto è corretto così, oppure potrà firmarlo con riserva, in modo da riferire al direttore dei lavori che uno o più righi del libretto hanno una percentuale di completamento non corretta. Sarà poi compito del direttore dei lavori accertarsi sulla reale percentuale di completamento ed esplicitare la riserva (sciogliere la riserva come si dice in gergo).	Dettaglio: solamente quando un libretto sarà stato decretato come terminato da parte del direttore dei lavori, ovvero quando si sarà passati alla compilazione di un nuovo libretto, dall'interfaccia utente della ditta sarà possibile firmare/firmare con riserva ognuno dei righi riportati su quel libretto. Quando la ditta firma con riserva, sull'interfaccia del direttore dei lavori comparirà un alert in modo da notificargli la necessità di sciogliere la riserva espressa dalla ditta. Quindi dovrà esprimere una nuova percentuale di completamento del lavoro (o la stessa, nel caso era corretta la vecchia percentuale) ed inserire una descrizione della scelta. Dopo che il direttore dei lavori avrà apposto una riserva, la percentuale di completamento della lavorazione espressa in precedenza sul rigo del libretto, non dovrà essere sovrascritta dalla riserva espressa dal direttore, ma dovrà rimanere immutata. Sarà invece consultabile in altro modo la nuova percentuale espressa con la riserva.

5.4.1. Task Card 4

Story ID: 4	Task ID: 4.1
Task tag: Schede CRC	Software Engineer: Davide, Iacopo
Descrizione: <ul style="list-style-type: none">• Preparazione delle schede CRC• Operazioni di selezione e disambiguazione delle schede CRC	

Story ID: 4	Task ID: 4.2
Task tag: Analisi orientata alle interfacce	Software Engineer: Davide, Iacopo
Descrizione:	
<ul style="list-style-type: none"> • Selezione schede CRC degli utenti • Controllo delle responsabilità che devono essere tramutate in interfacce 	

Story ID: 4	Task ID: 4.3
Task tag: Mockup e modelli	Software Engineer: Davide, Iacopo
Descrizione:	
<ul style="list-style-type: none"> • Creazione mockup • Creazione dei modelli di presentazione, navigazione e dialogo 	

Story ID: 4	Task ID: 4.4
Task tag: Analisi orientata alle classi	Software Engineer: Davide, Iacopo
Descrizione:	
<ul style="list-style-type: none"> • Classificare schede CRC in classi e attributi • Definire le relazioni tra classi e attributi • Creazione diagramma delle classi di analisi • Realizzare il diagramma dei package di analisi 	

Story ID: 4	Task ID: 4.5
Task tag: Analisi orientata ai comportamenti	Software Engineer: Davide, Iacopo
Descrizione:	
<ul style="list-style-type: none"> • Creazione diagramma delle sequenze e di stato 	

Story ID: 4	Task ID: 4.6
Task tag: Implementazione smart contract	Software Engineer: Roberto, Paolo
Descrizione:	
<ul style="list-style-type: none"> • Creazione smart contract Firme.sol • Implementazione struct in grado di contenere firme • Implementazione funzione di inserimento della firma per un rigo del libretto • Implementazione funzione di inserimento della firma con riserva per un rigo del libretto • Implementazione funzione di inserimento di una riserva da parte del direttore dei lavori (controllo che sia stata posta una firma con riserva) • Implementazione funzione di visualizzazione delle firme • Implementazione funzione di visualizzazione delle riserve 	
Test:	
<ul style="list-style-type: none"> • Inserimento della firma -> Icona che indica la completezza del rigo • Inserimento della firma con riserva -> Icona che indica il rigo non completo • Inserimento della riserva -> Icona della riserva non completa che si modifica in un rigo completo 	

Story ID: 4	Task ID: 4.7
Task tag: Implementazione smart contract per notificare la presenza di firme/firme con riserve	Software Engineer: Roberto, Paolo
Descrizione:	
<ul style="list-style-type: none"> • Implementazione funzione di controllo dei righi da firmare • Implementazione funzione di controllo dei righi firmati con riserva 	
Test:	
<ul style="list-style-type: none"> • Libretto terminato -> Funzione restituisce tutti i righi • Libretto non terminato -> Messaggio di errore di non terminazione del libretto • Libretto terminato con alcune firme inserite -> Funzione restituisce solamente i righi in cui non è presente la firma 	

Story ID: 4	Task ID: 4.8
Task tag: Implementazione funzione di migrazione	Software Engineer: Roberto, Paolo
Descrizione:	
<ul style="list-style-type: none"> • Implementazione funzione di migrazione del contratto Firme.sol 	
Test:	
<ul style="list-style-type: none"> • Migrazione Misure.sol -> Controllo su Blockchain di avvenuta migrazione 	

Story ID: 4	Task ID: 4.9
Task tag: Implementazione lato server notifiche al direttore dei lavori	Software Engineer: Roberto, Paolo
Descrizione:	
<ul style="list-style-type: none"> • Implementazione funzione di GET per chiamare la funzione presente nello smart contract che controlla le firme da inserire • Implementazione funzione di GET per chiamare la funzione presente nello smart contract che controlla i righi firmati con riserva 	
Test:	
<ul style="list-style-type: none"> • Libretto terminato -> Notifica alla ditta di firmare i righi del libretto in questione • Inserimento firma con riserva ad un rigo -> Notifica indirizzata al DL 	

Story ID: 4	Task ID: 4.10
Task tag: Implementazione lato server dell'inserimento firme e riserve	Software Engineer: Davide, Iacopo
Descrizione:	
<ul style="list-style-type: none"> • Creazione della funzione di POST per richiamare la funzione per l'inserimento di una firma in un rigo del libretto • Creazione della funzione di POST per richiamare la funzione per l'inserimento di una firma con riserva in un rigo del libretto • Creazione della funzione di GET per richiamare la funzione per restituire le firme apposte • Creazione della funzione di POST per richiamare la funzione per l'inserimento di una riserva in un rigo del libretto con una firma con riserva • Creazione della funzione di GET per richiamare la funzione per restituire una riserva apposta in un rigo del libretto 	

5.5. Story Card 5

Release: 1	Story ID: 5
Story Tag: Abilitazione alla creazione del registro di contabilità	
Descrizione: <p>il registro delle contabilità riassume l'intera contabilizzazione dell'opera, dato che ad ogni quantità di lavorazioni eseguite e registrate nel libretto delle misure vengono applicati i corrispondenti prezzi di contratto in modo da determinare l'avanzamento dei lavori sotto forma del corrispettivo maturato dalla ditta appaltatrice.</p> <p>Viene compilato dal RUP. Può essere consultato da tutti gli attori.</p>	Dettaglio: <p>per poter creare un registro della contabilità si necessita innanzitutto che almeno un libretto delle misure sia completo, ovvero:</p> <ul style="list-style-type: none">• Il direttore dei lavori ha completato l'inserimento o le modifiche di righi in quel libretto e ha quindi decretato il libretto come completo• La ditta ha firmato ogni rigo con una normale firma o con una firma con riserva• Eventuali riserve siano state sciolte dal direttore dei lavori, ovvero il direttore dei lavori abbia esplicitato le riserve presentate dalla ditta <p>Quando un libretto si completa, dovrà essere inoltrata una notifica al RUP per entrarne a conoscenza. A questo punto, il RUP sarà abilitato alla creazione del registro di contabilità.</p> <p>Si avrà quindi un bottone di abilitazione che creerà in modo automatico il registro di contabilità relativo al libretto completato.</p>

5.5.1. Task Card 5

Story ID: 5	Task ID: 5.1
Task tag: Studio registro delle contabilità	Software Engineer: Tutti
Descrizione:	
al fine di poter comprendere i dati che dovranno essere memorizzati in un registro, è necessario: <ul style="list-style-type: none">• Documentarsi su come è fatto un registro delle contabilità• Trovare dei modelli di registri delle contabilità	

Story ID: 5	Task ID: 5.2
Task tag: Schede CRC	Software Engineer: Roberto, Paolo
Descrizione:	
<ul style="list-style-type: none">• Preparazione delle schede CRC• Operazioni di selezione e disambiguazione delle schede CRC	

Story ID: 5	Task ID: 5.3
Task tag: Analisi orientata alle interfacce	Software Engineer: Roberto, Paolo
Descrizione:	
<ul style="list-style-type: none">• Selezione schede CRC degli utenti• Controllo delle responsabilità che devono essere tramutate in interfacce	

Story ID: 5	Task ID: 5.4
Task tag: Mockup e modelli	Software Engineer: Roberto, Paolo
Descrizione:	
<ul style="list-style-type: none">• Creazione mockup• Creazione dei modelli di presentazione, navigazione e dialogo	

Story ID: 5	Task ID: 5.5
Task tag: Analisi orientata alle classi	Software Engineer: Roberto, Paolo
Descrizione:	
<ul style="list-style-type: none"> • Classificare schede CRC in classi e attributi • Definire le relazioni tra classi e attributi • Creazione diagramma delle classi di analisi • Realizzare il diagramma dei package di analisi 	

Story ID: 5	Task ID: 5.6
Task tag: Analisi orientata ai comportamenti	Software Engineer: Roberto, Paolo
Descrizione:	
<ul style="list-style-type: none"> • Creazione diagramma delle sequenze e di stato 	

Story ID: 5	Task ID: 5.7
Task tag: Implementazione smart contract per controlli sulla completezza di un libretto	Software Engineer: Davide, Iacopo
Descrizione:	
<ul style="list-style-type: none"> • Creazione smart contract Registro.sol • Implementazione funzioni per il controllo della firma su un rigo del libretto • Implementazione funzioni per il controllo della firma con riserva su un rigo del libretto • Implementazione funzioni per il controllo che una riserva presentata dalla ditta sia stata esplicitata dal direttore dei lavori 	
Test:	
<ul style="list-style-type: none"> • Firma di tutti i righi di un libretto -> Libretto completo • Firma con riserva di alcuni righi -> Libretto non completo • Riserve completamente esplicitate -> Libretto completo 	

Story ID: 5	Task ID: 5.8
Task tag: Implementazione funzione migrazione contratto	Software Engineer: Davide, Iacopo
Descrizione:	
<ul style="list-style-type: none"> • Implementazione funzione di migrazione del contratto Registro.sol 	
Test:	
<ul style="list-style-type: none"> • Migrazione Registro.sol -> Controllo su Blockchain di avvenuta migrazione 	

Story ID: 5	Task ID: 5.9
Task tag: Implementazione lato server funzioni di controllo	Software Engineer: Roberto, Paolo
Descrizione:	
<ul style="list-style-type: none"> • Implementazione funzione di PUT che richiami la funzione di controllo della firma in un rigo • Implementazione funzione di PUT che richiami la funzione di controllo della firma con riserva in un rigo • Implementazione funzione di PUT che richiami la funzione di controllo dell'inserimento di una riserva in un rigo 	

Story ID: 5	Task ID: 5.10
Task tag: Implementazione lato server notifiche al RUP	Software Engineer: Davide, Iacopo
Descrizione:	
<ul style="list-style-type: none"> • Implementazione funzione di GET che chiama la funzione dello smart contract per il controllo di avvenuta esplicitazione della riserva 	

Story ID: 5	Task ID: 5.11
Task tag: Implementazione lato frontend per la visualizzazione dei righi completi o meno	Software Engineer: Roberto, Paolo
Descrizione:	
in aggiunta all'interfaccia precedentemente creata per la visualizzazione di un libretto delle misure, devono essere previste delle icone per la visualizzazione se un rigo sia completo ovvero sia stato firmato dalla ditta, firmato con riserva e la riserva sia stata esplicitata dal direttore dei lavori (quindi considerabile come completo) oppure se firmato con riserva ma la riserva non sia ancora stata esplicitata	

Story ID: 5	Task ID: 5.12
Task tag: Implementazione smart contract per il calcolo dei valori monetari da inserire nel registro	Software Engineer: Davide, Iacopo
Descrizione:	
dato che il calcolo del valore maturato dalle lavorazioni viene inserito in modo automatico nel registro delle contabilità, a partire dalla percentuale di completamento delle lavorazioni, devono essere previste funzioni per il calcolo automatico del valore monetario.	
<ul style="list-style-type: none"> • Implementazione funzione che calcoli il costo complessivo dell'opera edilizia • Implementazione funzione che calcoli la suddivisione delle percentuali delle lavorazioni in base ai loro costi • Implementazione funzione che a partire dalla percentuale di completamento, calcoli il costo maturato 	

Story ID: 5	Task ID: 5.13
Task tag: Implementazione lato frontend per la visualizzazione dell'avanzamento dei lavori	Software Engineer: Roberto, Paolo
Descrizione:	
creazione interfaccia che permetta di visualizzare il totale rimanente (in percentuale), i costi totali, i costi maturati e l'andamento delle lavorazioni	

Story ID: 5	Task ID: 5.14
Task tag: Implementazione smart contract per l'attivazione del registro di contabilità	Software Engineer: Roberto, Paolo
Descrizione:	
<ul style="list-style-type: none"> • Implementazione funzione che metta in comunicazione lo smart contract Registro.sol allo smart contract Libretto.sol • Implementazione struct in grado di contenere righi di un registro • Implementazione funzione che abiliti l'attivazione del registro di contabilità (per ragioni di sicurezza deve richiamare all'interno le funzioni per il controllo che il libretto sia completo) • Implementazione funzione per l'inserimento di un rigo del registro • Implementazione funzione per la visualizzazione del registro 	

Story ID: 5	Task ID: 5.15
Task tag: Implementazione lato frontend per l'abilitazione all'attivazione del registro di contabilità	Software Engineer: Davide, Iacopo
Descrizione:	
creazione interfaccia che attivi le funzioni create per l'abilitazione del registro di contabilità	

Story ID: 5	Task ID: 5.16
Task tag: Implementazione lato server funzione di visualizzazione del registro di contabilità	Software Engineer: Roberto, Paolo
Descrizione:	
<ul style="list-style-type: none"> • Implementazione funzione di GET che richiami la funzione contenuta sullo smart contract per la visualizzazione di un registro 	

Story ID: 5	Task ID: 5.17
Task tag: Implementazione lato frontend per visualizzare il registro delle contabilità	Software Engineer: Davide, Iacopo
Descrizione:	
<ul style="list-style-type: none"> • Creazione interfaccia per la visualizzazione del registro delle contabilità 	

5.6. Story Card 6

Release: 1	Story ID: 6
Story Tag: Stato avanzamento dei lavori	
Descrizione: <p>Lo stato di avanzamento lavori serve per il pagamento delle rate di acconto e riassume tutte le lavorazioni e tutte le somministrazioni dall'inizio dell'appalto fino al momento della sua emissione.</p> <p>Viene redatto dal direttore dei lavori.</p> <p>Può essere visibile a tutti gli attori.</p>	Dettaglio: <p>per poter redigere uno stato di avanzamento dei lavori, si necessita innanzitutto che il costo maturato con l'avanzamento delle lavorazioni da parte della ditta appaltatrice risulti aver superato la rata di acconto fissata dal RUP. In questo modo si va ad accertare che la ditta ha completato delle lavorazioni per un certo valore. Questo valore, dopo la redazione del SAL, può essere quindi pagato alla ditta.</p> <p>Quando una rata di acconto viene superata, dovrà essere inviata una notifica al direttore che lo informi di questo superamento. Quindi sarà predisposto un bottone di avvio del SAL che andrà a generare in modo automatico il SAL corrente.</p>

5.6.1. Task Card 6

Story ID: 6	Task ID: 6.1
Task tag: Studio stato avanzamento lavori	Software Engineer: Tutti
Descrizione:	
al fine di poter comprendere i dati che dovranno essere memorizzati in un SAL, è necessario: <ul style="list-style-type: none">• Documentarsi su come è fatto un SAL• Trovare dei modelli di SAL	

Story ID: 6	Task ID: 6.2
Task tag: Schede CRC	Software Engineer: Davide, Iacopo
Descrizione:	
<ul style="list-style-type: none">• Preparazione delle schede CRC• Operazioni di selezione e disambiguazione delle schede CRC	

Story ID: 6	Task ID: 6.3
Task tag: Analisi orientata alle interfacce	Software Engineer: Davide, Iacopo
Descrizione:	
<ul style="list-style-type: none">• Selezione schede CRC degli utenti• Controllo delle responsabilità che devono essere tramutate in interfacce	

Story ID: 6	Task ID: 6.4
Task tag: Mockup e modelli	Software Engineer: Davide, Iacopo
Descrizione:	
<ul style="list-style-type: none">• Creazione mockup• Creazione dei modelli di presentazione, navigazione e dialogo	

Story ID: 6	Task ID: 6.5
Task tag: Analisi orientata alle classi	Software Engineer: Davide, Iacopo
Descrizione:	
<ul style="list-style-type: none"> • Classificare schede CRC in classi e attributi • Definire le relazioni tra classi e attributi • Creazione diagramma delle classi di analisi • Realizzare il diagramma dei package di analisi 	

Story ID: 6	Task ID: 6.6
Task tag: Analisi orientata ai comportamenti	Software Engineer: Davide, Iacopo
Descrizione:	
<ul style="list-style-type: none"> • Creazione diagramma delle sequenze e di stato 	

Story ID: 6	Task ID: 6.7
Task tag: Implementazione smart contract per controlli sul superamento di una rata di conto	Software Engineer: Roberto, Paolo
Descrizione:	
<ul style="list-style-type: none"> • Implementazione funzione per il calcolo della soglia da superare 	
Test:	
<ul style="list-style-type: none"> • Superamento di una rata -> Valore da pagare corrispondente • Non superamento di una rata -> Nessun valore da pagare • Superamento di più soglie successive -> Valore da pagare corrispondente all'ultima soglia superata 	

Story ID: 6	Task ID: 6.8
Task tag: Implementazione smart contract per controlli delle lavorazioni da inserire nel SAL	Software Engineer: Davide, Iacopo
Descrizione:	
<p>dato che un SAL contiene l'avanzamento dei lavori sin dal primo registro di contabilità, un SAL non corrisponde alle lavorazioni di un unico registro, ma a quelle di più registri. Quindi più registri possono contenere una stessa lavorazione, ovviamente con percentuali aggiornate (maggiori quindi) nel caso di un registro più recente. Per questo motivo si dovranno avere funzioni che andranno a controllare tra i vari righi dei registri in modo da prendere solamente le lavorazioni più aggiornate.</p> <ul style="list-style-type: none"> • Creazione contratto Avanzamento.sol • Implementazione funzione per far comunicare lo smart contract Avanzamento.sol allo smart contract Registro.sol • Implementazione funzione per inserire nel SAL tutte le lavorazioni che abbiano fatto superare la soglia 	
Test:	
<ul style="list-style-type: none"> • Diversi registri con lavorazioni tutte differenti -> Prende tutte le lavorazioni • Diversi registri con stesse lavorazioni -> Prende tutte le lavorazioni del registro più recente e dai registri meno recenti va a scartare quelle già prese 	

Story ID: 6	Task ID: 6.9
Task tag: Implementazione smart contract per l'abilitazione all'avvio del SAL	Software Engineer: Roberto, Paolo
Descrizione:	
<ul style="list-style-type: none"> • Implementazione funzione per l'abilitazione al SAL (deve richiamare all'interno le funzioni per il controllo del superamento della rata di acconto e per il controllo dei righi da inserire) • Implementazione funzione per l'inserimento di un rigo del SAL • Implementazione funzione per la visualizzazione del SAL 	

Story ID: 6	Task ID: 6.10
Task tag: Implementazione funzione migrazione contratto	Software Engineer: Roberto, Paolo
Descrizione:	
<ul style="list-style-type: none"> • Implementazione funzione di migrazione del contratto Avanzamento.sol 	
Test:	
<ul style="list-style-type: none"> • Migrazione Avanzamento.sol -> Controllo su Blockchain di avvenuta migrazione 	

Story ID: 6	Task ID: 6.11
Task tag: Implementazione lato server notifiche al direttore dei lavori	Software Engineer: Davide, Iacopo
Descrizione:	
<ul style="list-style-type: none"> • Implementazione funzione di GET che richiami la funzione dello smart contract che controlla l'avvenuto superamento di una soglia 	

Story ID: 6	Task ID: 6.12
Task tag: Implementazione lato server funzioni di visualizzazione del SAL	Software Engineer: Davide, Iacopo
Descrizione:	
<ul style="list-style-type: none"> • Implementazione funzione di GET che richiami la funzione contenuta nello smart contract per la visualizzazione di un SAL 	

Story ID: 6	Task ID: 6.13
Task tag: Implementazione lato frontend per la visualizzazione	Software Engineer: Roberto, Paolo
Descrizione:	
creazione interfaccia per la visualizzazione di un SAL	

5.7. Story Card 7

Release: 3	Story ID: 7
Story Tag: Inserimento certificato di pagamento	
Descrizione: il certificato di pagamento serve per certificare che una certa somma può essere pagata da parte della committenza verso la ditta appaltatrice. Questa certificazione viene posta dal RUP, che in seguito alla visura di un certo SAL e quindi dopo che è avvenuto un superamento di una rata di acconto, può emanare questo documento che verrà poi archiviato dal direttore dei lavori all'interno di un registro di contabilità. È quindi emanato dal RUP, ma è consultabile da tutti gli attori.	Dettaglio: ogni SAL redatto deve essere accompagnato da un certificato di pagamento. Questo viene redatto dal RUP, che va a certificare una somma da pagare alla ditta. In seguito alla compilazione del SAL, arriverà una notifica al RUP che gli permetterà di attivare la compilazione del certificato di pagamento. Il tutto viene fatto tramite un bottone che andrà in automatico ad emanare il certificato. Fatto ciò arriverà una notifica al direttore dei lavori, che potrà andare a questo punto ad inserire il certificato di pagamento all'interno del registro di contabilità. Ogni certificato di pagamento sarà poi tenuto conto nei SAL successivi in modo da detrarre dall'importo maturato, in quanto già certificato di essere pagato.

5.7.1. Task Card 7

Story ID: 7	Task ID: 7.1
Task tag: Studio certificato di pagamento	Software Engineer: Tutti
Descrizione: al fine di poter comprendere i dati che dovranno essere memorizzati in un certificato, è necessario: <ul style="list-style-type: none">• Documentarsi su come è fatto un certificato di pagamento• Trovare dei modelli di certificato di pagamento	

Story ID: 7	Task ID: 7.2
Task tag: Schede CRC	Software Engineer: Roberto, Paolo
Descrizione:	
<ul style="list-style-type: none"> Preparazione delle schede CRC Operazioni di selezione e disambiguazione delle schede CRC 	

Story ID: 7	Task ID: 7.3
Task tag: Analisi orientata alle interfacce	Software Engineer: Roberto, Paolo
Descrizione:	
<ul style="list-style-type: none"> Selezione schede CRC degli utenti Controllo delle responsabilità che devono essere tramutate in interfacce 	

Story ID: 7	Task ID: 7.4
Task tag: Mockup e modelli	Software Engineer: Roberto, Paolo
Descrizione:	
<ul style="list-style-type: none"> Creazione mockup Creazione dei modelli di presentazione, navigazione e dialogo 	

Story ID: 7	Task ID: 7.5
Task tag: Analisi orientata alle classi	Software Engineer: Roberto, Paolo
Descrizione:	
<ul style="list-style-type: none"> Classificare schede CRC in classi e attributi Definire le relazioni tra classi e attributi Creazione diagramma delle classi di analisi Realizzare il diagramma dei package di analisi 	

Story ID: 7	Task ID: 7.6
Task tag: Analisi orientata ai comportamenti	Software Engineer: Roberto, Paolo
Descrizione:	
<ul style="list-style-type: none"> • Creazione diagramma delle sequenze e di stato 	

Story ID: 7	Task ID: 7.7
Task tag: Implementazione lato server notifiche al RUP	Software Engineer: Davide, Iacopo
Descrizione:	
<ul style="list-style-type: none"> • Implementazione funzione di GET che richiami la funzione dello smart contract per l'avvenuta emissione di un certo SAL 	

Story ID: 7	Task ID: 7.8
Task tag: Implementazione smart contract per inserimento di un certificato di pagamento da parte del RUP	Software Engineer: Davide, Iacopo
Descrizione:	
<ul style="list-style-type: none"> • Implementazione funzione per l'inserimento di un certificato di pagamento • Implementazione funzione per la visualizzazione dei certificati di pagamento 	
Test:	
<ul style="list-style-type: none"> • Inserimento certificato -> Visualizzazione certificato 	

Story ID: 7	Task ID: 7.9
Task tag: Implementazione lato server notifiche al direttore dei lavori	Software Engineer: Roberto, Paolo
Descrizione:	
attraverso il database creato si andranno a memorizzare le notifiche da indirizzare al direttore dei lavori riguardanti l'inserimento di un certificato di pagamento	

Story ID: 7	Task ID: 7.10
Task tag: Implementazione smart contract per inserimento di un certificato di pagamento in un registro di contabilità da parte del direttore dei lavori	Software Engineer: Davide, Iacopo
Descrizione:	
<ul style="list-style-type: none"> • Implementazione funzione per l'inserimento di un certificato di pagamento • Implementazione funzione per la visualizzazione dei certificati di pagamento 	
Test:	
<ul style="list-style-type: none"> • Inserimento certificato nel registro -> Visualizzazione del certificato nel registro corrente 	

Story ID: 7	Task ID: 7.11
Task tag: Implementazione smart contract per detrazione dei certificati di pagamento dal registro di contabilità	Software Engineer: Davide, Iacopo
Descrizione:	
<p>dato che il registro andrà a contenere l'importo maturato dall'inizio delle lavorazioni, si dovranno andare a detrarre i certificati di pagamento già emessi, ovvero l'importo già certificato da essere pagato.</p> <ul style="list-style-type: none"> • Implementazione funzione per calcolare la somma da detrarre • Implementazione funzione per la detrazione 	
Test:	
<ul style="list-style-type: none"> • Somma di più certificati -> Somma corretta • Detrazione della somma -> Detrazione corretta 	

Story ID: 7	Task ID: 7.12
Task tag: Implementazione lato server funzioni di visualizzazione dei certificati di pagamento	Software Engineer: Roberto, Paolo
Descrizione:	
<ul style="list-style-type: none"> • Implementazione funzione di GET che richiami la funzione contenuta nello smart contract per la visualizzazione dei certificati di pagamento 	

Story ID: 7	Task ID: 7.13
Task tag: Implementazione lato frontend per la visualizzazione	Software Engineer: Roberto, Paolo
Descrizione: creazione interfaccia per la visualizzazione dei certificati di pagamento	

5.8. Story Card 8

Release: 1	Story ID: 8
Story Tag: Autenticazione al primo login	
Descrizione: in seguito al primo login attraverso nome utente e password sulla nostra web app, l'utente che sta accedendo deve essere autenticato e indirizzato alla pagina che gli compete.	Dettaglio: in base alla tipologia di utente, dovranno essere indirizzati a diverse funzionalità. In particolare, in seguito ad un primo login, il RUP dovrà essere indirizzato alla pagina di inserimento delle lavorazioni. Il direttore dei lavori e la ditta appaltatrice saranno invece indirizzati ad una pagina che richiede all'utente di attendere che il RUP abbia completato la prima fase di inserimento delle lavorazioni e della rata di acconto.

5.8.1. Task Card 8

Story ID: 8	Task ID: 8.1
Task tag: Studio JWT	Software Engineer: Tutti
Descrizione:	
<ul style="list-style-type: none">• Studio di JWT per l'autenticazione delle sessioni	

Story ID: 8	Task ID: 8.2
Task tag: Schede CRC	Software Engineer: Tutti
Descrizione:	
<ul style="list-style-type: none">• Preparazione delle schede CRC• Operazioni di selezione e disambiguazione delle schede CRC	

Story ID: 8	Task ID: 8.3
Task tag: Analisi orientata alle interfacce	Software Engineer: Davide, Iacopo
Descrizione:	
<ul style="list-style-type: none">• Selezione schede CRC degli utenti• Controllo delle responsabilità che devono essere tramutate in interfacce	

Story ID: 8	Task ID: 8.4
Task tag: Mockup e modelli	Software Engineer: Davide, Iacopo
Descrizione:	
<ul style="list-style-type: none">• Creazione mockup• Creazione dei modelli di presentazione, navigazione e dialogo	

Story ID: 8	Task ID: 8.5
Task tag: Analisi orientata alle classi	Software Engineer: Davide, Iacopo
Descrizione:	
<ul style="list-style-type: none"> • Classificare schede CRC in classi e attributi • Definire le relazioni tra classi e attributi • Creazione diagramma delle classi di analisi • Realizzare il diagramma dei package di analisi 	

Story ID: 8	Task ID: 8.6
Task tag: Analisi orientata ai comportamenti	Software Engineer: Davide, Iacopo
Descrizione:	
<ul style="list-style-type: none"> • Creazione diagramma delle sequenze e di stato 	

Story ID: 8	Task ID: 8.7
Task tag: Implementazione DBMS	Software Engineer: Roberto, Paolo
Descrizione:	
<p>in modo da poter contenere dati riguardanti i tre attori, risulta essere necessario un database</p> <ul style="list-style-type: none"> • Studio miglior database nel nostro caso • Creazione database • Creazione tabella che possa contenere nome utente, password e dati necessari dell'attore 	

Story ID: 8	Task ID: 8.8
Task tag: Implementazione meccanismo JWT	Software Engineer: Roberto, Paolo
Descrizione:	
<p>ad ogni nuova sessione, verrà richiesto dall'applicazione un token necessario per l'autenticazione. Verrà fatto un controllo sulla presenza o meno del token e verrà fatto un controllo sul token scaduto o meno</p> <ul style="list-style-type: none"> • Creazione token necessario all'autenticazione 	

Story ID: 8	Task ID: 8.9
Task tag: Verifica credenziali	Software Engineer: Davide, Iacopo
Descrizione:	
L'utente deve inserire le proprie credenziali nella pagina di login per poter effettuare l'accesso all'applicazione. Le credenziali inserite potranno essere salvate, secondo volontà dell'utente in modo da facilitare i futuri login. Le credenziali vengono prese dal backend e vengono verificate tramite le credenziali presenti sul DB. In seguito alla verifica, in base alle credenziali e quindi all'utente che si autentica, saranno indirizzati a diverse interfacce.	
Test:	
<ul style="list-style-type: none"> Inserimento credenziali corrette da parte del RUP -> Pagina contenente la form per l'inserimento di lavorazioni Inserimento credenziali corrette da parte del direttore dei lavori o dalla ditta -> Pagina di attesa che il RUP completi l'inserimento di lavorazioni e rata di acconto Inserimento credenziali errate -> Messaggio di errore di login Accesso da parte di un utente che abbia salvato le proprie credenziali -> Credenziali inserite automaticamente nella form di login 	

Story ID: 8	Task ID: 8.10
Task tag: Implementazione interfaccia di login	Software Engineer: Roberto, Paolo
Descrizione:	
<ul style="list-style-type: none"> Form inserimento credenziali 	

5.9. Story Card 9

Release: 2	Story ID: 9
Story Tag: Visualizzare lavorazioni, valore rata di acconto, rata acconto corrente...	
Descrizione: è possibile consultare tutte le lavorazioni (con la percentuale di completamento attuale) inserite all'interno della blockchain con la form di inserimento predisposta al RUP, la rata di acconto che è stata precedentemente fissata, la rata di acconto corrente, il costo totale dei lavori, il costo totale maturato e quindi il costo totale rimanente, la percentuale maturate e la percentuale rimanente.	Dettaglio: qualsiasi utente, a partire dalla home page del sito, potrà entrare nell'area riservata alla consultazione delle lavorazioni e dell'andamento dei lavori in generale.

5.9.1. Task Card 9

Story ID: 9	Task ID: 9.1
Task tag: Schede CRC	Software Engineer: Roberto, Paolo
Descrizione: <ul style="list-style-type: none">• Preparazione delle schede CRC• Operazioni di selezione e disambiguazione delle schede CRC	

Story ID: 9	Task ID: 9.2
Task tag: Analisi orientata alle interfacce	Software Engineer: Roberto, Paolo
Descrizione: <ul style="list-style-type: none">• Selezione schede CRC degli utenti• Controllo delle responsabilità che devono essere tramutate in interfacce	

Story ID: 9	Task ID: 9.3
Task tag: Mockup e modelli	Software Engineer: Roberto, Paolo
Descrizione:	
<ul style="list-style-type: none"> • Creazione mockup • Creazione dei modelli di presentazione, navigazione e dialogo 	

Story ID: 9	Task ID: 9.4
Task tag: Analisi orientata alle classi	Software Engineer: Roberto, Paolo
Descrizione:	
<ul style="list-style-type: none"> • Classificare schede CRC in classi e attributi • Definire le relazioni tra classi e attributi • Creazione diagramma delle classi di analisi • Realizzare il diagramma dei package di analisi 	

Story ID: 9	Task ID: 9.5
Task tag: Analisi orientata ai comportamenti	Software Engineer: Roberto, Paolo
Descrizione:	
<ul style="list-style-type: none"> • Creazione diagramma delle sequenze e di stato 	

Story ID: 9	Task ID: 9.6
Task tag: Implementazione smart contract per la variazione delle percentuali di completamento di una lavorazione	Software Engineer: Davide, Iacopo
Descrizione:	
<ul style="list-style-type: none"> • Implementazione funzione per variare la percentuale di completamento di una lavorazione (deve essere inserita all'interno della funzione che certifica il completamento di un libretto, in questo modo il tutto verrà fatto in modo automatico) • Implementazione funzione per variare la percentuale complessiva dell'opera da completare • Implementazione funzione per il calcolo della percentuale complessiva dell'opera che è stata completata 	
Test:	
<ul style="list-style-type: none"> • Inserimento di percentuali di completamento -> Corretto calcolo della percentuale completata e della rimanente 	

Story ID: 9	Task ID: 9.7
Task tag: Implementazione lato frontend per la visualizzazione	Software Engineer: Roberto, Paolo
Descrizione:	
creazione interfaccia per la visualizzazione delle informazioni necessarie quali lavorazioni con percentuali di completamento, valore della rata di acconto, rata di acconto attuale...	

Story ID: 9	Task ID: 9.8
Task tag: Implementazione lato backend per la visualizzazione	Software Engineer: Roberto, Paolo
Descrizione:	
<ul style="list-style-type: none"> • Implementazione funzione di GET per richiamare la funzione contenuta nello smart contract per la visualizzazione delle lavorazioni • Implementazione funzione di GET per richiamare la funzione contenuta nello smart contract per la visualizzazione della rata di acconto • Implementazione funzione di GET per richiamare la funzione contenuta nello smart contract per la visualizzazione del costo totale di progetto • Implementazione funzione di GET per richiamare la funzione contenuta nello smart contract per la visualizzazione del costo maturato • Implementazione funzione di GET per richiamare la funzione contenuta nello smart contract per la visualizzazione della soglia attuale 	

5.10. Story Card 10

Release: 2	Story ID: 10
Story Tag: Visualizzare grafico andamento lavori	
Descrizione: all'interno della home page del sito sarà presente un grafico che mostra l'andamento delle lavorazioni in base al tempo (asse delle ascisse) e al valore maturato (asse delle ordinate) rispetto ad ogni stato di avanzamento delle lavorazioni. Quindi ad ogni SAL si avrà un certo valore.	Dettaglio: dal grafico deve essere possibile cliccare sui valori per poter avere informazioni riguardanti il SAL che ha generato quel valore, come il numero del SAL, la data, il valore monetario calcolato.

5.10.1. Task Card 10

Story ID: 10	Task ID: 10.1
Task tag: Schede CRC	Software Engineer: Davide, Iacopo
Descrizione:	
<ul style="list-style-type: none">• Preparazione delle schede CRC• Operazioni di selezione e disambiguazione delle schede CRC	

Story ID: 10	Task ID: 10.2
Task tag: Analisi orientata alle interfacce	Software Engineer: Davide, Iacopo
Descrizione:	
<ul style="list-style-type: none">• Selezione schede CRC degli utenti• Controllo delle responsabilità che devono essere tramutate in interfacce	

Story ID: 10	Task ID: 10.3
Task tag: Mockup e modelli	Software Engineer: Davide, Iacopo
Descrizione:	
<ul style="list-style-type: none">• Creazione mockup• Creazione dei modelli di presentazione, navigazione e dialogo	

Story ID: 10	Task ID: 10.4
Task tag: Analisi orientata alle classi	Software Engineer: Davide, Iacopo
Descrizione:	
<ul style="list-style-type: none"> • Classificare schede CRC in classi e attributi • Definire le relazioni tra classi e attributi • Creazione diagramma delle classi di analisi • Realizzare il diagramma dei package di analisi 	

Story ID: 10	Task ID: 10.5
Task tag: Analisi orientata ai comportamenti	Software Engineer: Davide, Iacopo
Descrizione:	
<ul style="list-style-type: none"> • Creazione diagramma delle sequenze e di stato 	

Story ID: 10	Task ID: 10.6
Task tag: Implementazione smart contract per visualizzazione grafico	Software Engineer: Roberto, Paolo
Descrizione:	
<ul style="list-style-type: none"> • Implementazione funzione per restituire i dati necessari al grafico, ovvero la data del SAL e il valore monetario calcolato con il SAL 	

Story ID: 10	Task ID: 10.7
Task tag: Implementazione lato server per visualizzazione grafico	Software Engineer: Roberto, Paolo
Descrizione:	
<ul style="list-style-type: none"> • Implementazione funzione di GET per richiamare la funzione per la restituzione dei dati necessari alla visualizzazione 	

Story ID: 10	Task ID: 10.8
Task tag: Implementazione lato frontend per la visualizzazione grafico	Software Engineer: Davide, Iacopo
Descrizione: creazione grafico che contenga le date del SAL e il valore monetario calcolato con il SAL	

6. Analisi orientata alle interfacce

Durante questa fase si sono analizzate le story card prodotte nella fase precedente in modo da poter progettare le interfacce necessarie agli utenti. Una volta definite le interfacce andremo ad analizzare le strutture dati che dovrà utilizzare il software.

6.1. Schede CRC

6.1.1. Story Card 1

Classe: Web app	
Descrizione: Applicazione web che stiamo realizzando	
Responsabilità: <ul style="list-style-type: none">• Visualizzare form di inserimento lavorazioni• Visualizzare schermata con la lavorazione inserita• Visualizzare schermata con errore di non inserimento della lavorazione• Visualizzare form di modifica con dati preinseriti• Visualizzare schermata con la modifica inserita• Visualizzare schermata con errore di non avvenuta modifica• Visualizzare form di inserimento rata di acconto• Visualizzazione inserimento rata di acconto avvenuto con successo• Visualizzazione errore di inserimento della rata di acconto• Visualizzazione form di modifica della rata di acconto• Visualizzare home page	Collaboratori: <ul style="list-style-type: none">• Server• Blockchain• Dati lavorazione• Valore rata di acconto

Classe: Utente - RUP	
Descrizione: Responsabile unico del procedimento	
Responsabilità: <ul style="list-style-type: none"> • Inserire lavorazione • Decidere di modificare lavorazione • Modificare lavorazione • Visualizzare lavorazione • Terminare inserimento lavorazioni • Inserire rata di acconto • Decidere di modificare la rata di acconto • Modificare rata di acconto • Visualizzare rata di acconto • Visualizzare resoconto lavorazioni-rata di acconto • Terminare inserimenti 	Collaboratori: <ul style="list-style-type: none"> • Web app • Server • Blockchain • Dati lavorazione • Valore rata di acconto

Classe: Lavorazione	
Descrizione: Indica una lavorazione che sarà parte del progetto edilizio	
Responsabilità:	Collaboratori: <ul style="list-style-type: none"> • Utente – RUP • Server • Blockchain

Classe: Rata di acconto	
Descrizione: Indica una soglia monetaria da superare	
Responsabilità:	Collaboratori: <ul style="list-style-type: none"> • Utente – RUP • Server • Blockchain

Classe: Server	
Descrizione: Si occupa della gestione della comunicazione tra web app e blockchain	
Responsabilità: <ul style="list-style-type: none"> • Richiamare funzione di inserimento Lavorazione • Richiamare funzione di modifica lavorazione • Richiamare funzione di visualizzazione lavorazione • Funzione di terminazione inserimento lavorazioni • Richiamare funzione di inserimento rata di acconto • Richiamare funzione di modifica rata di acconto • Richiamare funzione di visualizzazione rata di acconto • Funzione di terminazione di inserimenti 	Collaboratori: <ul style="list-style-type: none"> • Web app • Blockchain

Classe: Blockchain	
Descrizione: Indica gli smart contract necessari per salvare dati all'interno della blockchain	
Responsabilità: <ul style="list-style-type: none"> • Inserire lavorazione • Modificare lavorazione • Visualizzare lavorazione • Inserire rata di acconto • Modificare rata di acconto • Visualizzare rata di acconto • Visualizzare resoconto lavorazioni-rata di acconto 	Collaboratori: <ul style="list-style-type: none"> • Server

6.1.2. Story Card 2

Flusso 1: per direttore dei lavori

- L'utente è loggato all'interno del sito
- L'applicazione si trova sulla home page del sito
- L'utente può premere sull'icona dedicata al giornale dei lavori ed entrare nella pagina dedicata al giornale
- L'applicazione visualizza il giornale dei lavori più recente
- L'utente può inserire, modificare un rigo e terminare (nel caso in cui il giornale sia stato già compilato) la compilazione

Classe: Utente - DL	
Descrizione: Direttore dei lavori	
Responsabilità: <ul style="list-style-type: none">• Accedere giornale dei lavori corrente• Decidere di inserire rigo• Inserire rigo• Decidere di modificare rigo• Modificare rigo• Visualizzare giornale• Decidere di terminare giornale• Terminare giornale	Collaboratori: <ul style="list-style-type: none">• Web app• Server• Blockchain

Classe: Web app	
Descrizione: Applicazione web che stiamo realizzando	
Responsabilità: <ul style="list-style-type: none"> • Visualizza home page • Visualizza schermata giornale non compilato e non terminato • Visualizza giornale compilato ma non terminato • Visualizza giornale compilato e terminato • Visualizza inserimento rigo avvenuto con successo • Visualizza inserimento rigo non avvenuto con successo • Visualizza form di modifica rigo • Visualizza modifica rigo avvenuta con successo • Visualizza modifica rigo non avvenuta con successo 	Collaboratori: <ul style="list-style-type: none"> • Server • Blockchain • Utente - DL

Classe: Server	
Descrizione: Si occupa della gestione della comunicazione tra web app e blockchain	
Responsabilità: <ul style="list-style-type: none"> • Richiamare funzione di inserimento rigo • Richiamare funzione di modifica rigo • Richiamare funzione di visualizzazione giornale • Richiamare funzione di terminazione inserimento giornale • Funzione di decisione di modifica di un rigo • Funzione di decisione di terminazione di un rigo 	Collaboratori: <ul style="list-style-type: none"> • Web app • Blockchain

Classe: Blockchain	
Descrizione: Indica gli smart contract necessari per salvare dati all'interno della blockchain	
Responsabilità: <ul style="list-style-type: none"> • Inserire rigo • Modificare rigo • Visualizzare giornale • Terminare giornale 	Collaboratori: <ul style="list-style-type: none"> • Server

Classe: Giornale dei lavori	
Descrizione: Documento di gestione dell'appalto	
Responsabilità:	Collaboratori: <ul style="list-style-type: none"> • Utente – DL • Server • Blockchain

Classe: Rigo giornale	
Descrizione: Indica un rigo del giornale dei lavori	
Responsabilità:	Collaboratori: <ul style="list-style-type: none"> • Utente – DL • Server • Blockchain

Flusso 2: per utente che non sia il direttore dei lavori

- L'utente è loggato al sito
- L'applicazione si trova sulla home page
- L'utente può premere sull'icona dedicata al giornale dei lavori ed entrare nella pagina dedicata al giornale
- L'applicazione visualizza il giornale dei lavori più recente

Classe: Utente	
Descrizione: Riguarda un utente che non sia il direttore dei lavori	
Responsabilità: <ul style="list-style-type: none">• Accedere giornale dei lavori corrente• Visualizzare giornale	Collaboratori: <ul style="list-style-type: none">• Web app• Server• Blockchain

Classe: Web app	
Descrizione: Applicazione web che stiamo realizzando	
Responsabilità: <ul style="list-style-type: none">• Visualizza home page• Visualizza schermata giornale non compilato e non terminato• Visualizza giornale compilato ma non terminato• Visualizza giornale compilato e terminato	Collaboratori: <ul style="list-style-type: none">• Server• Blockchain• Utente

Classe: Server	
Descrizione: Si occupa della gestione della comunicazione tra web app e blockchain	
Responsabilità: <ul style="list-style-type: none">• Richiamare funzione di visualizzazione giornale	Collaboratori: <ul style="list-style-type: none">• Web app• Blockchain

Classe: Blockchain	
Descrizione: Indica gli smart contract necessari per salvare dati all'interno della blockchain	
Responsabilità: <ul style="list-style-type: none"> • Visualizzare giornale 	Collaboratori: <ul style="list-style-type: none"> • Server

6.1.3. Story Card 3

Classe: Utente - DL	
Descrizione: Utente dell'applicazione	
Responsabilità: <ul style="list-style-type: none"> • Accedere libretto delle misure corrente • Decidere di inserire nuovo rigo • Inserimento rigo • Decidere di modificare un rigo • Modificare un rigo • Decidere di terminare un libretto • Terminazione libretto • Decidere di inserire allegato • Conferma inserire allegato • Decidere di ricercare allegato nel file system • Inserire allegato ricercato nel file system 	Collaboratori: <ul style="list-style-type: none"> • Web app • Server • Blockchain

Classe: Server	
Descrizione: Si occupa della gestione della comunicazione tra web app e blockchain	
Responsabilità: <ul style="list-style-type: none"> • Richiamare funzione di inserimento rigo • Richiamare funzione di modifica rigo • Richiamare funzione di visualizzazione libretto • Richiamare funzione di terminazione inserimento libretto • Funzione di decisione di modifica di un rigo • Funzione di decisione di terminazione di un rigo • Funzione di decisione inserimento allegato • Funzione di inserimento allegato • Funzione di conferma di inserimento allegato 	Collaboratori: <ul style="list-style-type: none"> • Web app • Blockchain

Classe: Web app	
Descrizione: Applicazione web che stiamo realizzando	
Responsabilità: <ul style="list-style-type: none"> • Visualizza home page • Visualizza libretto non compilato e non terminato • Visualizza libretto compilato ma non terminato con possibilità di modifica rigo • Visualizza libretto compilato e terminato con possibilità di inserimento firme o firme con riserve • Visualizza libretto compilato ma non terminato • Visualizza libretto compilato e terminato • Visualizzazione form di inserimento rigo • Visualizzazione schermata di inserimento rigo avvenuto con successo • Visualizzazione schermata di inserimento rigo non avvenuto con successo • Visualizzazione form di modifica di un rigo 	Collaboratori: <ul style="list-style-type: none"> • Server • Blockchain • Utente -DL

<ul style="list-style-type: none"> • Visualizzazione schermata di avvenuta modifica di un rigo • Visualizzazione schermata con errore di non avvenuta modifica di un rigo • Visualizzazione alert box di conferma terminazione libretto • Visualizzazione libretto terminato • Visualizzazione form di inserimento allegato • Visualizzazione schermata rappresentativa del file system • Visualizzazione schermata di caricamento dell'allegato • Visualizzazione form di inserimento rigo con l'allegato inserito 	
---	--

Classe: Blockchain	
Descrizione: Indica gli smart contract necessari per salvare dati all'interno della blockchain	
Responsabilità: <ul style="list-style-type: none"> • Inserire rigo • Modificare rigo • Visualizzare libretto • Terminare libretto 	Collaboratori: <ul style="list-style-type: none"> • Server

Classe: Libretto delle misure	
Descrizione: Documento di gestione dell'appalto	
Responsabilità:	Collaboratori: <ul style="list-style-type: none"> • Utente – DL • Server • Blockchain

Classe: Rigo libretto	
Descrizione: Indica un rigo del libretto delle misure	
Responsabilità:	<p>Collaboratori:</p> <ul style="list-style-type: none"> • Utente – DL • Server • Blockchain

6.1.4. Story Card 4

Classe: Utente - DL	
Descrizione: Utente dell'applicazione	
Responsabilità: <ul style="list-style-type: none"> • Accedere al libretto terminato • Decidere di inserire riserva al rigo con una firma con riserva • Inserire riserva al rigo con una firma con riserva 	<p>Collaboratori:</p> <ul style="list-style-type: none"> • Web app • Server • Blockchain

Classe: Utente - RUP	
Descrizione: Utente dell'applicazione	
Responsabilità: <ul style="list-style-type: none"> • Accedere al libretto terminato 	<p>Collaboratori:</p> <ul style="list-style-type: none"> • Web app • Server • Blockchain

Classe: Utente - Ditta	
Descrizione: Utente dell'applicazione	
Responsabilità: <ul style="list-style-type: none"> • Accedere al libretto terminato • Decidere di inserire firma al rigo • Inserire firma al rigo • Decidere di inserire firma con riserva al rigo • Inserire firma con riserva al rigo 	Collaboratori: <ul style="list-style-type: none"> • Web app • Server • Blockchain

Classe: Web app	
Descrizione: Applicazione web che stiamo realizzando	
Responsabilità: <ul style="list-style-type: none"> • Visualizza libretto terminato • Visualizza libretto terminato con possibilità di firma/firma con riserva • Visualizza libretto terminato con possibilità di esplicitare riserve • Visualizza schermata di firma avvenuta con successo • Visualizzazione schermata di firma con riserva avvenuta con successo • Visualizza form di inserimento riserva • Visualizza schermata di inserimento riserva avvenuta con successo • Visualizza schermata di inserimento riserva non avvenuta con successo 	Collaboratori: <ul style="list-style-type: none"> • Server • Blockchain • Utente - DL • Utente – Ditta • Utente - RUP

Classe: Blockchain	
Descrizione: Indica gli smart contract necessari per salvare dati all'interno della blockchain	
Responsabilità: <ul style="list-style-type: none"> • Inserire firma • Inserire firma con riserva • Inserire riserva 	Collaboratori: <ul style="list-style-type: none"> • Server

Classe: Server	
Descrizione: Si occupa della gestione della comunicazione tra web app e blockchain	
Responsabilità: <ul style="list-style-type: none"> • Richiamare funzione di inserimento firma • Richiamare funzione di inserimento firma con riserva • Richiamare funzione di inserimento riserva 	Collaboratori: <ul style="list-style-type: none"> • Web app • Blockchain

Classe: Firma	
Descrizione: Indica la firma posta dalla ditta	
Responsabilità:	Collaboratori: <ul style="list-style-type: none"> • Utente – Ditta • Server

Classe: Riserva	
Descrizione: Indica la riserva esplicitata dal DL	
Responsabilità:	Collaboratori: <ul style="list-style-type: none"> • Utente – DL • Server

6.1.5. Story Card 5

Classe: Utente - RUP	
Descrizione: Utente dell'applicazione	
Responsabilità: <ul style="list-style-type: none">• Accedere area registro di contabilità• Decidere di compilare il registro	Collaboratori: <ul style="list-style-type: none">• Web app• Server• Blockchain

Classe: Web app	
Descrizione: Applicazione web che stiamo realizzando	
Responsabilità: <ul style="list-style-type: none">• Visualizzare home page con notifica sul registro della contabilità• Visualizzare schermata del registro della contabilità non compilato ma compilabile• Visualizzare schermata del registro della contabilità non compilato ma non compilabile• Visualizzare schermata del registro compilato	Collaboratori: <ul style="list-style-type: none">• Server• Blockchain• Utente – RUP

Classe: Server	
Descrizione: Si occupa della gestione della comunicazione tra web app e blockchain	
Responsabilità: <ul style="list-style-type: none">• Richiamare funzione di compilazione del registro	Collaboratori: <ul style="list-style-type: none">• Web app• Blockchain

Classe: Blockchain	
Descrizione: Indica gli smart contract necessari per salvare dati all'interno della blockchain	
Responsabilità: <ul style="list-style-type: none"> • Compilazione registro della contabilità • Inserire rigo registro 	Collaboratori: <ul style="list-style-type: none"> • Server

6.1.6. Story Card 6

Classe: Utente - DL	
Descrizione: Utente dell'applicazione	
Responsabilità: <ul style="list-style-type: none"> • Accedere area registro di contabilità • Decidere di compilare il registro 	Collaboratori: <ul style="list-style-type: none"> • Web app • Server • Blockchain

Classe: Web app	
Descrizione: Applicazione web che stiamo realizzando	
Responsabilità: <ul style="list-style-type: none"> • Visualizzare home page con notifica sul SAL • Visualizzare schermata del SAL non compilato ma compilabile • Visualizzare schermata del SAL non compilato ma non compilabile • Visualizzare schermata del SAL compilato 	Collaboratori: <ul style="list-style-type: none"> • Server • Blockchain • Utente – DL

Classe: Server	
Descrizione: Si occupa della gestione della comunicazione tra web app e blockchain	
Responsabilità: <ul style="list-style-type: none"> • Richiamare funzione di compilazione del SAL 	Collaboratori: <ul style="list-style-type: none"> • Web app • Blockchain

Classe: Blockchain	
Descrizione: Indica gli smart contract necessari per salvare dati all'interno della blockchain	
Responsabilità: <ul style="list-style-type: none"> • Compilazione SAL • Inserire rigo SAL 	Collaboratori: <ul style="list-style-type: none"> • Server

6.1.7. Story Card 7

Classe: Utente - RUP	
Descrizione: Utente dell'applicazione	
Responsabilità: <ul style="list-style-type: none"> • Accedere SAL • Emanare certificato di pagamento 	Collaboratori: <ul style="list-style-type: none"> • Web app • Server • Blockchain

Classe: Utente - DL	
Descrizione: Utente dell'applicazione	
Responsabilità: <ul style="list-style-type: none"> • Accedere registro • Inserire certificato di pagamento 	Collaboratori: <ul style="list-style-type: none"> • Web app • Server • Blockchain

Classe: Web app	
Descrizione: Applicazione web che stiamo realizzando	
Responsabilità: <ul style="list-style-type: none"> • Visualizzare home page con notifica sul SAL • Visualizzare schermata con possibilità di emanare il certificato di pagamento • Visualizzare schermata con emanazione certificato avvenuta con successo • Visualizza home page con notifica sul registro di pagamento • Visualizza schermata con possibilità di inserimento certificato • Visualizzare schermata con inserimento certificato avvenuto con successo 	Collaboratori: <ul style="list-style-type: none"> • Server • Blockchain • Utente – RUP

Classe: Server	
Descrizione: Si occupa della gestione della comunicazione tra web app e blockchain	
Responsabilità: <ul style="list-style-type: none"> • Richiamare funzione di emanazione del certificato di pagamento • Richiamare funzione di inserimento del certificato di pagamento all'interno del registro di contabilità 	Collaboratori: <ul style="list-style-type: none"> • Web app • Blockchain

Classe: Blockchain	
Descrizione: Indica gli smart contract necessari per salvare dati all'interno della blockchain	
Responsabilità: <ul style="list-style-type: none"> • Emanazione certificato • Inserimento certificato nel registro 	Collaboratori: <ul style="list-style-type: none"> • Server

6.1.8. Story Card 8

Classe: Utente	
Descrizione: Riguarda gli attori considerati per l'applicazione	
Responsabilità: <ul style="list-style-type: none"> • Inserire le credenziali di accesso • Salvare le proprie credenziali • Accedere tramite il tasto di login 	Collaboratori: <ul style="list-style-type: none"> • Credenziali • Web app

Classe: Web app	
Descrizione: Applicazione web che stiamo realizzando	
Responsabilità: <ul style="list-style-type: none"> • Visualizzare la schermata di login • Visualizzare la schermata di login con il messaggio di errato inserimento delle credenziali di accesso • Visualizza schermata di inserimento lavorazioni da parte del RUP • Visualizza schermata di attesa di inserimento delle lavorazioni da parte del RUP • Visualizza home page • Controllare il corretto inserimento delle credenziali 	Collaboratori: <ul style="list-style-type: none"> • Utente • Server • Credenziali

6.1.9. Story Card 9

Classe: Utente	
Descrizione: Riguarda gli attori considerati per l'applicazione	
Responsabilità: <ul style="list-style-type: none"> • Accedere lista lavorazioni 	Collaboratori: <ul style="list-style-type: none"> • Web app

Classe: Web app	
Descrizione: Applicazione web che stiamo realizzando	
Responsabilità: <ul style="list-style-type: none"> • Visualizzare schermata con lavorazioni, rata di acconto e statistiche sui lavori 	Collaboratori: <ul style="list-style-type: none"> • Utente • Server • Blockchain

Classe: Server	
Descrizione: Si occupa della gestione della comunicazione tra web app e blockchain	
Responsabilità: <ul style="list-style-type: none"> • Richiamare funzione di visualizzazione lavorazioni • Richiamare funzione di visualizzazione rata di acconto • Visualizzare funzione di visualizzazione totale rimanente • Visualizzazione funzione di visualizzazione totale completato • Visualizzazione funzione di visualizzazione costo maturato • Visualizzazione funzione di visualizzazione costo totale 	Collaboratori: <ul style="list-style-type: none"> • Web app • Blockchain

Classe: Blockchain	
Descrizione: Indica gli smart contract necessari per salvare dati all'interno della blockchain	
Responsabilità: <ul style="list-style-type: none"> • Visualizzare lavorazioni • Visualizzare rata di acconto • Visualizzare totale rimanente • Visualizzare totale completato • Visualizzare costo maturato • Visualizzare costo totale 	Collaboratori: <ul style="list-style-type: none"> • Server

6.1.10. Story Card 10

Classe: Utente	
Descrizione: Riguarda gli attori considerati per l'applicazione	
Responsabilità: <ul style="list-style-type: none">• Accedere lista SAL	Collaboratori: <ul style="list-style-type: none">• Web app

Classe: Web app	
Descrizione: Applicazione web che stiamo realizzando	
Responsabilità: <ul style="list-style-type: none">• Visualizzare schermata con grafico andamento lavori• Visualizzare schermata con lista SAL	Collaboratori: <ul style="list-style-type: none">• Utente• Server• Blockchain

Classe: Server	
Descrizione: Si occupa della gestione della comunicazione tra web app e blockchain	
Responsabilità: <ul style="list-style-type: none">• Richiamare funzione di visualizzazione dati del SAL	Collaboratori: <ul style="list-style-type: none">• Web app• Blockchain

Classe: Blockchain	
Descrizione: Indica gli smart contract necessari per salvare dati all'interno della blockchain	
Responsabilità: <ul style="list-style-type: none">• Visualizzare dati SAL	Collaboratori: <ul style="list-style-type: none">• Server

6.2. User e Task model

Per ogni storia che richiede un'interazione con l'utente, sono stati presentati lo user model e il task model in modo da comprendere le operazioni che dovranno realizzare le nostre interfacce.

6.2.1. Story Card 1

User model	Utente – RUP
Task model	
	<ul style="list-style-type: none">• Inserire lavorazione• Decidere di modificare lavorazione• Modificare lavorazione• Visualizzare lavorazione• Terminare inserimento lavorazioni• Inserire rata di acconto• Decidere di modificare la rata di acconto• Modificare rata di acconto• Visualizzare rata di acconto• Visualizzare resoconto lavorazioni-rata di acconto• Terminare inserimenti

6.2.2. Story Card 2

User model	Utente – DL
Task model	
	<ul style="list-style-type: none">• Accedere giornale dei lavori corrente• Decidere di inserire rigo• Inserire rigo• Decidere di modificare rigo• Modificare rigo• Visualizzare giornale• Decidere di terminare giornale• Terminare giornale

User model	Utente
Task model	
	<ul style="list-style-type: none">• Accedere giornale dei lavori corrente• Visualizzare giornale

6.2.3. Story Card 3

User model	Utente - DL
Task model	<ul style="list-style-type: none">• Accedere libretto delle misure corrente• Decidere di inserire nuovo rigo• Inserimento rigo• Decidere di modificare un rigo• Modificare un rigo• Decidere di terminare un libretto• Terminazione libretto• Decidere di inserire allegato• Conferma inserire allegato• Decidere di ricercare allegato nel file system• Inserire allegato ricercato nel file system

6.2.4. Story Card 4

User model	Utente - DL
Task model	<ul style="list-style-type: none">• Accedere al libretto terminato• Decidere di inserire riserva al rigo con una firma con riserva• Inserire riserva al rigo con una firma con riserva

User model	Utente - RUP
Task model	<ul style="list-style-type: none">• Accedere al libretto terminato

User model	Utente - Ditta
Task model	<ul style="list-style-type: none"> • Accedere al libretto terminato • Decidere di inserire firma al rigo • Inserire firma al rigo • Decidere di inserire firma con riserva al rigo • Inserire firma con riserva al rigo

6.2.5. Story Card 5

User model	Utente - RUP
Task model	<ul style="list-style-type: none"> • Accedere area registro di contabilità • Decidere di compilare il registro

6.2.6. Story Card 6

User model	Utente - DL
Task model	<ul style="list-style-type: none"> • Accedere area registro di contabilità • Decidere di compilare il registro

6.2.7. Story Card 7

User model	Utente - RUP
Task model	<ul style="list-style-type: none">• Accedere SAL• Emanare certificato di pagamento

User model	Utente - DL
Task model	<ul style="list-style-type: none">• Accedere registro• Inserire certificato di pagamento

6.2.8. Story Card 8

User model	Utente
Task model	<ul style="list-style-type: none">• Inserire le credenziali di accesso• Salvare le proprie credenziali• Accedere tramite il tasto di login

6.2.9. Story Card 9

User model	Utente
Task model	<ul style="list-style-type: none">• Accedere lista lavorazioni

6.2.10. Story Card 10

User model	Utente
Task model	<ul style="list-style-type: none">• Accedere lista SAL

6.3. Presentation model

In questo paragrafo abbiamo riportato i vari mockup preparati nella fase di design della nostra web app, andando a raffigurare tutte le possibili pagine in cui è possibile imbattersi utilizzando l'applicazione.

6.3.1. Story Card 1

Home page in seguito al primo login del RUP

A Web Page
https://
Luigi Bianchi
RUP

Codice Lavoro: A1
Nome Lavoro: Scavi di fondazione
Costo Lavoro: 10000
AGGIUNGI

Inserimento lavorazioni avvenuto con successo da parte del RUP

A Web Page
https://
Luigi Bianchi
RUP

Codice Lavoro: A1
Nome Lavoro: Scavi
Costo Lavoro: 10000
HOIUPICA

Totale Opera: 10000

TERMINA L'INSEGNIMENTO DI LAVORAZIONI

Inserimento lavorazioni non avvenuto con successo da parte del RUP

A Web Page
https://
Luigi Bianchi
RUP

Codice Lavoro: A1
Nome Lavoro: Scavi di fondazione
Costo Lavoro: 10000
AGGIUNGI

Non è stato possibile inserire la lavorazione desiderata!!!
Messaggio di errore!!!

Modifica lavorazione inserita da parte del RUP

A Web Page
https://
Luigi Bianchi
RUP

Codice Lavoro: A1
Nome Lavoro: Scavi
Costo Lavoro: 10000
HOIUPICA

Totale Opera: 10000

TERMINA L'INSEGNIMENTO DI LAVORAZIONI

Risultato modifica corretta di una lavorazione da parte del RUP

Codice	Nome	Costo
A1	Fondamenta	10000

Totali Opere: 10000

TERMINA L'INSERIMENTO DI LAVORAZIONI

Risultato modifica non corretta di una lavorazione da parte del RUP

Codice	Nome	Costo
A1	Scavi	10000

Totali Opere: 10000

MODIFICA

TERMINA L'INSERIMENTO DI LAVORAZIONI

Inserimento rata di acconto da parte del RUP

Valore monetario rata di acconto
10000

INSERISCI E INIZIA IL CONTRATTO.

Inserimento/modifica rata di acconto avvenuto con successo da parte del RUP

Valore monetario rata di acconto
10000

Rata di acconto: 10000

MODIFICA

INSERISCI E INIZIA IL CONTRATTO.

Inserimento rata di acconto non avvenuto con successo da parte del RUP

Valore monetario rata di acconto
10000

Inserimento non avvenuto!
*** Messaggio di errore***

INSERISCI E INIZIA IL CONTRATTO.

Form di modifica della rata di acconto da parte del RUP

Valore monetario rata di acconto
10000

Rata di acconto: 10000

MODIFICA

INSERISCI E INIZIA IL CONTRATTO.

Modifica rata di acconto non avvenuta con successo da parte del RUP

Valore monetario rata di acconto
10000

Rata di acconto: 10000

MODIFICA E INIZIA IL CONTRATTO.

Modifica non avvenuta con successo!
*** Messaggio di errore***

Resoconto lavorazioni e rata di acconto/non accettazione dell'alert box per l'inizializzazione del contratto da parte del RUP

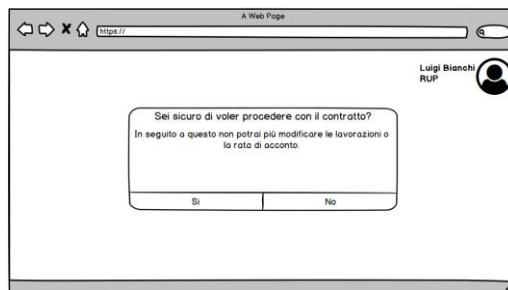
Codice	Nome	Costo
A1	Scavi	10000

Totali Opere: 10000

MODIFICA

TERMINA LE MODIFICHE E INIZIALIZZA IL CONTRATTO.

Alert Box per l'inizializzazione del contratto da parte del RUP



6.3.2. Story Card 2

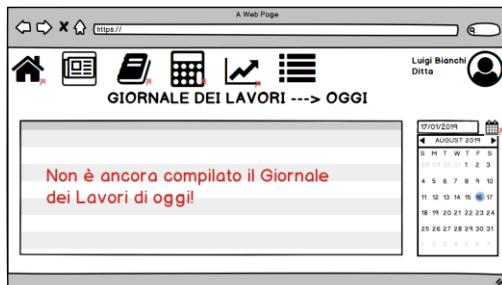
Notifica di non avvenuta compilazione del giornale dei lavori odierno (solo DL)



Visualizzazione giornale dei lavori non ancora compilato (solo DL)



Visualizzazione giornale dei lavori non ancora compilato (tutti gli altri utenti)



Form di compilazione giornale dei lavori (solo DL)



Inserimento corretto di un rigo del giornale (solo DL)

A screenshot of a web-based application titled "GIORNALE DEI LAVORI ---> OGGI". The interface includes a header with icons for home, calendar, calculator, chart, and grid. A user profile for "Mario Rossi DL" is shown. The main area displays a table with columns: "Descr lavoro", "Qualità pers", "Altrezz-numero", and "Ore di presenza del personale". A date picker shows "14/01/2019". Below the table is a weekly calendar for August 2019. At the bottom right is a "MODIFICA" button.

Inserimento non corretto di un rigo del giornale (solo DL)

A screenshot of the same web-based application. The "MODIFICA" button has been clicked, but an error message "Non è stato possibile inserire in nuovo rigo. ####Messaggio di errore####" appears in red text below the input fields. The rest of the interface is identical to the correct entry screenshot.

Form di sovrascrittura di un rigo del giornale (solo DL)

A screenshot of the web-based application showing an update operation. The "MODIFICA" button is present at the bottom right of the form area. The rest of the interface is identical to the correct entry screenshot.

Sovrascrittura non corretta di un rigo (solo DL)

A screenshot of the web-based application showing an update attempt that failed. An error message "Non è stato possibile inserire in nuovo rigo. ####Messaggio di errore####" is displayed in red text. The rest of the interface is identical to the correct update screenshot.

Sovrascrittura corretta di un rigo del giornale (solo DL)

A screenshot of the web-based application showing a successful update. The "MODIFICA" button is present at the bottom right. The rest of the interface is identical to the correct update screenshot.

Terminazione di un giornale (solo DL)

A screenshot of the web-based application showing the termination of a journal. A confirmation dialog box at the top asks "Sei sicuro di voler terminare la compilazione del giornale odierno? Una volta terminato non potrai più fare aggiunte o modifiche." Buttons for "Sì" and "No" are shown. The main form area shows the journal table with the last row highlighted in blue. The "MODIFICA" button is visible at the bottom right.

Visualizzazione di un giornale dei lavori terminato (tutti gli attori)

A screenshot of the web-based application showing a completed journal. The status is marked as "TERMINATO" at the bottom right. The journal table shows the last row in blue. The "MODIFICA" button is visible at the bottom right.

Visualizzazione di un giornale dei lavori non ancora terminato (no DL)

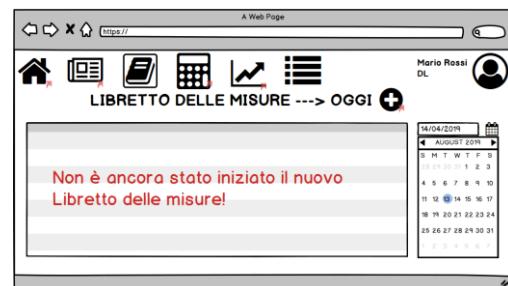
A screenshot of the web-based application showing an incomplete journal. The status is marked as "NON ANCORA TERMINATO" at the bottom right. The journal table shows the last row in blue. The "MODIFICA" button is visible at the bottom right.

6.3.3. Story Card 3

Home page (per tutti gli utenti)



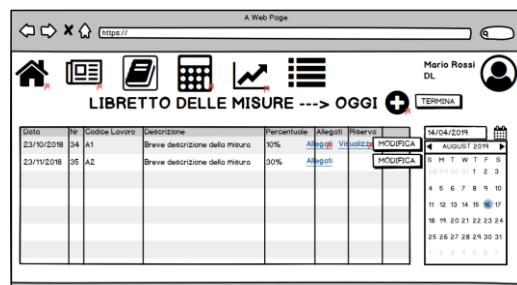
Visualizzazione libretto non ancora iniziato (solo DL)



Visualizzazione libretto non ancora iniziato (no DL)



Visualizzazione libretto compilato ma non terminato (DL)



Visualizzazione libretto compilato e terminato (ditta)



Visualizzazione libretto compilato ma non terminato (ditta)



Visualizzazione libretto compilato ma non terminato (RUP)



Visualizzazione libretto compilato e terminato (RUP)



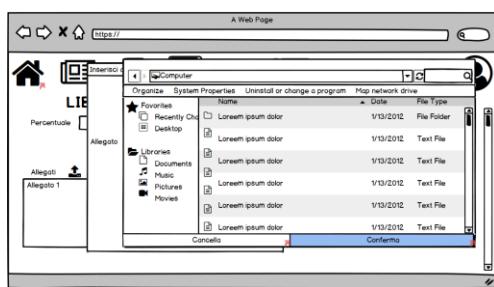
Form di inserimento di un nuovo rigo del libretto (solo DL)

A screenshot of a web-based application titled 'LIBRETTO DELLE MISURE ---> OGGI'. The interface includes a header with icons for home, documents, calculator, and chart, and a user profile for 'Mario Rossi DL'. Below the header, there are input fields for 'Percentuale' (10%) and 'Descrizione' (Breve descrizione della misura). A large button labeled 'Allegati' is present, with a sub-field 'Allegato 1' showing a placeholder box.

Form di inserimento di un allegato in un rigo (solo DL)

A screenshot of the same application interface. The 'Allegati' section has been expanded to show a dashed rectangular area for file uploads, with the text 'Choose File to Upload or drag and drop them here.' displayed inside.

Form di scelta di un allegato nel file system locale (solo DL)



Form di conferma dell'inserimento dell'allegato (solo DL)

A screenshot of the application after file selection. The 'Allegati' section now lists 'Allegato: file_selectionato.jpg'. A 'Conferma' (Confirm) button is visible at the bottom of the page.

Avvenuto inserimento dell'allegato (solo DL)



Inserimento di un nuovo rigo avvenuto con successo (solo DL)

A screenshot of the application showing the newly inserted row in the main table. The table includes columns for Data (Date), Nr (Number), Codice Lavoro (Work Code), Descrizione (Description), Percentuale (Percentage), Allegati (Attachments), and Riserva (Reserve). The last row shows the entry '23/10/2018 34 A1 Breve descrizione della misura 10% Allegati Riserva MOUPICA'.

Inserimento di un nuovo rigo non avvenuto (solo DL)

A screenshot of the application showing an error message: 'Non è stato possibile inserire in nuovo rigo. ###Messaggio di errore###'. This indicates that the attempt to insert a new row failed.

Visualizzazione di una riserva apportata in un rigo (tutti gli utenti)

A screenshot of the application showing a detailed view of a row. The table includes columns for Data (Date), Nr (Number), Descrizione (Description), Percentuale (Percentage), Allegati (Attachments), and Riserva (Reserve). The last column shows a 'Riserva Nr. Riserva 1 Percentuale: 40%' entry, along with a note 'Breve descrizione della riserva'.

Visualizzazione di un allegato presente in un rigo (tutti gli utenti)

A screenshot of a web-based application titled 'LIBRETTO DELLE MISURE'. The interface includes a header with icons for home, documents, calculator, and chart. A sidebar on the right shows a user profile for 'Mario Rossi DL'. The main content area displays a table with two rows of data. The first row has columns for 'Data' (23/10/2018), 'Nr' (34), 'Codice Lavoro' (A1), 'Descrizione' (Breve descrizione della misura), 'Percentuale' (10%), 'Allegati' (with a document icon), and 'Riserva' (Visualizza). The second row has similar columns. A 'Visualizza' button is located at the bottom right of the table.

Sovrascrittura di un rigo del libretto (solo DL)

A screenshot of the same 'LIBRETTO DELLE MISURE' application. The table now has three rows. The third row has the same columns as the others. A 'Percentuale' input field is present above the table, and a 'Descrizione' input field is to its right. A 'COMPILA' button is in the top right corner. A 'Allegati' section with a plus sign and 'Allegato 1' is visible below the table.

Sovrascrittura avvenuta con successo (solo DL)

A screenshot of the application after the update. The table now has four rows. The fourth row has the same columns as the others. The 'Percentuale' input field is now empty. A 'TERMINA' button is visible in the top right corner. A 'MODIFICA' button is located at the bottom right of the table.

Sovrascrittura non avvenuta con successo (solo DL)

A screenshot of the application where the update failed. The table has three rows. A red error message 'Non è stato possibile inserire in nuovo rigo. #!!!Messaggio di errore!!!' is displayed at the bottom. The 'Percentuale' input field is empty, and the 'Descrizione' input field is empty. A 'COMPILA' button is in the top right corner. A 'Allegati' section with a plus sign and 'Allegato 1' is visible below the table.

Alert Box per la terminazione libretto delle misure (solo DL)

A screenshot of the application showing a confirmation dialog box. The box asks 'Sei sicuro di voler terminare la compilazione del libretto? Una volta terminato non potrai più fare aggiunte o modifiche.' with 'Si' and 'No' buttons. The table below shows four rows of data. A 'TERMINA' button is in the top right corner. A 'MODIFICA' button is located at the bottom right of the table.

Libretto delle misure terminato (solo DL)

A screenshot of the application after the termination. The table now has five rows. The fifth row has the same columns as the others. The 'Percentuale' input field is empty, and the 'Descrizione' input field is empty. A 'TERMINA' button is in the top right corner. A 'MODIFICA' button is located at the bottom right of the table.

6.3.4. Story Card 4

Visualizzazione libretto compilato e terminato (ditta)

Data	Nr	Codice Lavoro	Descrizione	Percentuale	Allegati	Riserva	FIRMA	RISERVA
23/11/2018	34	A1	Breve descrizione della misura	10%	Allegati	Visualizza		
23/11/2018	35	A2	Breve descrizione della misura	30%	Allegati	Visualizza		

Alert Box per l'inserimento della firma/firma con riserva ad un rigo del libretto (ditta)

Sei sicuro di voler inserire la firma/firma con riserva a questo rigo?

SI

Firma avvenuta con successo (ditta)

Data	Nr	Codice Lavoro	Descrizione	Percentuale	Allegati	Riserva	FIRMA	RISERVA
23/11/2018	34	A1	Breve descrizione della misura	10%	Allegati	Visualizza	✓	
23/11/2018	35	A2	Breve descrizione della misura	30%	Allegati	Visualizza		

Firma con riserva avvenuta con successo (ditta)

Data	Nr	Codice Lavoro	Descrizione	Percentuale	Allegati	Riserva	FIRMA	RISERVA
23/11/2018	34	A1	Breve descrizione della misura	10%	Allegati	Visualizza		✓
23/11/2018	35	A2	Breve descrizione della misura	30%	Allegati	Visualizza		

Notifica firma con riserva inserita da parte della ditta (solo DL)

Mario Bianchi
DL

< TERMINATO

LIBRETTO DELLE MISURE ---> OGGI

Data	Nr	Codice Lavoro	Descrizione	Percentuale	Allegati	Riserva	INSERITO!	FIRMA	RISERVA
23/11/2018	34	A1	Breve descrizione della misura	10%	Allegati	Visualizza			
23/11/2018	35	A2	Breve descrizione della misura	30%	Allegati	Visualizza			

Form di inserimento della riserva (solo DL)

Mario Bianchi
DL

LIBRETTO DELLE MISURE ---> OGGI

Percentuale

Descrizione

Inserimento riserva avvenuto con successo (solo DL)

The screenshot shows a table with columns: Data, Nr, Codice Lavoro, Descrizione, Percentuale, Allegati, Riserva, and Visualizza. Two rows are present: one for 23/11/2018, Nr 34, A1, 'Breve descrizione della misura', 10%, Allegati, and Riserva 14/04/2019; another for 23/11/2018, Nr 35, A2, 'Breve descrizione della misura', 30%, Allegati, and Riserva 14/04/2019. A calendar for August 2019 is visible on the right.

Inserimento riserva non avvenuto con successo (solo DL)

The screenshot shows a table with a single row for 'Breve descrizione della misura'. A message at the bottom states: 'Non è stato possibile inserire la riserva! ***Messaggio di errore***'.

6.3.5. Story Card 5

Notifica di possibilità di compilazione di un registro della contabilità (solo RUP)



Schermata di compilazione del registro di contabilità (solo RUP)

The screenshot shows a message: 'Il presente registro della contabilità non è ancora stato compilato. Cliccando sul tasto COMPILA verrà compilato in modo automatico.'

Visualizzazione registro compilato (tutti gli attori)

The screenshot shows a table with columns: Nro riga Lib, Codice lavoro, Nome lavoro, Perc raggiunto, Perc singolo lavoro, Costo singolo, Perc totale, and Costo raggiunto. One row is present: Nro riga Lib 1, Codice lavoro A1, Nome lavoro Struttura A-Scovi, Perc raggiunto 50%, Perc singolo lavoro 5%, Costo singolo 10000, Perc totale 15%, and Costo raggiunto 5000.

Visualizzazione registro non compilato (no RUP)

The screenshot shows a message: 'Il presente registro della contabilità non è ancora stato compilato. Prego attendere la compilazione da parte del RUP.'

6.3.6. Story Card 6

Notifica di possibilità di compilazione di un SAL (solo DL)



Schermata di compilazione del SAL (solo DL)



Visualizzazione SAL compilato (tutti gli attori)

A screenshot of a web browser window titled "A Web Page". The URL bar shows "https://". The main content area has a header "STATO AVANZAMENTO LAVORI" with icons for home, document, pen, calculator, and chart. On the right, it shows "Mario Rossi DL" and a user icon. Below the header is a table:

Nr.rigo	Lib	Codice lavoro	Nome lavoro	Struttura A-Scovi	Perc raggiunto	Perc singolo lavoro	Costo singolo	Perc totale	Costo raggiunto
1-2	A1				50%	3%	10000	15%	5000

Visualizzazione SAL non compilato (no DL)

A screenshot of a web browser window titled "A Web Page". The URL bar shows "https://". The main content area has a header "STATO AVANZAMENTO LAVORI" with icons for home, document, pen, calculator, and chart. On the right, it shows "Luigi Bianchi RUP" and a user icon. Below the header is a message: "Il presente registro della contabilità non è ancora stato compilato." (The current accounting register has not yet been compiled.) and "Prego attendere la compilazione da parte del RUP." (Please wait for the RUP to compile it.)

6.3.7. Story Card 7

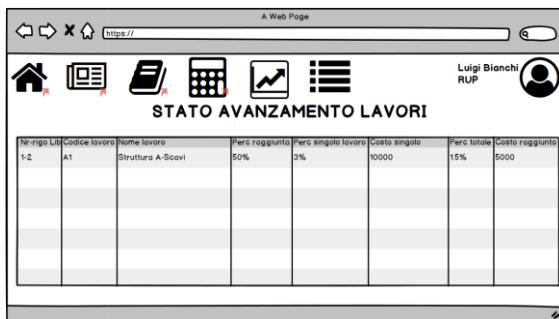
Notifica di possibilità di inserimento del certificato relativo al SAL compilato (solo RUP)



Visualizzazione schermata con bottone di inserimento certificato (solo RUP)



Inserimento certificato avvenuto



Notifica di possibilità di inserimento del certificato emanato dal RUP all'interno di un registro di contabilità (solo DL)



Visualizzazione schermata con bottone di inserimento certificato all'interno del registro (solo DL)



Inserimento certificato nel registro avvenuto



6.3.8. Story Card 8

Pagina di login

Inserisci le credenziali di accesso per poter accedere al sito
Nome Utente
Password
LOGIN

Credenziali errate

Inserisci le credenziali di accesso per poter accedere al sito
Nome Utente
Password
Nome Utente o password non sono corretti.
LOGIN

Home page in seguito al primo login di un utente che non sia il RUP (nel caso in cui il RUP non abbia ancora completato la procedura di inizializzazione del contratto in seguito il primo accesso)



Home page in seguito al primo login di un utente che non sia il RUP (nel caso in cui il RUP abbia già completato la procedura di inizializzazione del contratto in seguito al primo accesso)



6.3.9. Story Card 9

Home page (tutti gli utenti)



Lista lavorazioni (tutti gli utenti)

A screenshot of a web browser window titled "A Web Page". The address bar shows "https://". The main content area has a header with icons for home, documents, calculator, and a graph. Below the header, it says "LISTA LAVORAZIONI". There is a table with columns: "Codice Lavoro", "Nome Lavoro", "Costo", and "Percentuale raggiunta". One row is visible: "A1" (Nome Lavoro), "Muro esterne" (Nome Lavoro), "102438" (Costo), "20%" (Percentuale raggiunta). In the top right corner, there is a user profile for "Luigi Bianchi RUP". At the bottom, there are summary statistics: "Totale raggiunto: 0.6%", "Costo totale: 283292842", and "Costo raggiunto: 2388".

6.3.10. Story Card 10

Home page (tutti gli utenti)



Visualizzazione andamento dei lavori in base ai SAL emessi



6.4. Navigation Model

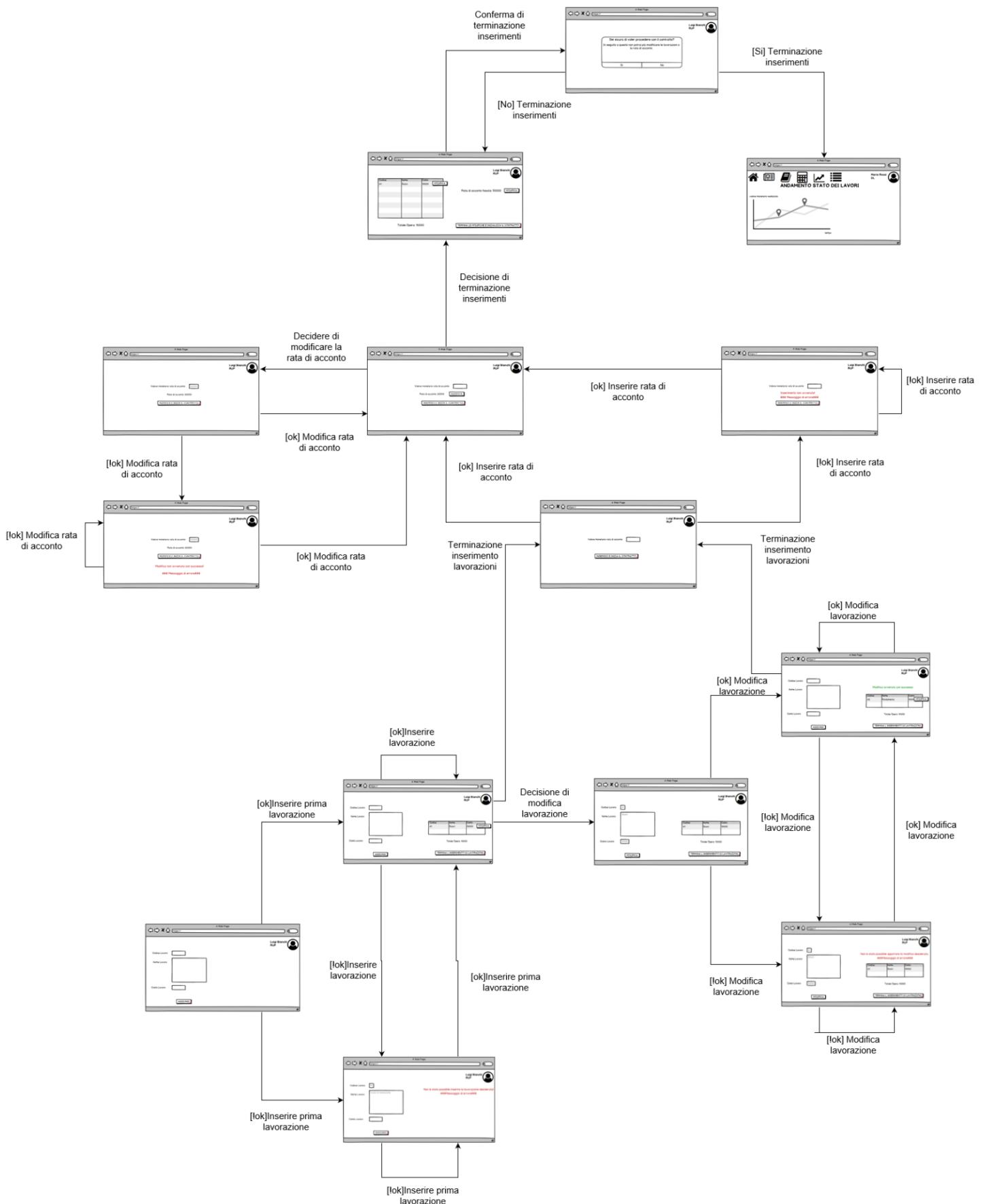
In sequenza sono stati riportati i navigation model relativi alle varie storie utente, che sono utili ad una maggior comprensione della navigazione all'interno del sito web.

6.4.1. Story Card 1

Per quanto riguarda la prima story card abbiamo riportato il processo di inserimento dei dati necessari per poter avviare il processo di gestione di un contratto d'appalto, ovvero lavorazioni e rata di acconto. L'attore che può avviare questo processo di inserimento è solamente il RUP, per questo abbiamo deciso di sotto intendere la condizione booleana [RUP] necessaria per ogni azione all'interno del navigation model.

Com'è possibile osservare nella figura sottostante, il tutto si alterna in due differenti momenti. Innanzitutto, il RUP va ad inserire le lavorazioni con possibilità di modifica e visualizzazione di queste. Dopo di che, conferma la terminazione di inserimenti delle lavorazioni e procede con l'inserimento della rata di acconto. La rata di acconto è una misura unica, ma abbiamo ritenuto possibile per il RUP andare a modificarla in caso di inserimenti non corretti.

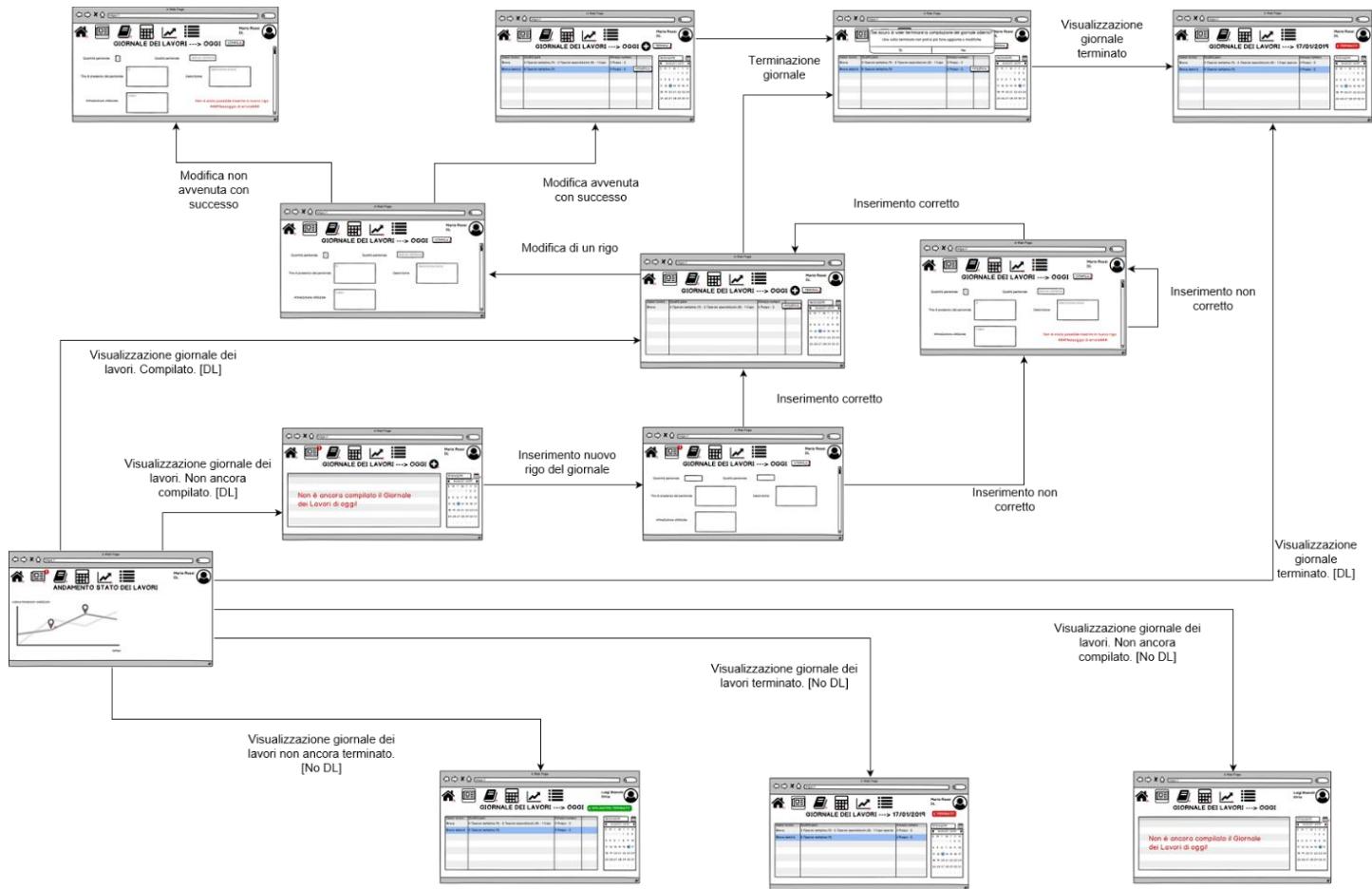
Al termine di tutti gli inserimenti, sarà proposto un resoconto riepilogativo dei vari inserimenti fatti. In seguito, si passerà alla fase di gestione del contratto di appalto, dalla quale non sarà più possibile fare modifiche o nuovi inserimenti delle lavorazioni.



6.4.2. Story Card 2

Abbiamo qui riportato il navigation model che riguarda inserimenti e modifiche di righi appartenenti ad un giornale dei lavori. Inoltre, è riportato il processo di terminazione di un giornale, che come si può notare, è un'operazione possibile solamente dopo l'inserimento di almeno un rigo del giornale, andando quindi ad impedire l'inserimento di un giornale vuoto.

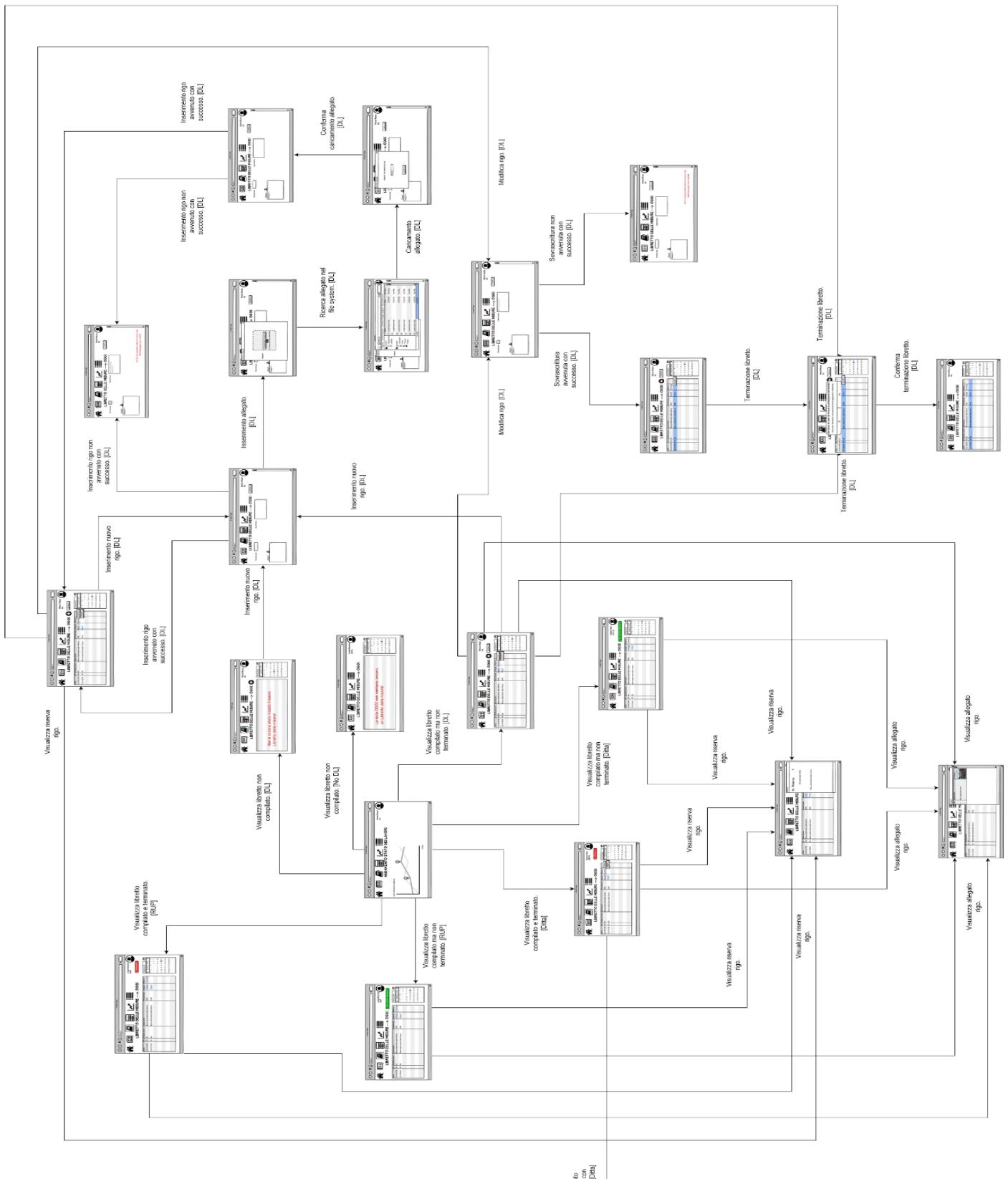
Anche in questo caso, le operazioni riportate sono possibili solamente ad uno specifico attore, che in questo caso è il direttore dei lavori. A differenza del navigation model precedente, abbiamo però inserito la codizione booleana [DL] o la negazione [!DL] per indicare se quel ramo è percorribile ad un certo attore. Si è però riportati la condizione solamente per la prima operazione, ovvero la visualizzazione di un giornale per differenti condizioni. Quindi nelle sotto operazioni dei diversi rami, la condizione [DL] o [!DL] sarà implicita.



6.4.3. Story Card 3

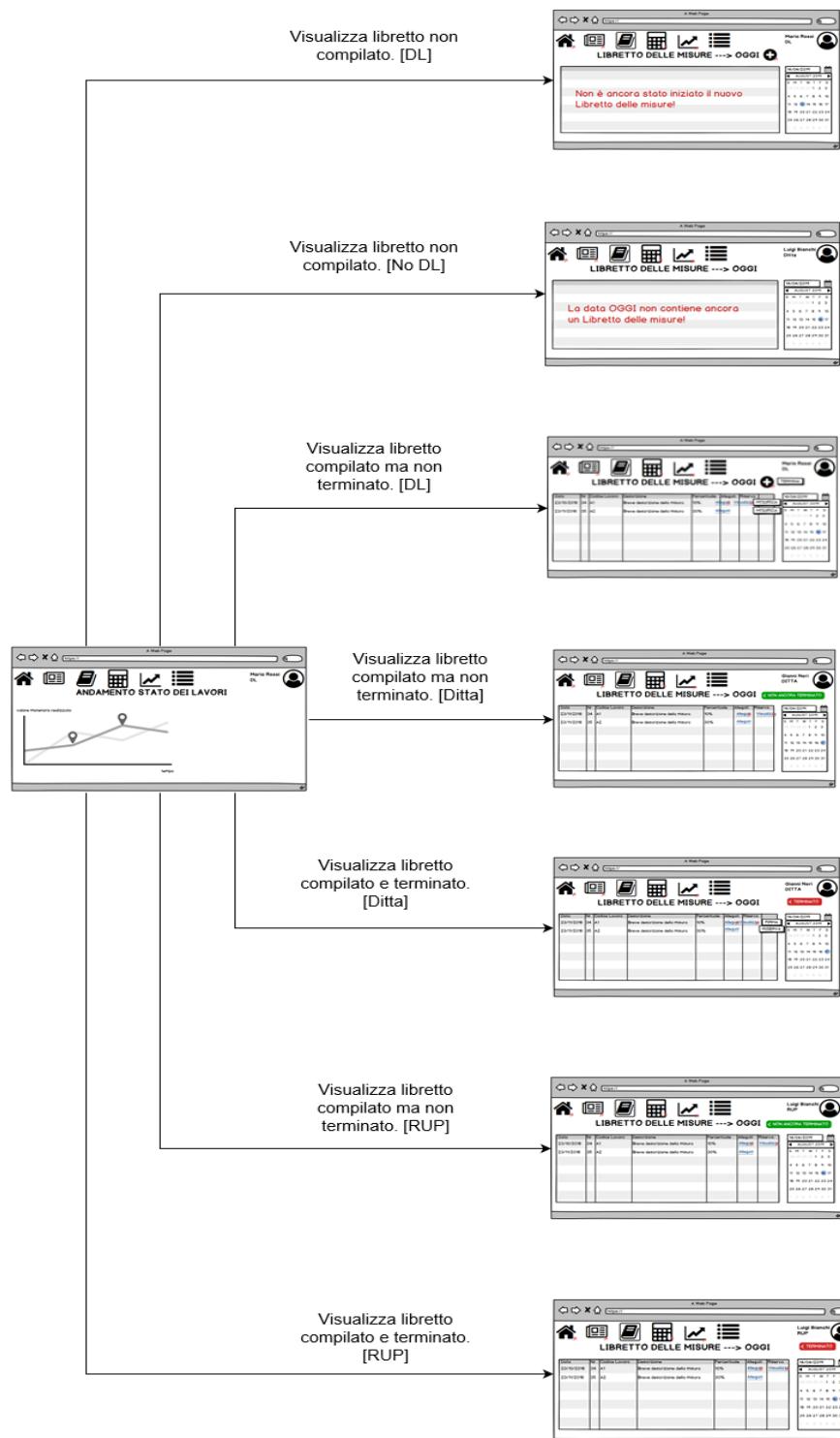
Per quanto riguarda la terza e la quarta story card, dato che le due sono molto collegate tra loro, abbiamo voluto rappresentare innanzitutto un navigation model che le racchiuda entrambe (prima figura).

Essendo però la figura molto estesa, abbiamo poi separato le varie parti che compongono le due story card e raffigurato in diversi modelli.



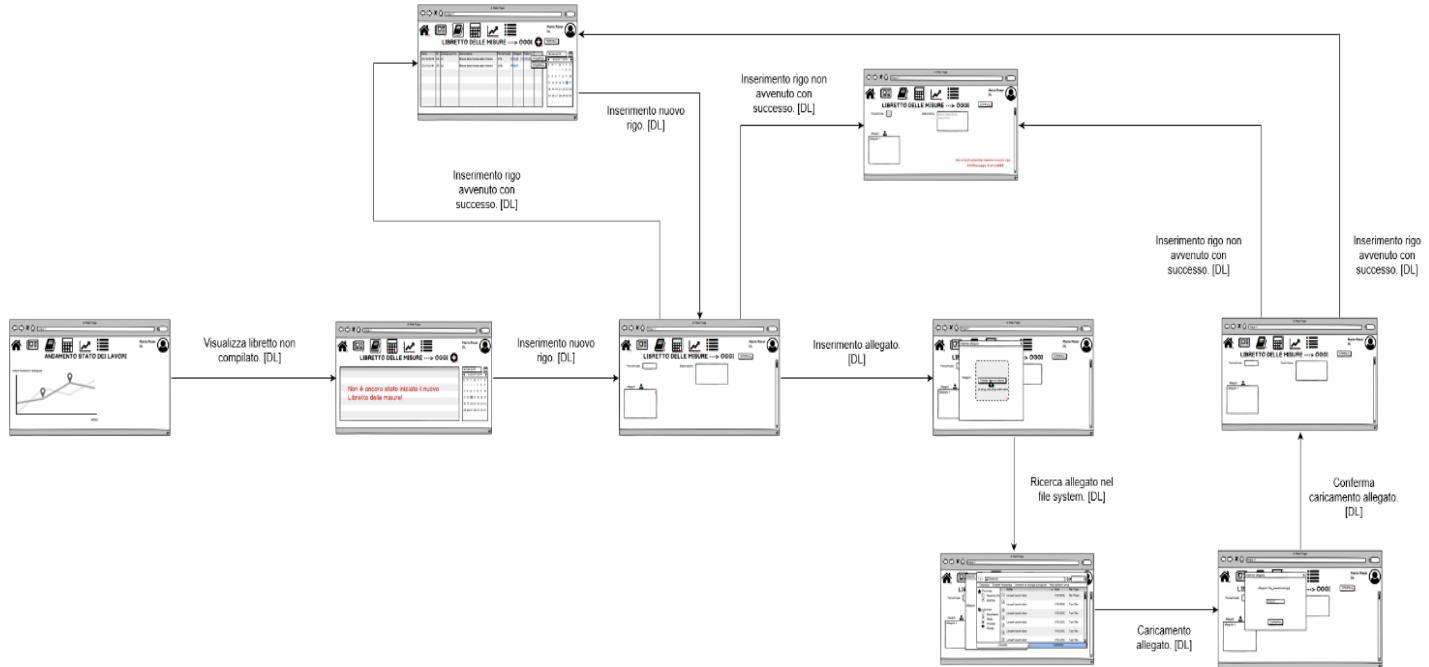
La figura sottostante rappresenta le diverse possibilità (secondo diverse condizioni verificabili) che si hanno nell'andare a consultare un libretto delle misure.

Per ragioni di maggior leggibilità non abbiamo reso in modo esplicito le condizioni [compilato], [terminato] o le loro negazioni, ma ovviamente sono da considerare in ogni differente ramo, infatti la visualizzazione di un libretto che sia compilato, terminato o viceversa, dipende appunto dal verificarsi delle condizioni sopra citate.

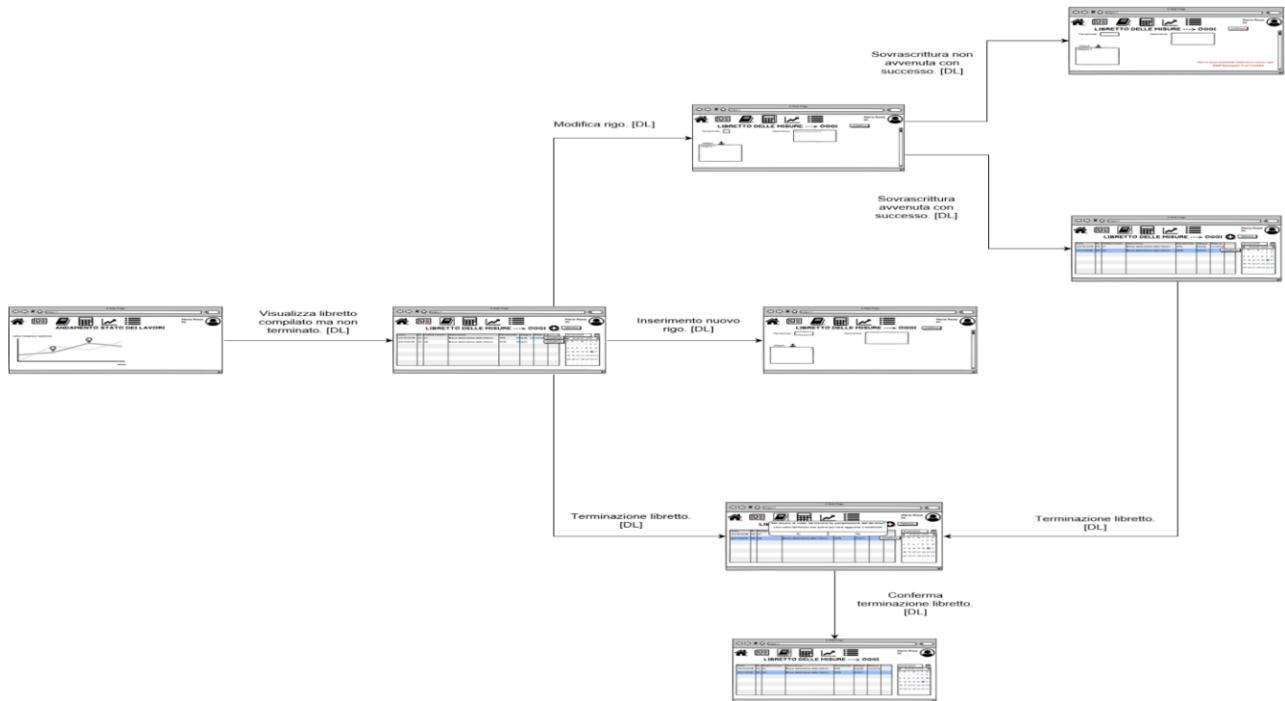


In questo presentation model è rappresentata la fase di inserimento di un nuovo rigo nel libretto.

Non è stata inserita la fase di modifica di un rigo preesistente, che invece viene presentata nella figura successiva.



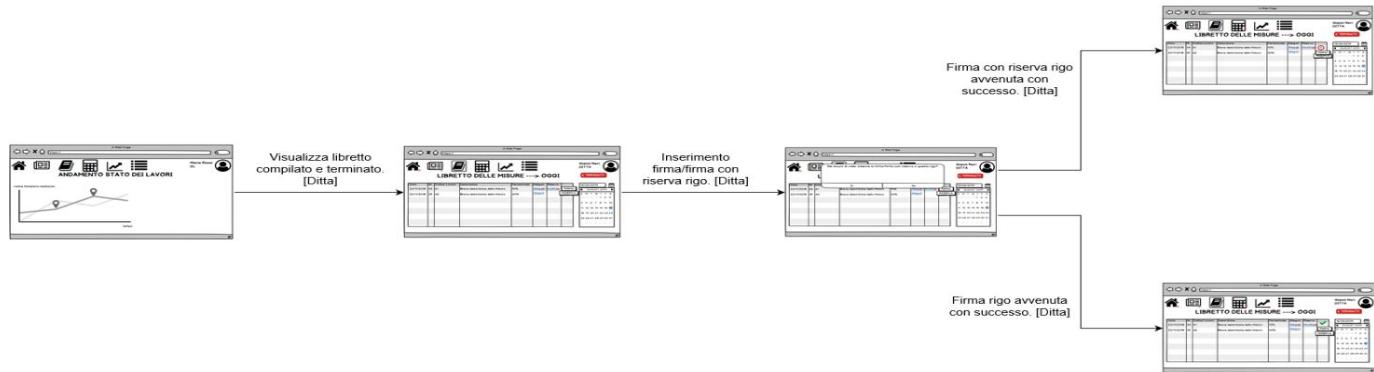
Come accennato sopra, la figura sottostante presenta la fase di sovrascrittura di un rigo e di terminazione.



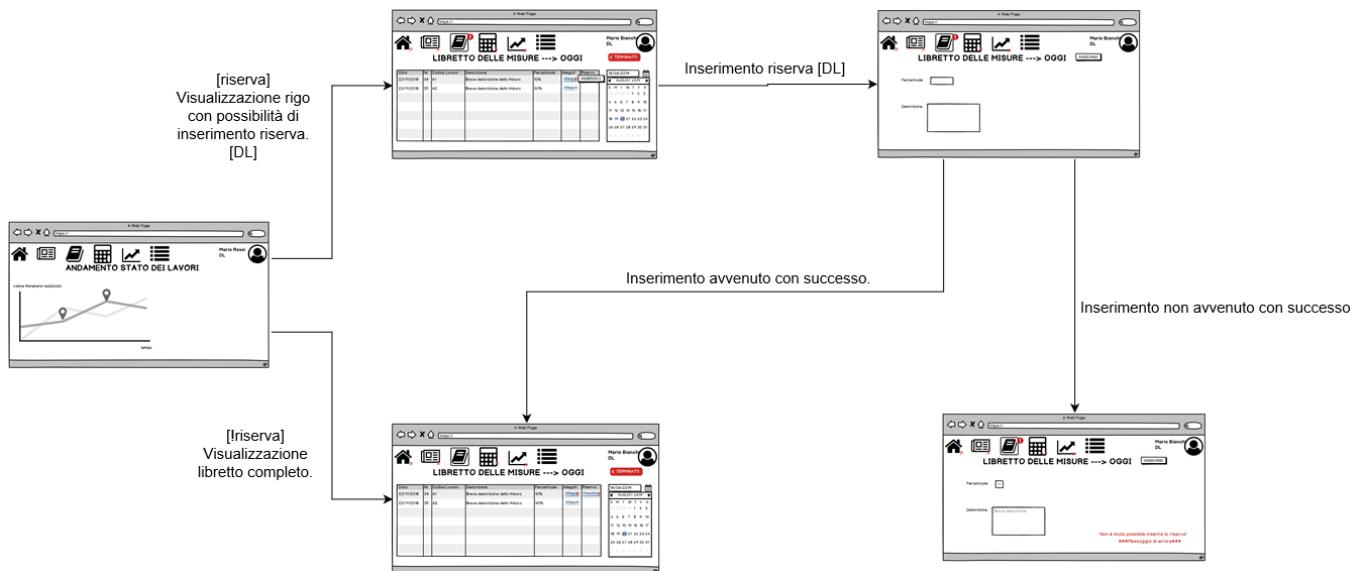
6.4.4. Story Card 4

La story card 4 riporta la fase di firma o firma con riserva di uno specifico rigo del libretto. L'operazione di firma è molto semplice dal punto di vista del modello di navigazione (riportata nel primo navigation model), infatti la ditta (unico attore che può dar via alle operazioni di firma) può decidere se firmare o firmare con riserva un rigo del libretto.

Nel caso di firma normale, il rigo viene considerato come completo di tutti i requisiti necessari; mentre nel caso di firma con riserva, si necessita dell'inserimento di una riserva da parte del direttore dei lavori (riportata nel secondo navigation model).



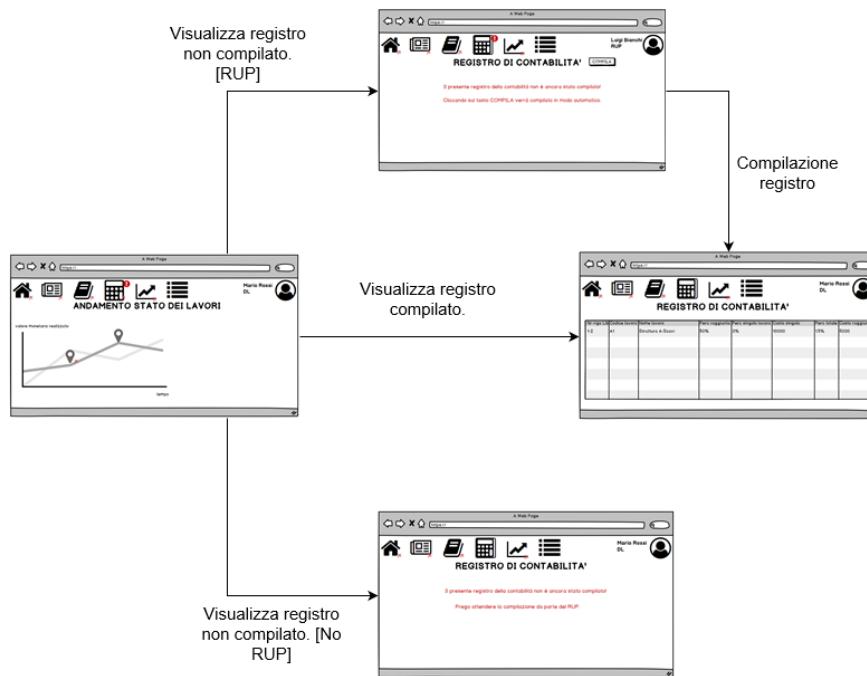
Per quanto riguarda le riserve andiamo a considerare l'inserimento di una riserva in un unico rigo del libretto e andiamo a separare in due diversi casi: quando la riserva è già stata esplicitata per quel rigo ([riserva] in figura) e quando invece non è ancora stata presentata la riserva da parte del direttore ([!riserva]).



6.4.5. Story Card 5

In questo modello è rappresentata l'operazione di inserimento di un nuovo registro. Dato che il tutto verrà gestito in modo automatico, il Responsabile Unico del Procedimento dovrà solamente abilitare alla creazione del nuovo registro, senza dover inserire i vari dati in modo manuale.

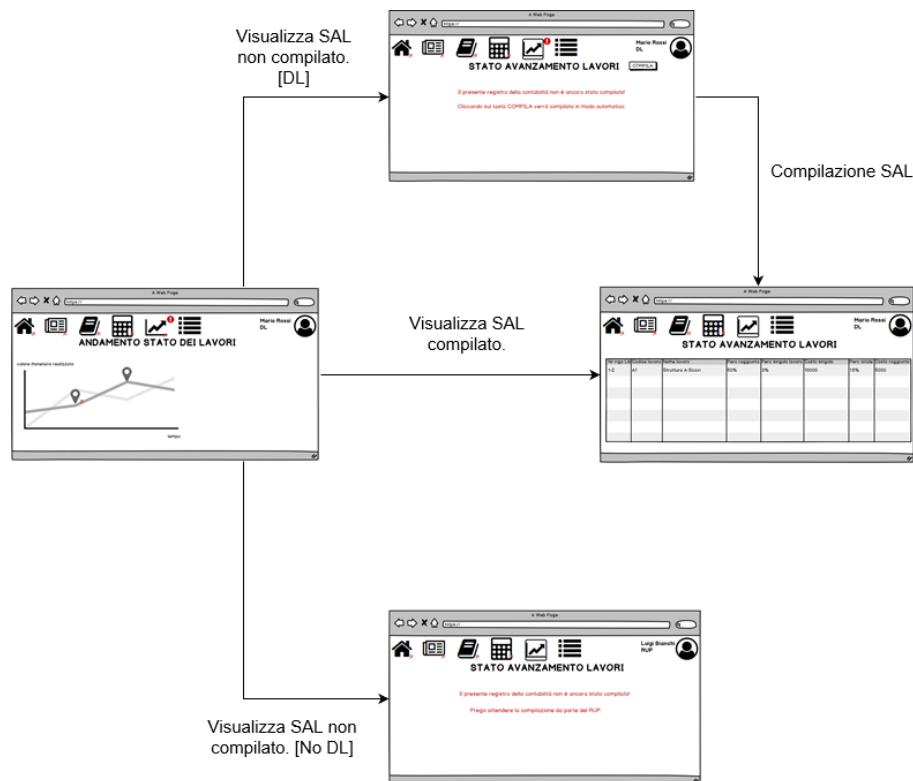
Anche in questo caso abbiamo separato la visualizzazione del registro di contabilità nei casi in cui sia già stato compilato o meno, ma senza riportare le condizioni [compilato]/[!compilato]. Questo perché ovviamente la visualizzazione di un registro compilato presuppone appunto il verificarsi della condizione booleana sopra citata.



6.4.6. Story Card 6

Simile, se non del tutto uguale al registro, avviene la compilazione di un nuovo SAL. Infatti, il DL in questo caso andrà semplicemente ad abilitare la creazione automatica di un SAL e il tutto verrà gestito in modo trasparente.

Anche in questo caso non si sono riportate le condizioni [compilato] / [!compilato] che sono da considerarsi implicite.



6.4.7. Story Card 7

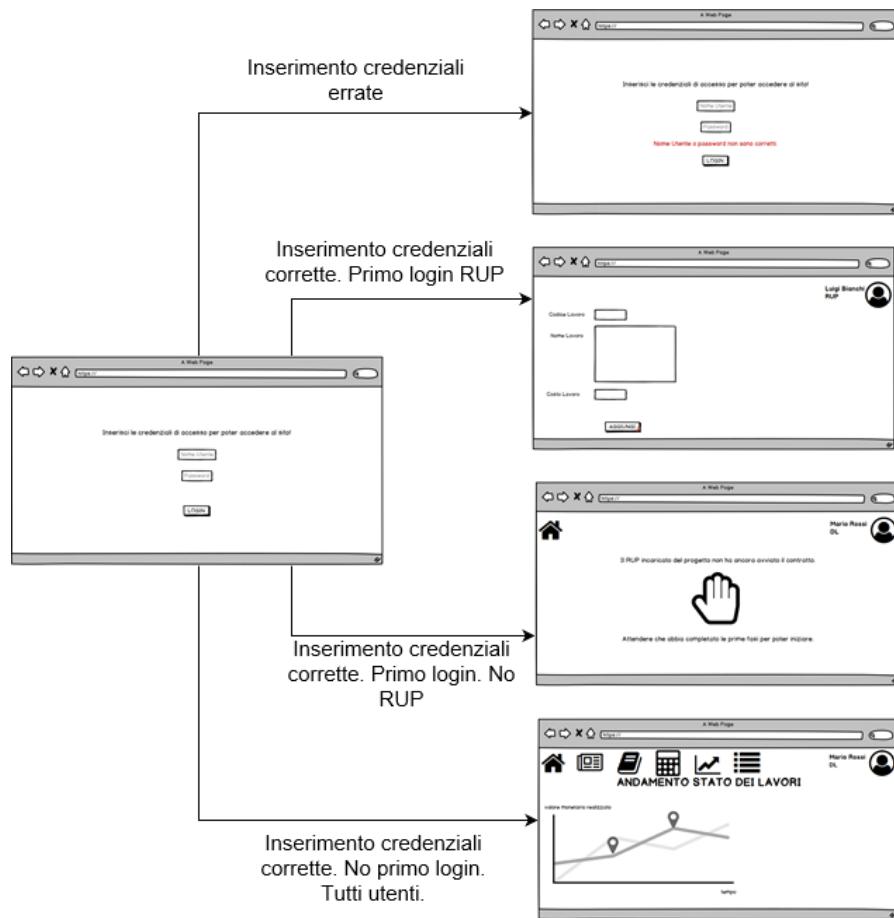
I navigation model della story card 7 sono stati separati in due, dato che il primo riguarda il RUP, che ha innanzitutto lo scopo di inserire il certificato di pagamento relativo ad uno stato di avanzamento dei lavori. Mentre il secondo riguarda il DL (direttore dei lavori) che ha invece lo scopo di inserire un certificato rilasciato dal RUP, all'interno di un registro di contabilità.



6.4.8. Story Card 8

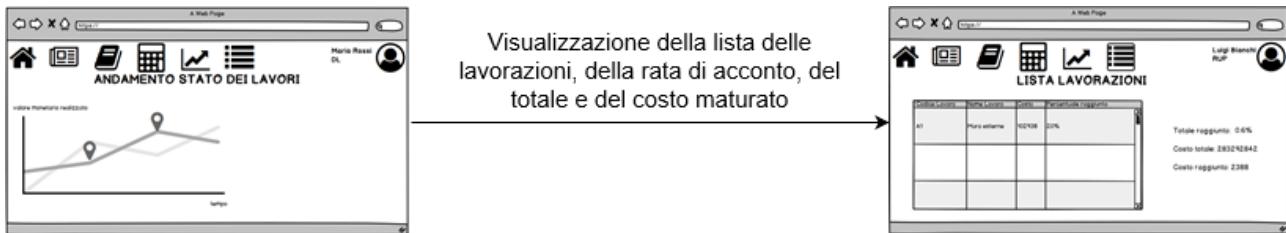
Nella presente story card abbiamo rappresentato l'operazione di login al sito AppAlto.

Siamo andati a separare differenti casi, come l'accesso di uno specifico attore o l'accesso per la prima volta alla web app. Infatti, proprio nel caso di un primo accesso, non dovranno essere visibili e possibili le operazioni di gestione dell'appalto, ma come accennato nella prima storia utente, solamente il RUP potrà fare un'operazione, che è quella di inserimento delle lavorazioni e della rata di acconto. Per tanto, nel caso di primo login, senza che quest'operazione di inserimento non sia stata portata a termine dal RUP, non dovrà essere possibile alla Ditta o al Direttore dei Lavori poter accedere alla gestione dell'appalto.



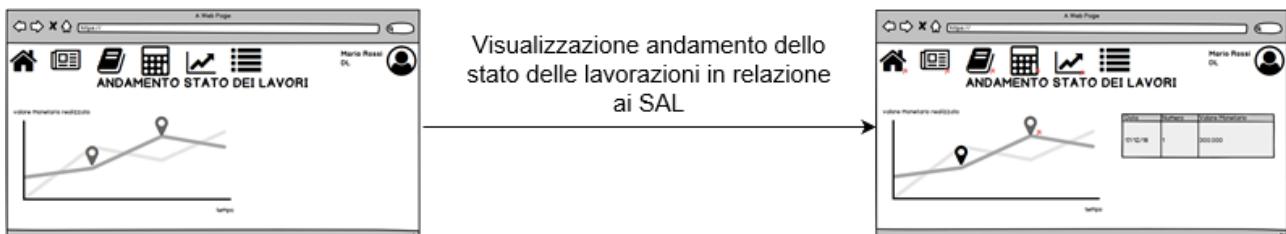
6.4.9. Story Card 9

L'operazione di visualizzazione delle lavorazioni e dell'avanzamento del progetto è molto semplice e intuitiva, ma abbiamo comunque riportato il navigation model per evitare confusioni.



6.4.10. Story Card 10

Anche per quanto riguarda il grafico presente nella home page di AppAlto non si hanno grosse difficoltà di comprensione, ma abbiamo comunque voluto riportare il modello.



6.5. Dialogue model

I dialogue model servono a mostrare come l'utente possa interagire con l'interfaccia per poter compiere i propri task.

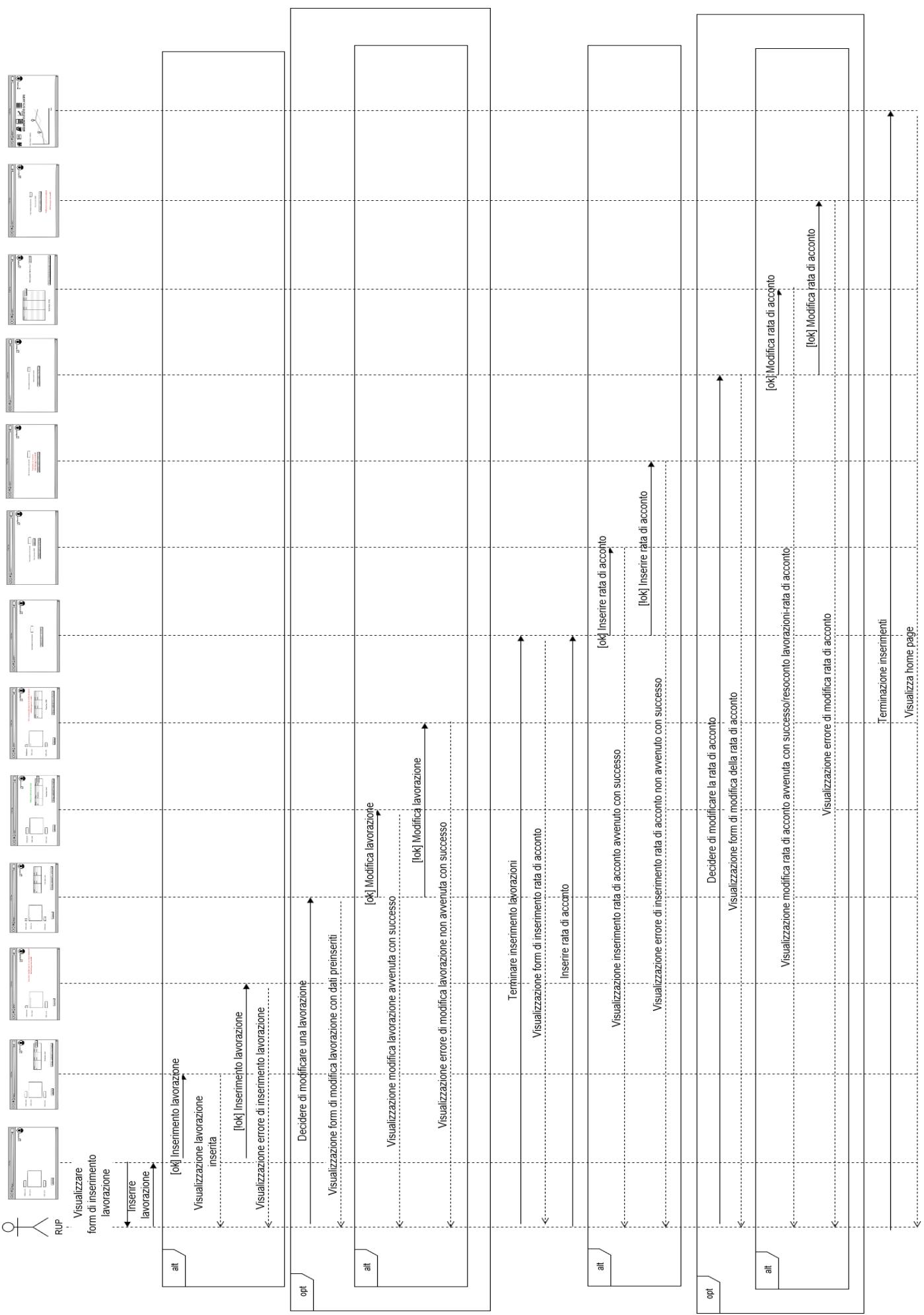
Di seguito abbiamo riportato i vari modelli, separati per le varie storie utente.

6.5.1. Story Card 1

Molto articolato è il dialogue model della prima story card, in cui il RUP può andare ad inserire una nuova lavorazione. In seguito a questa ha due differenti tipo di rami opzionali: andare ad inserire un nuovo rigo, oppure andare a modificare il rigo precedentemente inserito. In seguito a questi rami opzionali, che sono più volte percorribili dall'utente, dovrà andare a terminare l'inserimento delle lavorazioni.

Solo in seguito a ciò, sarà possibile inserire una rata di acconto. Anche in questo caso si ha la presenza di un ramo opzionale, che riguarda la possibilità di modifica della rata.

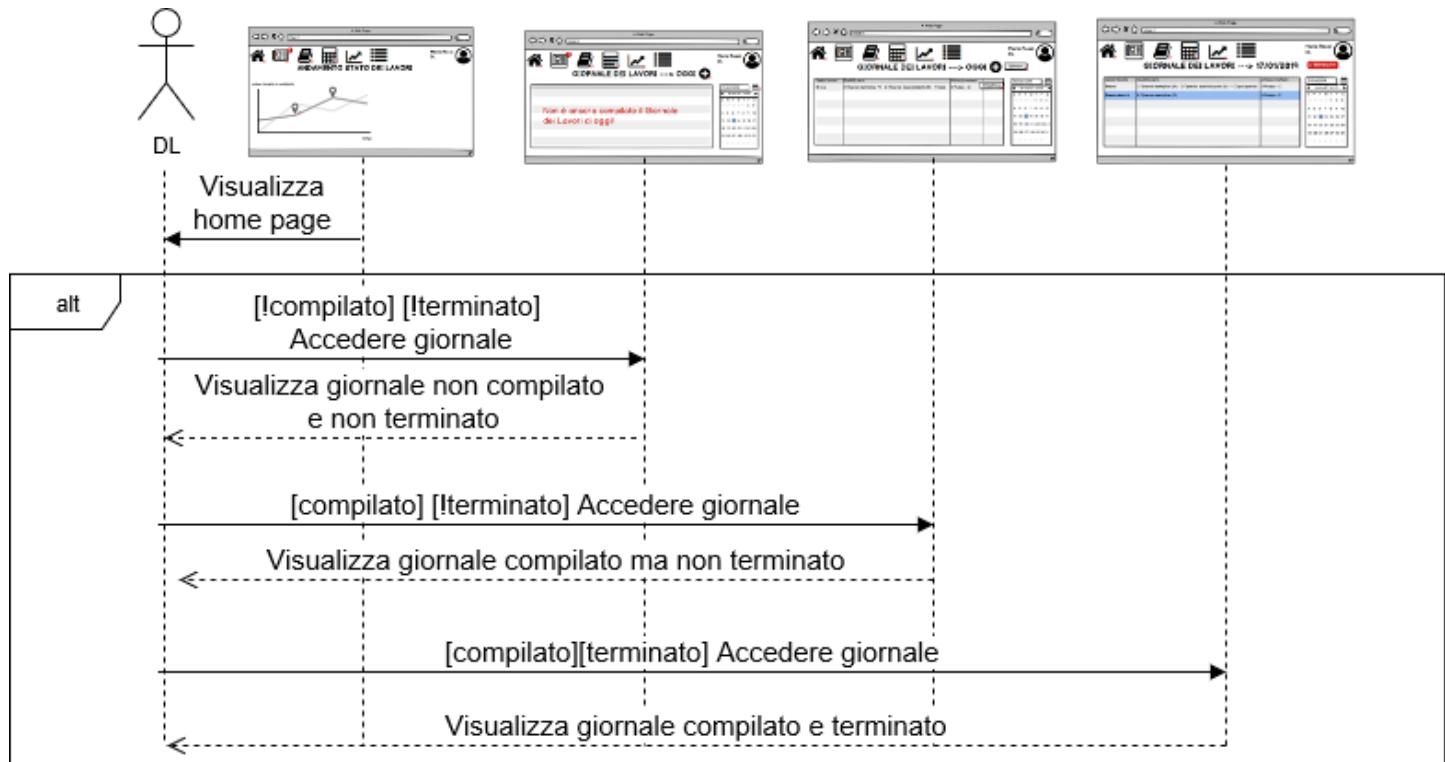
Il tutto si andrà a concludere con un riepilogo dei vari inserimenti e la conseguente terminazione di questi, che porterà il RUP alla home page del sito.



6.5.2. Story Card 2

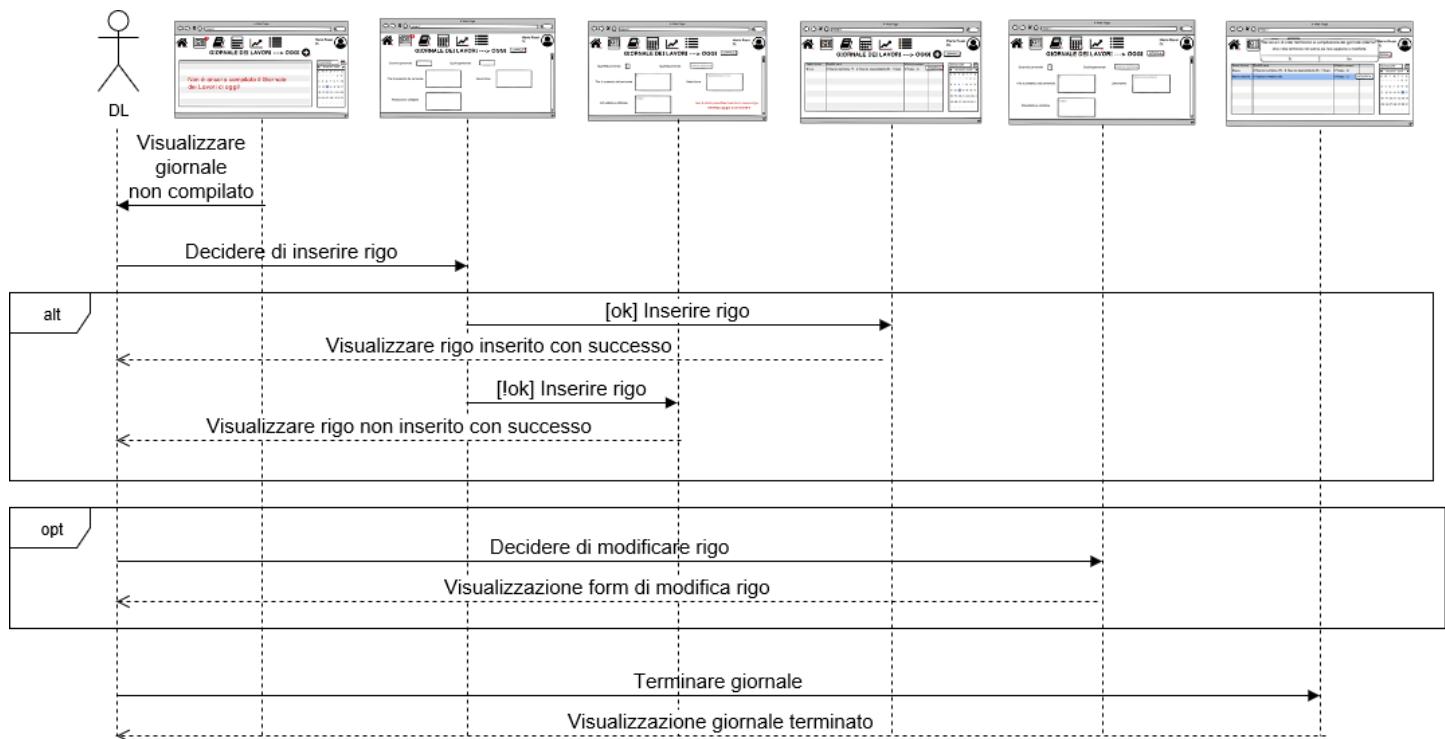
Per quanto riguarda il flusso 1 della seguente story card (l'utente ad interagire con l'applicazione è il direttore dei lavori), abbiamo pensato di suddividere il dialogue model in più dialogue model in modo da avere una rappresentazione più semplice e comprensibile di questi.

Nel primo dialogue model abbiamo rappresentato l'accesso da parte del direttore dei lavori al giornale dei lavori. Questo verrà reindirizzato a diverse pagine a seconda della veridicità o meno di diverse condizioni (come è possibile vedere in figura sotto).

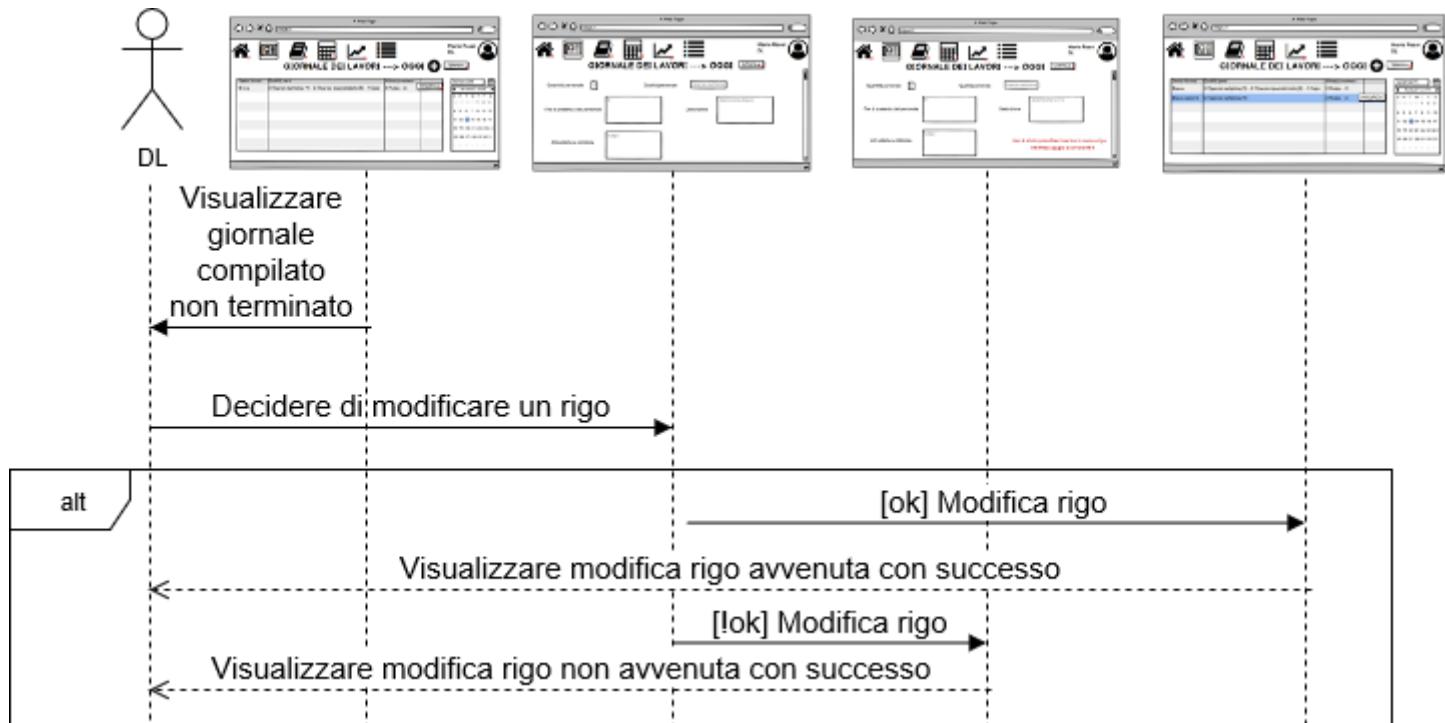


Nel secondo è invece stato presentato il flusso di lavoro nel caso in cui il direttore dei lavori si accinge ad inserire un nuovo rigo in un giornale dei lavori che non è ancora stato compilato (una delle condizioni rappresentate nel primo diagramma). Inoltre, il flusso di lavoro rappresentato è stato anch'esso semplificato. Infatti, nel ramo opzionale in cui il DL può decidere se sovrascrivere un rigo già esistente del libretto, non è stato presentato nella sua totalità questo ramo opzionale, che viene invece rappresentato nel diagramma seguente al sottostante.

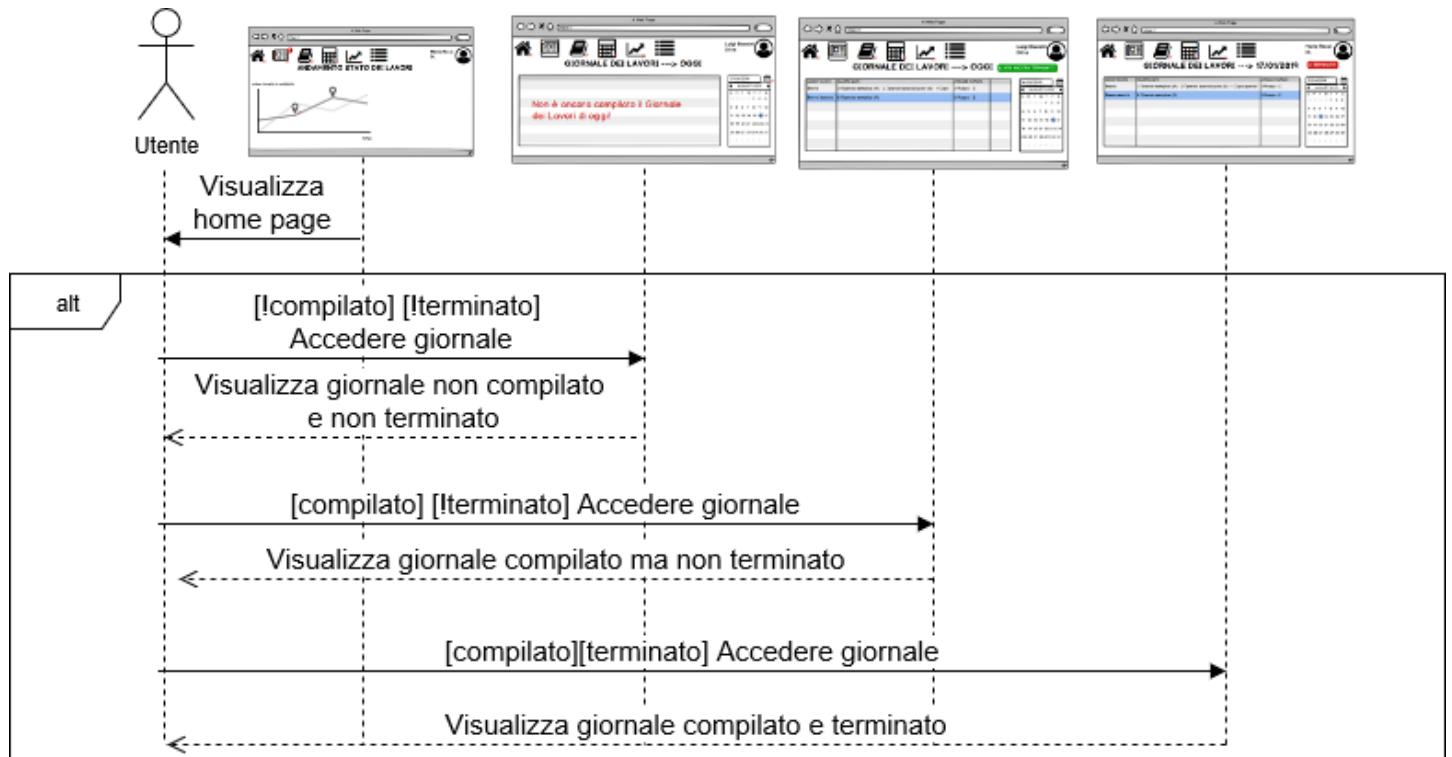
Un dialogue model praticamente identico si ha nel caso in cui il DL voglia andare ad inserire un rigo ad un giornale già iniziato ad essere compilato, infatti in questo caso andrebbe a cambiare solamente la pagina iniziale. Per questo motivo abbiamo ritenuto inutile riportare un diagramma molto simile al sottostante.



Nel seguente diagramma, come accennato prima, è riportato il ramo opzionale della modifica di un rigo.



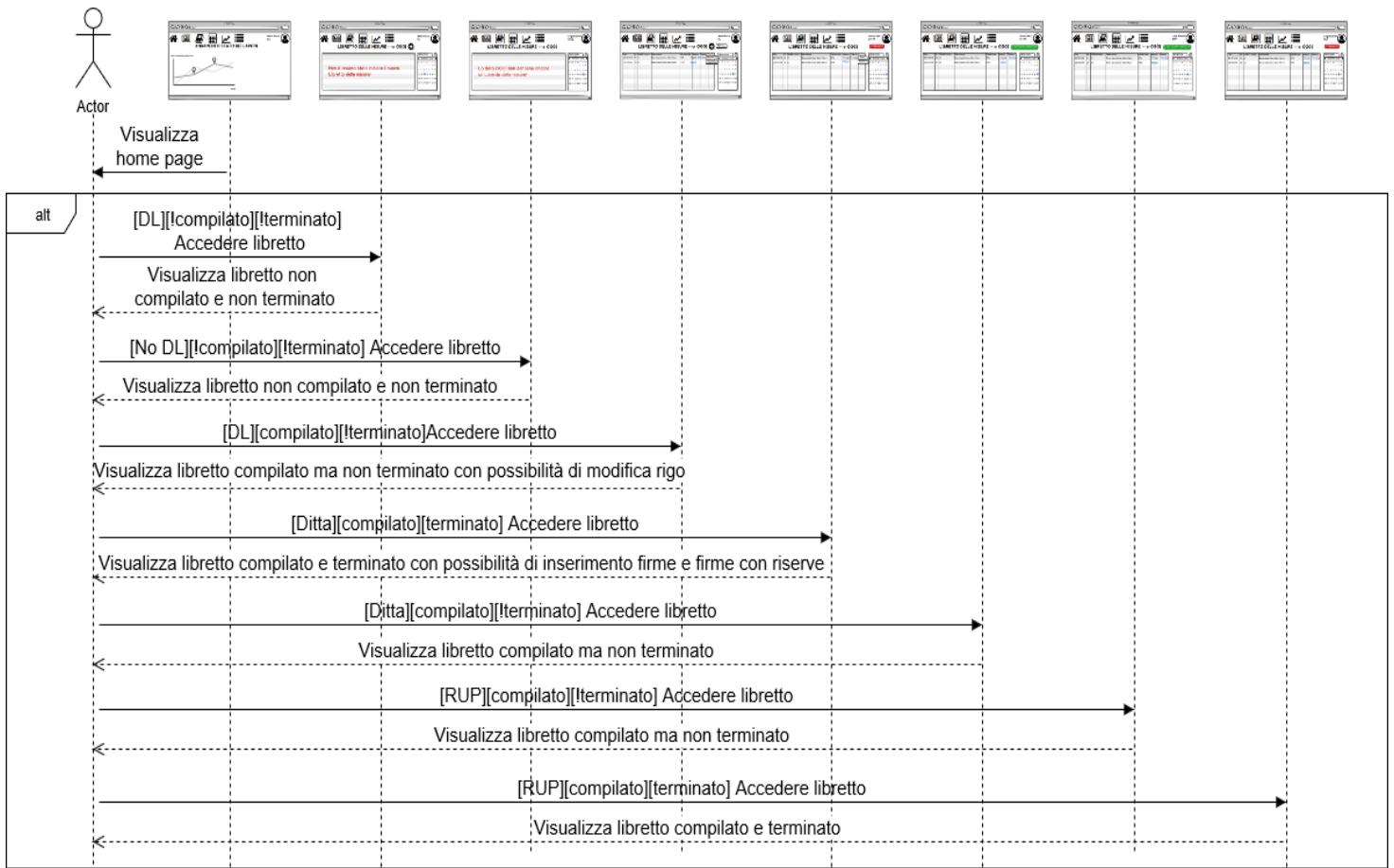
Per quanto riguarda invece il flusso 2, ovvero il flusso in cui l'utente che interagisce con la nostra applicazione non è il direttore dei lavori, ma il RUP o la ditta appaltatrice, si ha un unico dialogue model, che risulta essere molto simile al primo dialogue model per il flusso 1. Per maggiori informazioni abbiamo riportato sotto il diagramma.



6.5.3. Story Card 3

Anche in questo caso siamo andati a suddividere i dialogue model in più modelli in modo da rendere più comprensibile il tutto.

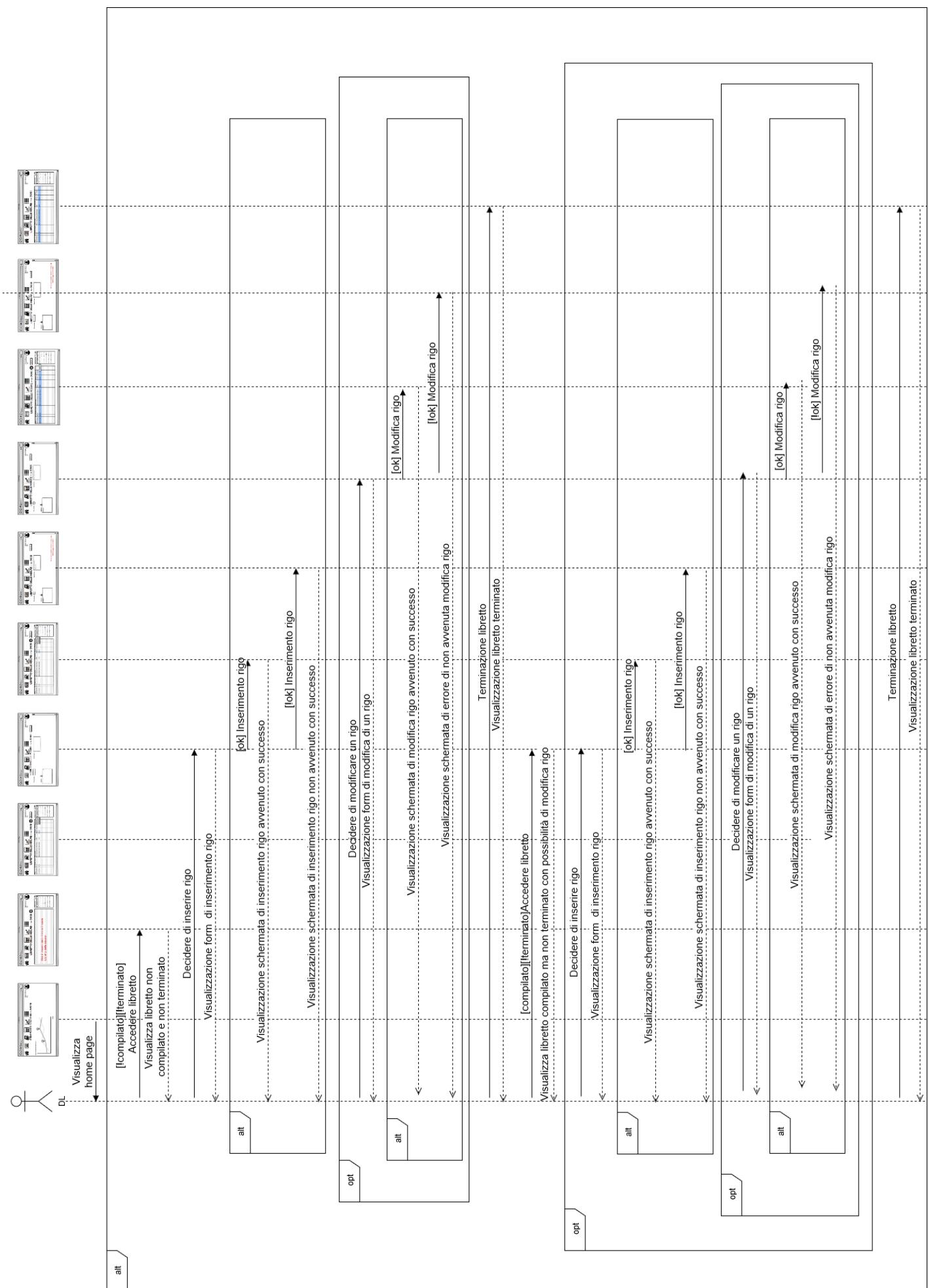
Nel primo, rappresentato nella figura sottostante, sono raffigurate richieste di accesso all'area riservata al libretto delle misure. In base all'attore chiamante e ad altre condizioni che riguardano invece il libretto in sé (quali la compilazione o la terminazione di esso), a partire dalla home page del sito, si è reindirizzati in diverse pagine.

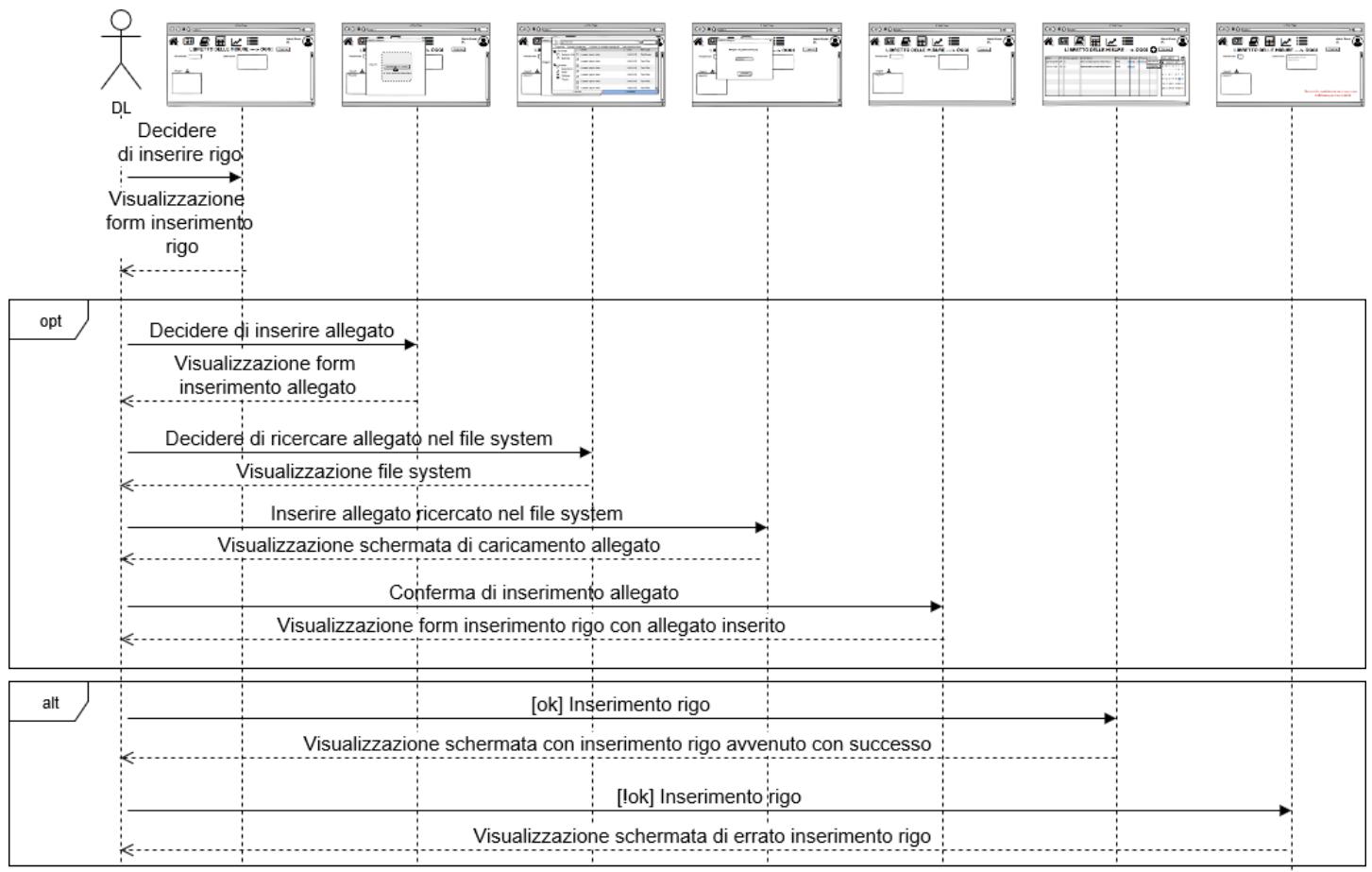


Nella seconda figura abbiamo invece raffigurato il flusso di lavoro nel caso in cui l'attore coinvolto sia il direttore dei lavori. Come è possibile vedere si hanno due rami alternativi, il primo nel caso in cui il libretto non sia ancora stato compilato, il secondo in cui il direttore acceda ad un libretto già inizializzato.

Nel primo caso quindi si ha un obbligo di compilazione di almeno un rigo del libretto, infatti come possiamo vedere, la funzione “Decisione di inserimento rigo” non è opzionale. Funzione che invece diventa opzionale nel secondo caso, ovvero quando abbiamo un libretto delle misure già compilato. Infatti, in questo caso, il DL può decidere di terminare il libretto senza andare a fare nuove aggiunte di righi.

Per ragioni semplificative, nel seguente diagramma non abbiamo esplicitato la funzione di inserimento e di sovrascrittura di un rigo, che andiamo a presentare però nel diagramma successivo.



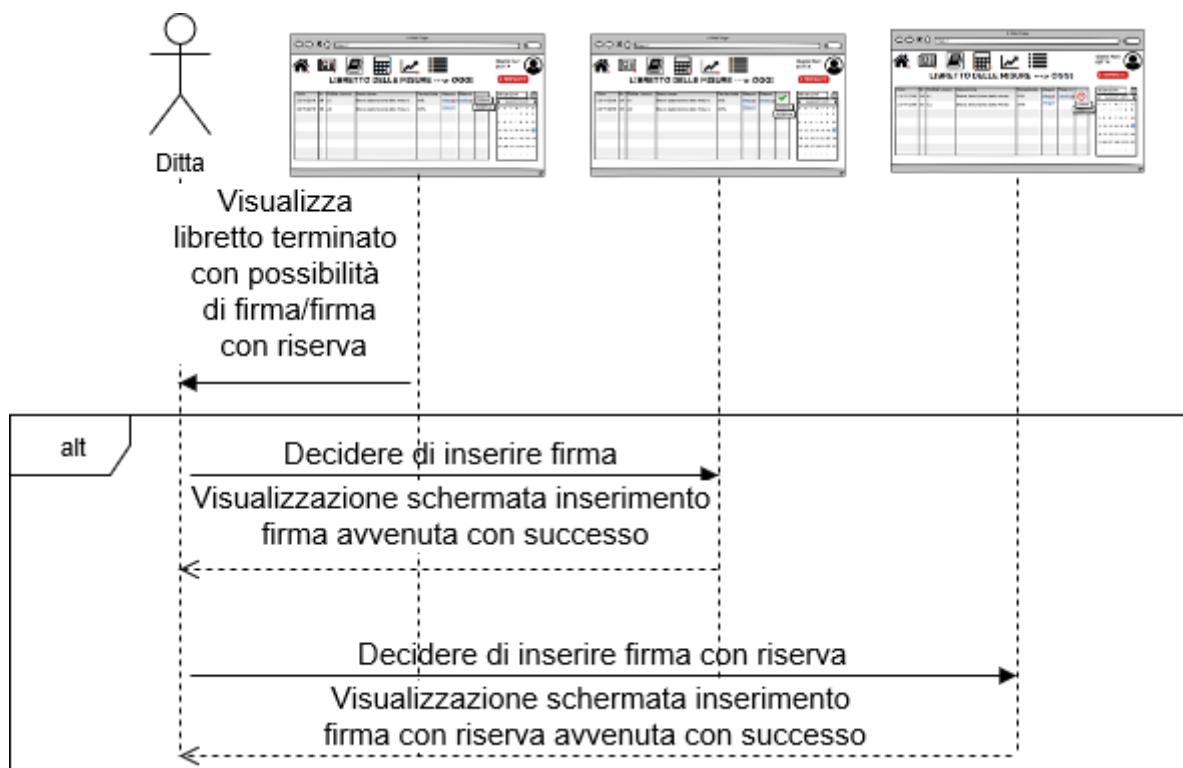


Dato che la funzione di sovrascrittura di un rigo ha uno stesso workflow della funzione sopra presentata, abbiamo pensato di non rappresentare in modo esplicito il dialogue model anche per la modifica di un rigo, ma di attenerci a quello sopra, con le dovute modifiche.

6.5.4. Story Card 4

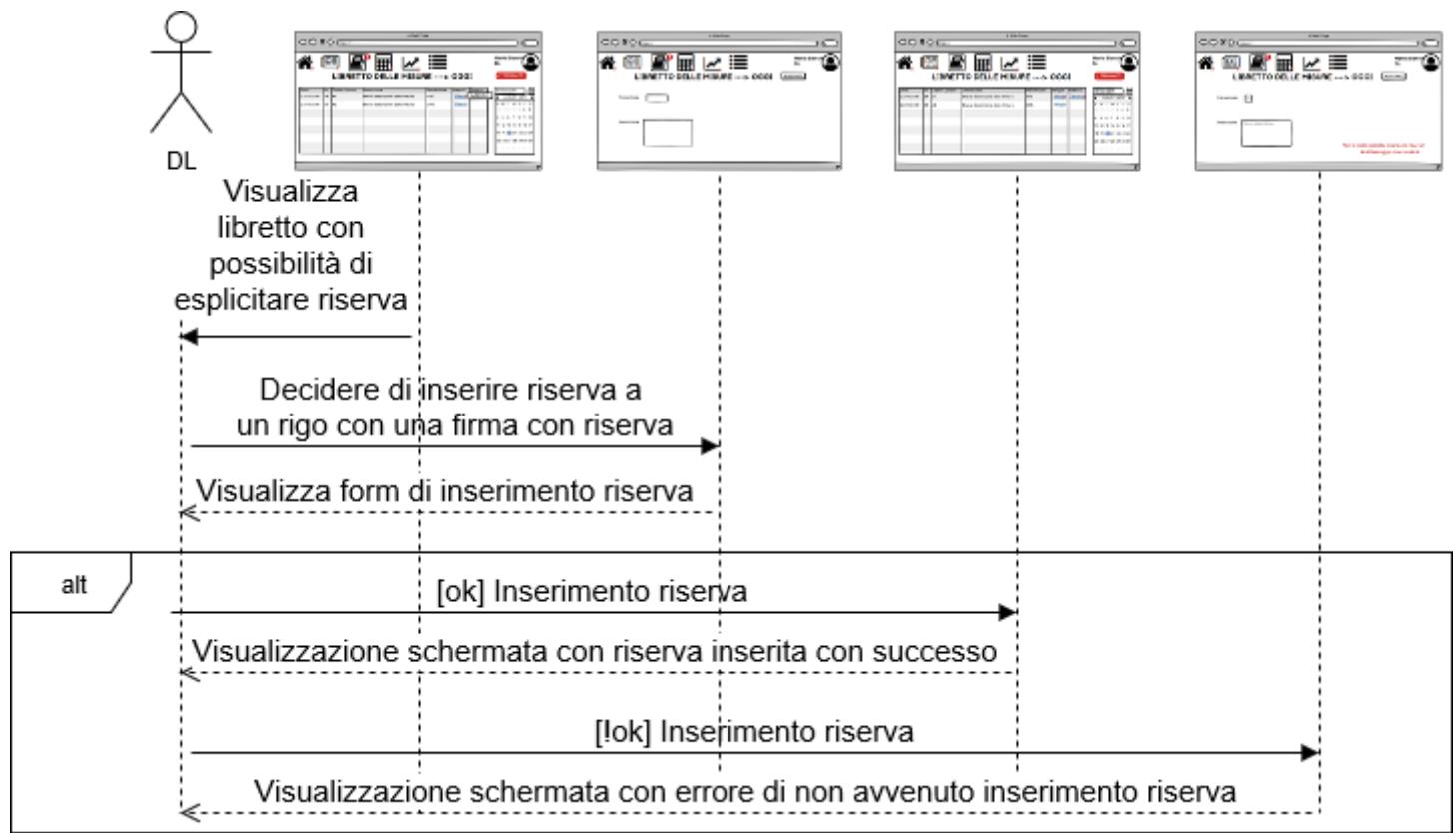
Flusso 1: ditta inserisce firma/firma con riserva

- Il libretto è terminato
- La ditta decide se apporre una firma o una firma con riserva ad ogni singolo rigo
- La firma o firma con riserva viene compilato



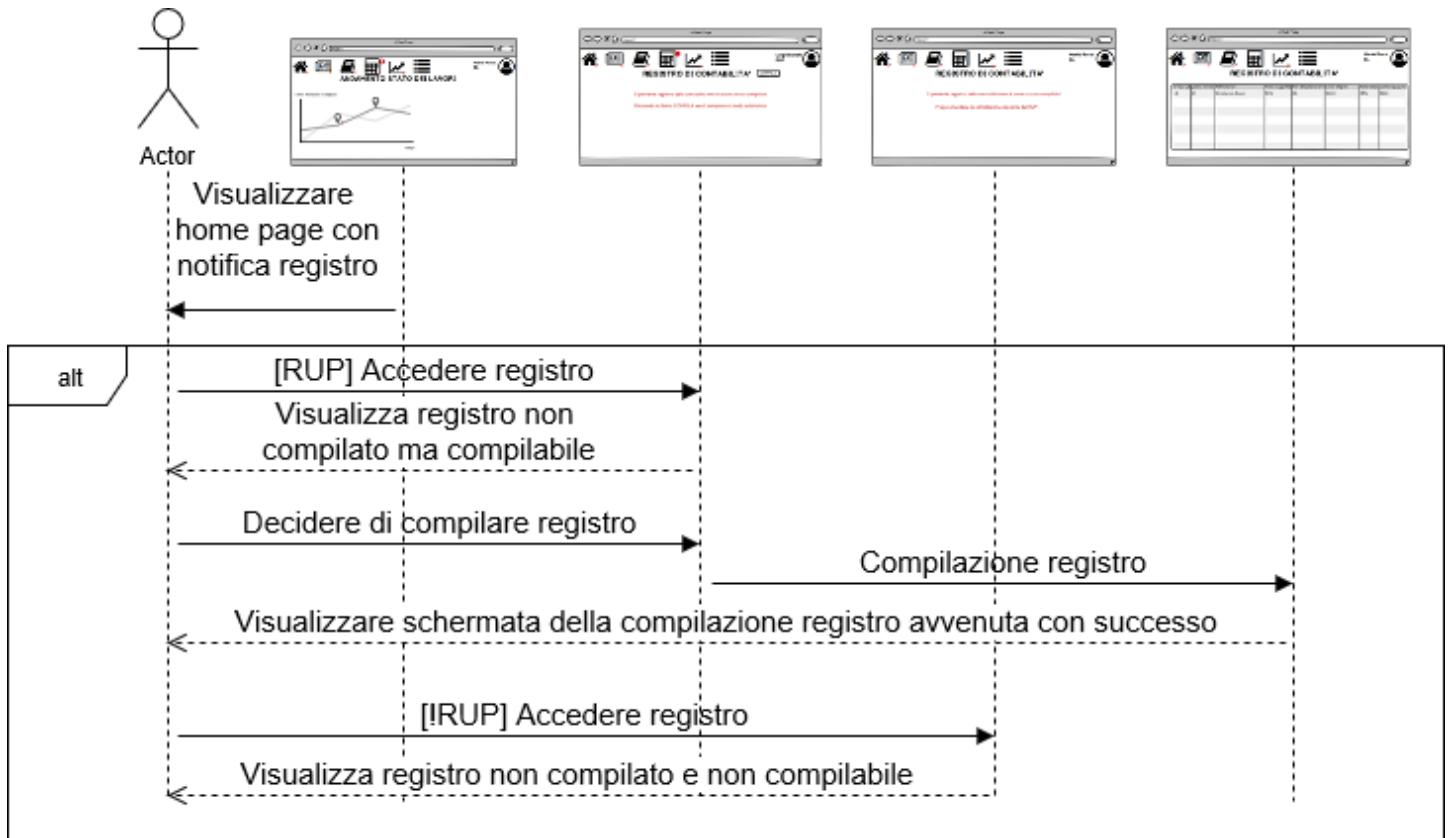
Flusso 2: direttore inserisce la riserva

- La ditta ha apposto la firma con riserva ad un rigo
- Il direttore esplicita la riserva

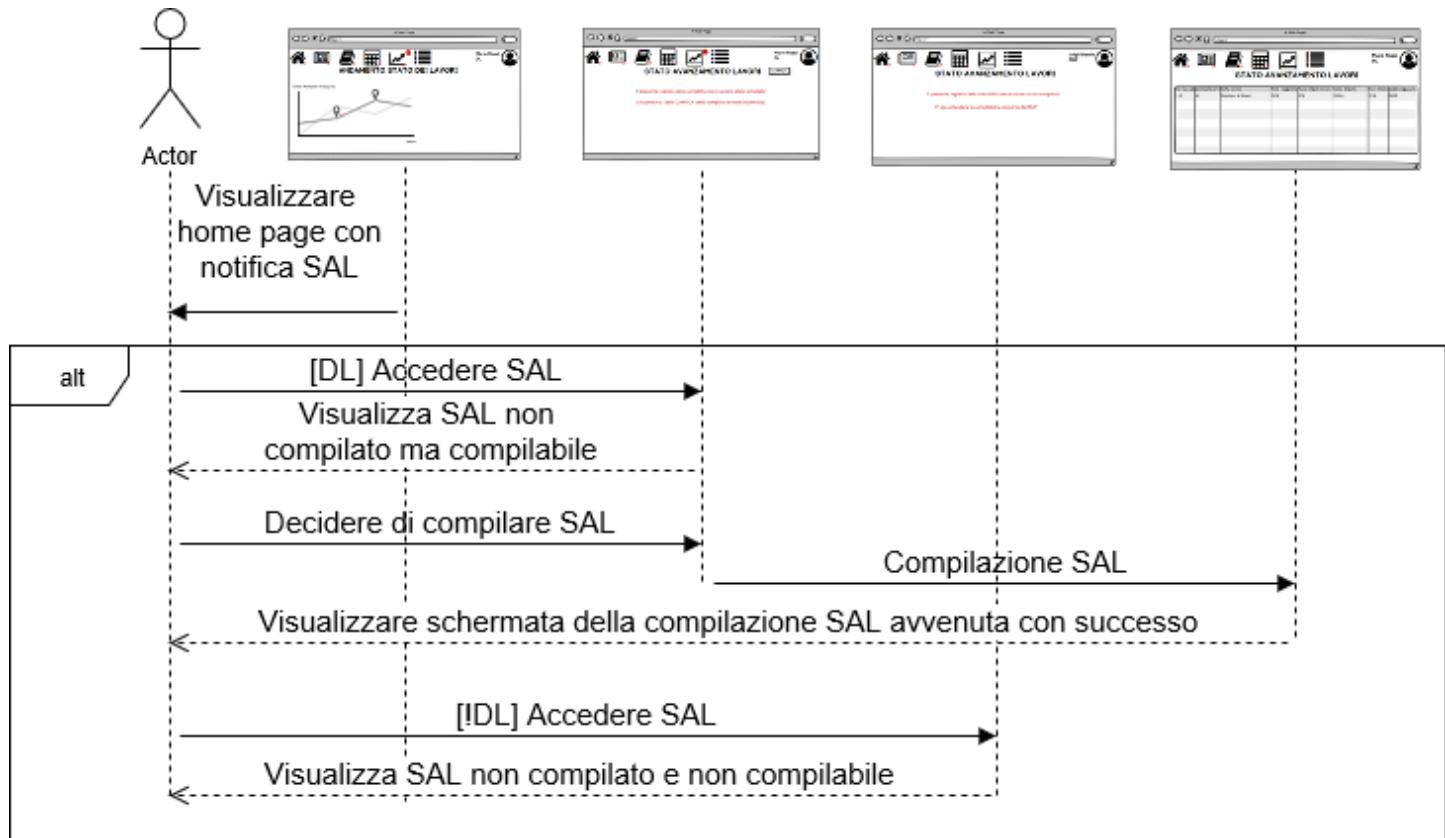


6.5.5. Story Card 5

Nel navigation model sottostante troviamo rappresentata la fase di compilazione di un registro della contabilità. Come è possibile vedere, a seconda dell'attore si hanno due rami alternativi. Nel primo il RUP che ha l'abilitazione all'inserimento del registro; nell'altro il direttore dei lavori o la ditta che invece possono solamente visualizzare il registro.



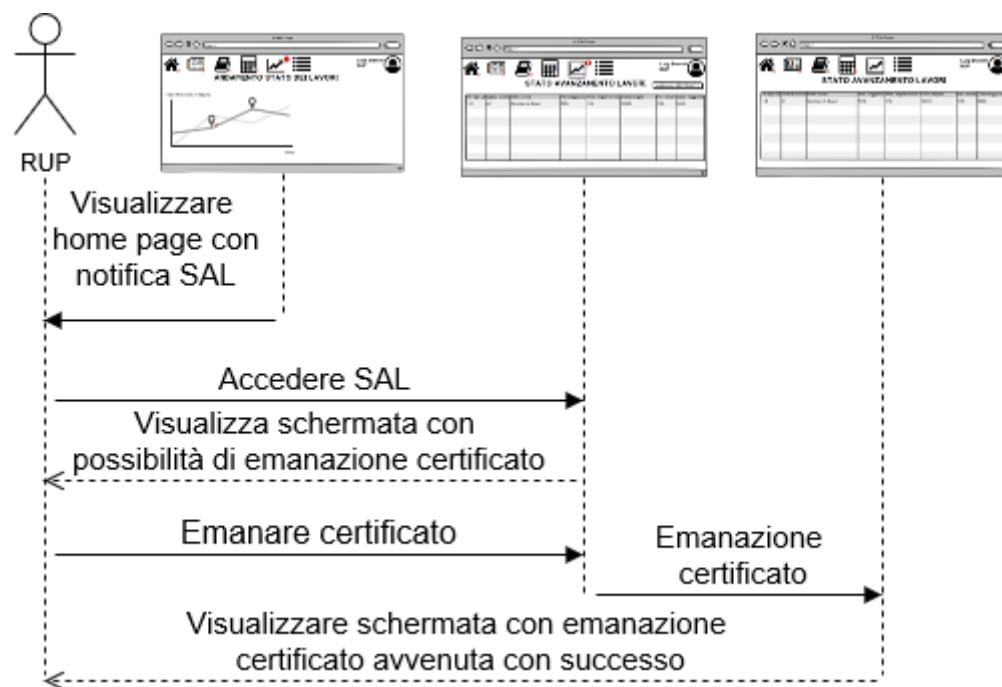
6.5.6. Story Card 6



6.5.7. Story Card 7

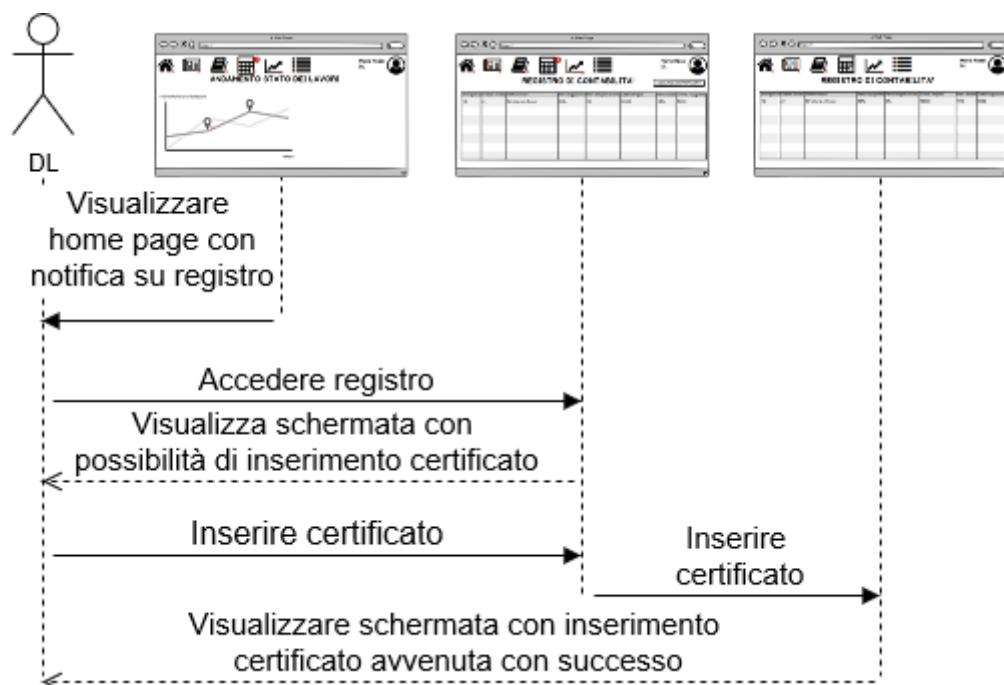
Flusso 1: il RUP emana un certificato di pagamento

- Il SAL è stato emanato
- Il RUP può emanare il certificato

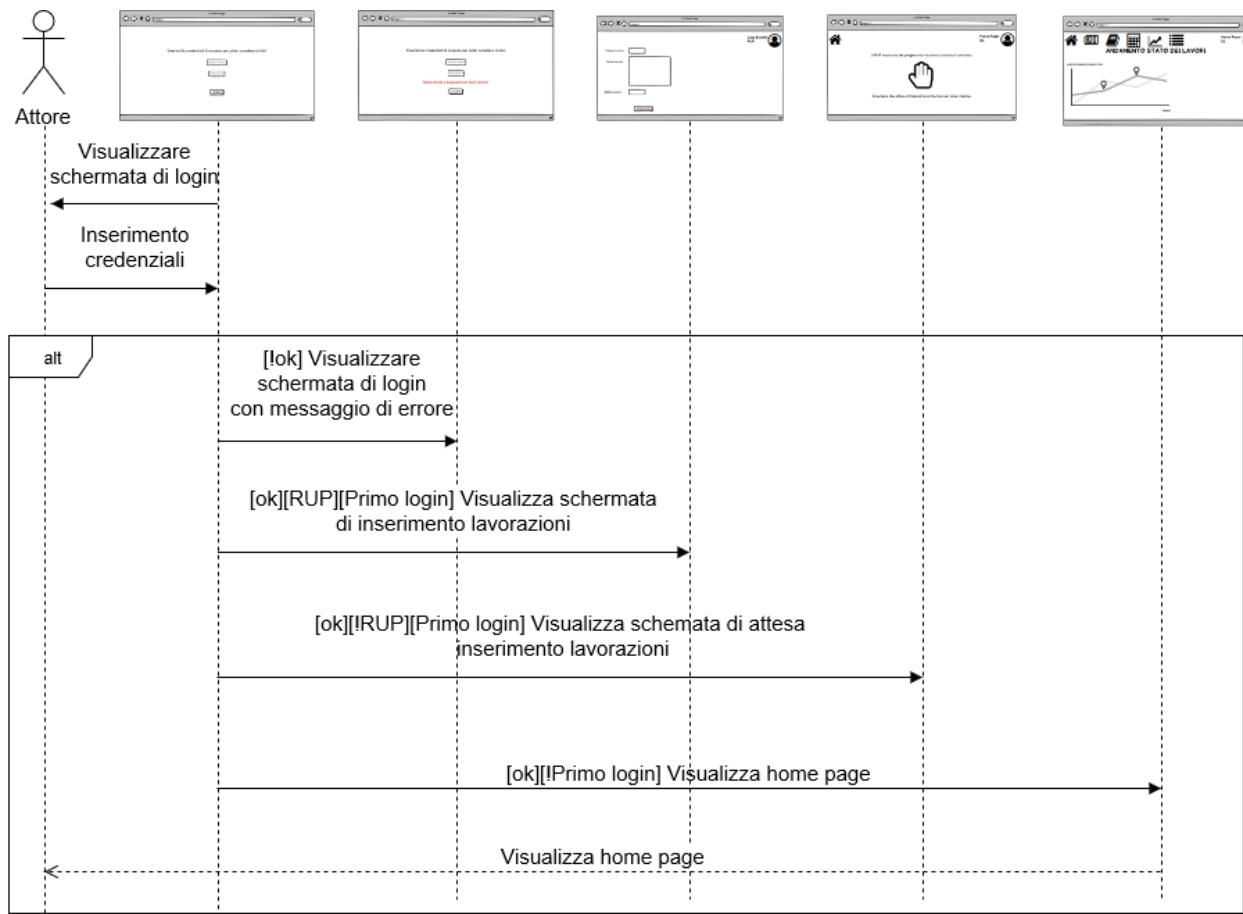


Flusso 2: il DL inserisce un certificato di pagamento

- Il certificato è stato messo dal RUP
- Il DL può inserire il certificato nel registro



6.5.8. Story Card 8



6.5.9. Story Card 9

Per quanto riguarda il dialogue model di questa story card, non abbiamo ritenuto essere fondamentale ad una maggior comprensione del funzionamento della richiesta di accesso alla lista delle lavorazioni e delle statistiche riguardanti la rata di acconto o della percentuale di completamento.

Per questo motivo si rimanda al navigation model della story card 9, che risulta essere già esplicativo in sé.

6.5.10. Story Card 10

Per quanto riguarda il dialogue model di questa story card, non abbiamo ritenuto essere fondamentale ad una maggior comprensione del funzionamento della richiesta di consultazione dell'andamento dei lavori.

Per questo motivo si rimanda al navigation model della story card 10, che risulta essere già esplicativo in sé.

7. Analisi orientata ai dati

7.1. Diagramma delle classi

Per ragioni di miglior comprensione delle varie classi, abbiamo preferito riportare innanzitutto le singole classi che abbiamo derivato dalle schede CRC e poi il diagramma delle classi (senza inserire attributi e funzioni della classe in modo da avere uno schema più facilmente comprensibile). In questo modo è possibile prima consultare i vari attributi e funzioni delle nostre classi e poi vedere le associazioni che ci sono tra esse.

Giornale dei lavori <<entity>>
<ul style="list-style-type: none">- data: uint- quantità_personale: uint[]- qualità_personale: string[]- descrizione: string- attrezzature: string[]- quantità_attrezzature: uint[]- sovrascritto: uint
<ul style="list-style-type: none">+ visualizzaGiornale(numero_giornale: uint)+ inserisciRigo(descrizione: string, quantità_personale: uint[], qualità_personale: string[], ore_personale: uint[], attrezzature: string[], quantità_attrezzature: uint[])+ modificaRigo(numero_rigo: uint, numero_giornale: uint, descrizione: string, quantità_personale: uint[], qualità_personale: uint[], ore_personale: uint[], attrezzature: string[], quantità_attrezzature: uint[])+ terminaGiornale()- getGiornaleData (data: uint)

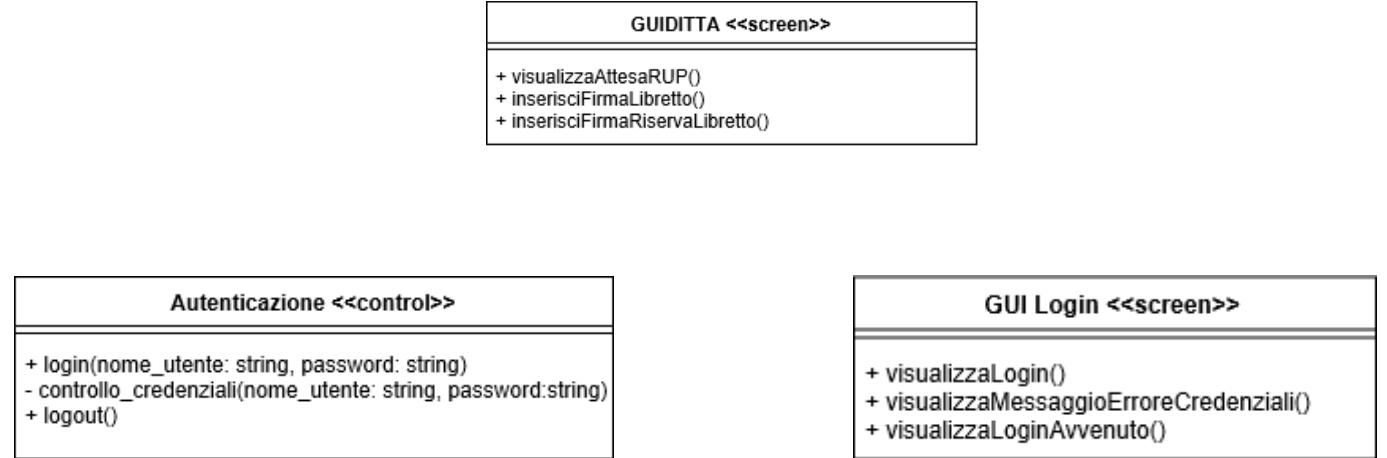
Libretto delle misure <<entity>>
<ul style="list-style-type: none">- data: uint- codice_lavoro: string- nome_lavoro: string- descrizione: string- percentuale: uint- allegato: string[]- sovrascritto: uint- percentuale_riserva: uint- descrizione_riserva: string
<ul style="list-style-type: none">+ visualizzaLibretto(numero_libretto: uint)+ inserisciRigo(codice_lavoro: string, descrizione: string, percentuale: uint, allegato: string[])+ modificaRigo(codice_lavoro: string, descrizione: string, percentuale: uint, allegato: string[])+ terminaLibretto()- getLibrettoData()- getLibrettoData(data: uint)- controlloRigo(numero_libretto: uint, numero_rigo: uint)- controlloLibretto(numero_libretto)+ inserisciRiserva(numero_libretto: uint, numero_rigo: uint, codice_lavoro: string, percentuale_riserva: uint, descrizione_riserva: string)+ visualizzaRiserva(numero_libretto: uint, numero_rigo: uint)- controllaRiserve(numero_libretto: uint)

Registro di contabilità <<entity>>	Stato di avanzamento dei lavori <<entity>>
<ul style="list-style-type: none"> - data: uint - numero_libretto: uint - rigo_libretto: uint - codice_lavoro: string - nome_lavoro: string - percentuale_raggiunta: uint - percentuale_sul_totale: uint - percentuale_singola_lavorazione: uint - costo_singola_lavorazione: uint - costo_maturato: uint <ul style="list-style-type: none"> + visualizzaRegistro(numero_registro: uint) + abilitaRegistro() - inserisciRigoRegistro(numero_libretto: uint, rigo_libretto: uint) - contaCostoRegistro(numero_registro: uint) + controllaSoglia() - controllaSuperamento(costo_registro: uint, n_soglia: uint) + visualizzaCertificatiRegistro(numero_registro: uint) - getRegistroData() - getRegistroData(data: uint) 	<ul style="list-style-type: none"> - data: uint - codice_lavoro: string - nome_lavoro: string - percentuale_raggiunta: uint - percentuale_sul_totale: uint - percentuale_singola_lavorazione: uint - costo_singola_lavorazione: uint - costo_maturato: uint <ul style="list-style-type: none"> + visualizzaSAL(numero_sal: uint) + abilitaSAL() - inserisciRigo(codice_lavoro: string, nome_lavoro: string, percentuale_raggiunta: uint, percentuale_sul_totale: uint, percentuale_singola_lavorazione: uint, costo_singola_lavorazione: uint, costo_maturato: uint) - trovaRighiRegistri() - getSALData() - getSALData(data: uint) - calcolaValoreSAL()

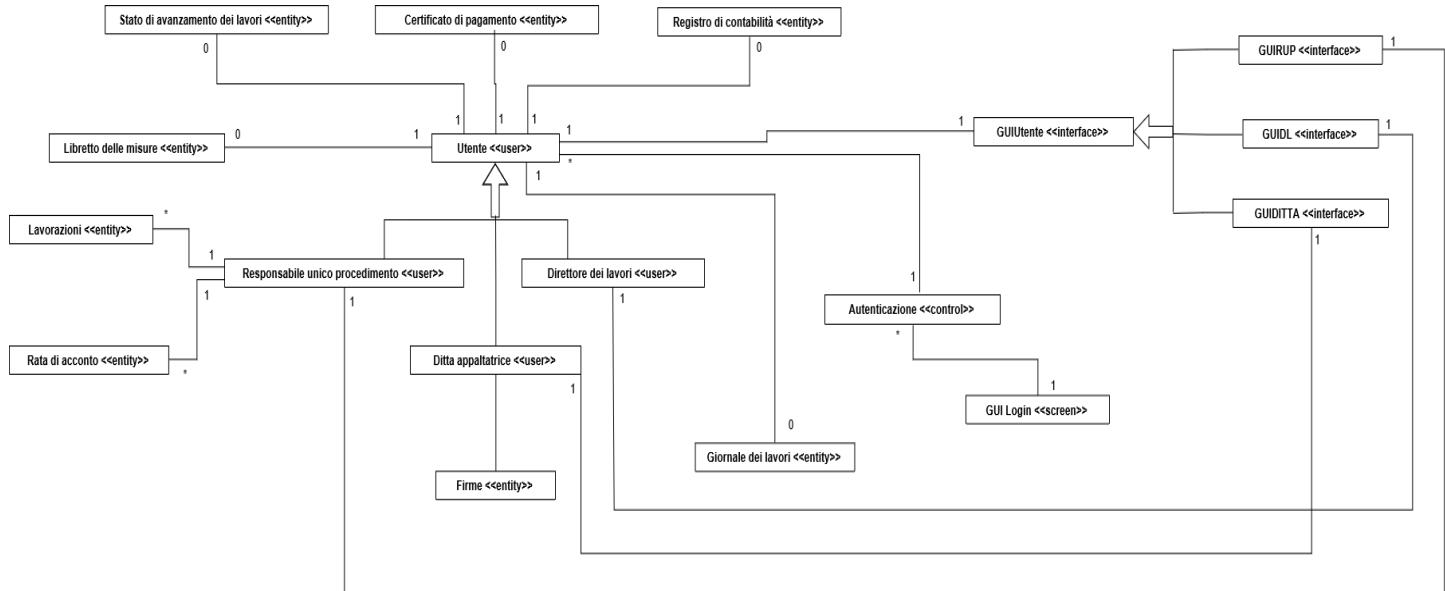
Certificato di pagamento <<entity>>
<ul style="list-style-type: none"> - data: uint - importo: uint - numero_sal: uint - numero_registro: uint <ul style="list-style-type: none"> + emanaCertificato(data: uint, importo: uint, sal: uint) + inserisciCertificato(registro: uint) - calcolaValoreCertificati() - controlloCertificato()

Firme <<entity>>	Lavorazioni <<entity>>
<ul style="list-style-type: none"> - hash_transazione: string - firmatario: address - numero_documento: uint - rigo_documento: uint - riserva: bool <ul style="list-style-type: none"> + inserisciFirmaDittaRigoLibretto() + inserisciFirmaRiservaDittaRigoLibretto() - inserisciFirma(hash_transazione: string, firmatario: address, numero_documento: uint, numero_rigo: uint) - inserisciFirma(hash_transazione: string, firmatario: address, numero_documento: uint, numero_rigo: uint, riserva: bool) - controlloHash(hash_transazione: string) + visualizzaFirme(numero_libretto: uint) 	<ul style="list-style-type: none"> - codice_lavoro: string - nome_lavoro: string - costo_lavoro: uint <ul style="list-style-type: none"> + inserisciLavorazione(codice_lavoro: string, nome_lavoro: string, costo_lavoro: uint) + modificaLavorazione(codice_lavoro: string, nome_lavoro: string, costo_lavoro: uint) + terminaLavorazioni() + visualizzaLavorazioni()

<p>GUI Firme <<screen>></p> <pre>+ visualizzaBottoneInserimentoFirma() + visualizzaBottoneInserimentoFirmaRiserva() + decidilInserireFirma() + decidilInserireFirmaRiserva() + visualizzalnserimentoFirmaAvvenuta() + visualizzalnserimentoFirmaRiservaAvvenuta() + visualizzaRiserveApposte()</pre>	<p>Rata di acconto <<entity>></p> <pre>- valore_monetario: uint - numero_rata: uint - superamento: enum {superata, non_superata, attesa} + inserisciRataAcconto(valore_monetario: uint) + modificaRataAcconto(valore_monetario: uint) + terminaRataAcconto() + visualizzaRataAcconto() + settaRataSuperata() + controllaRataSuperata()</pre>
<p>Utente <<user>></p> <pre>- nome: string - cognome: string - nome_utente: string - password: string - ruolo: string + login(nome_utente: string, password: string) + logout()</pre>	<p>Responsabile unico procedimento <<user>></p> <pre>- ruolo: string</pre> <p>Direttore dei lavori <<user>></p> <pre>- ruolo: string</pre> <p>Ditta appaltatrice <<user>></p> <pre>- ruolo: string</pre>
<p>GUIDL <<screen>></p> <pre>+ visualizzaAttesaRUP() + decidilInserireRigoGiornale() + inserisciRigoGiornale(descrizione: string, quantità_personale: uint[], qualità_personale: string[], ore_personale: uint[], attrezzature: string[], quantità_attrezzature: uint[]) + decidilModificareRigoGiornale() + modificaRigoGiornale(numero_giornale: uint, rigo_giornale : uint, descrizione: string, quantità_personale: uint[], qualità_personale: string[], ore_personale: uint[], attrezzature: string[], quantità_attrezzature: uint[]) + terminaGiornale() + decidilInserireRigoLibretto() + inserisciRigoLibretto(codice_lavoro: string, descrizione: string, percentuale: uint, allegato: string[]) + decidilInserireRigoLibretto() + modificaRigoLibretto(numero_libretto: uint, rigo_libretto: uint, codice_lavoro: string, descrizione: string, percentuale: uint, allegato: string[]) + terminaLibretto() + abilitaSAL() - inserisciRigoSAL() + terminaSAL() + inserisciCertificatoRegistro()</pre>	<p>GUIRUP <<screen>></p> <pre>+ inserisciLavorazioni(codice_lavoro: string, nome_lavoro: string, costo_lavoro: uint) + decidiModificaLavorazione() + modificaLavorazione(codice_lavoro: string, nome_lavoro: string, costo_lavoro: uint) + terminalInserimentoLavorazioni() + inserisciRataAcconto(valore_monetario: uint) + decidiModificaRataAcconto() + modificaRataAcconto(valore_monetario: uint) + terminalInserimentoRataAcconto() + terminalInserimenti() + abilitaRegistro() + decidiEmanazioneCertificato() + emanaCertificato() + inserisciCertificato()</pre>



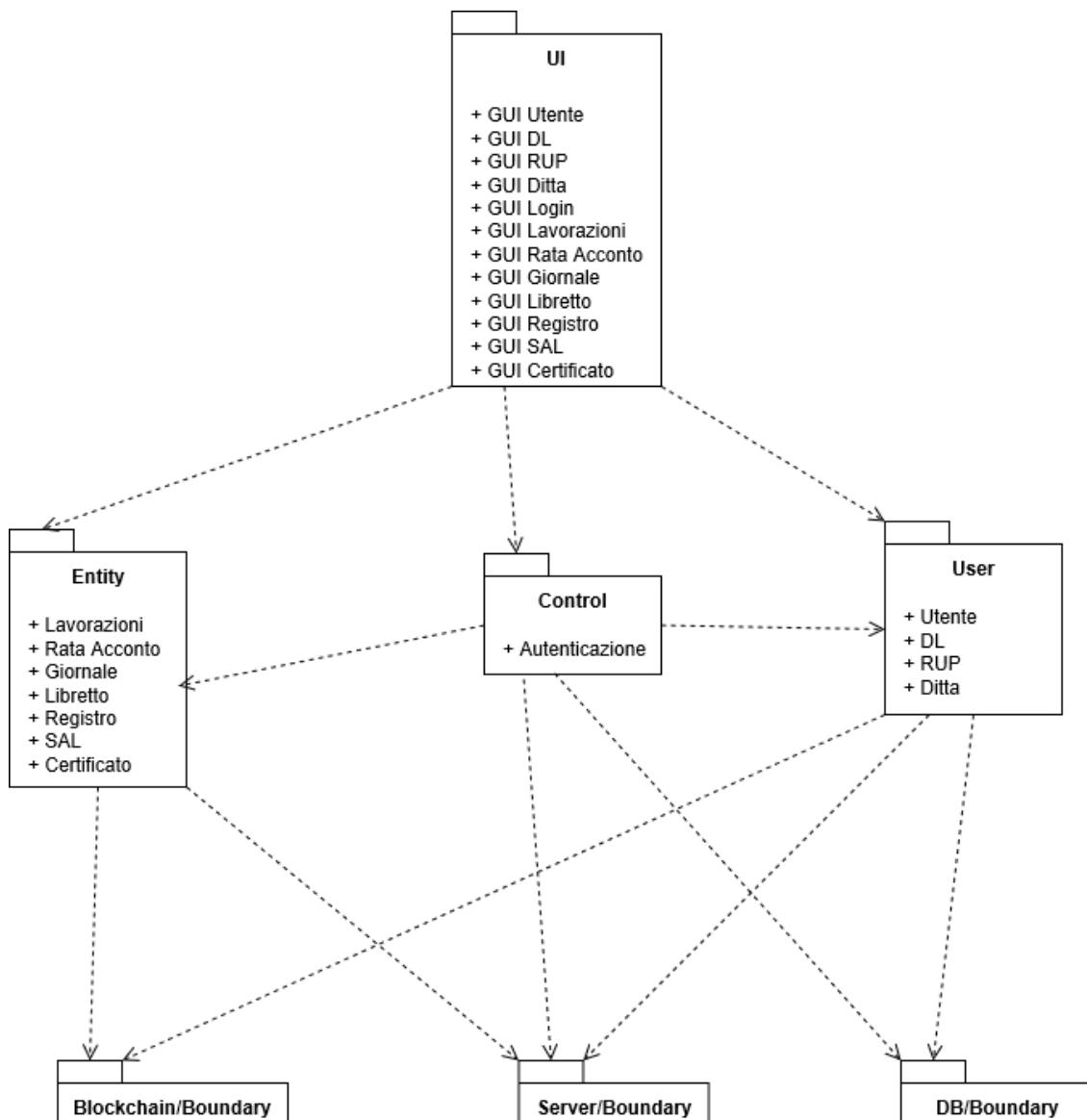
Nel diagramma delle classi non siamo andati a riportare gli attributi e le funzioni delle varie classi in quanto il modello complessivo sarebbe stato troppo dispersivo. Per questo motivo abbiamo preferito rappresentare una panoramica più semplificata delle relazioni tra le classi.



7.2. Diagramma dei packages

Per quanto riguarda il diagramma dei packages siamo riusciti ad individuare 3 differenti package:

- **UI**: contiene le classi associate ai mockup realizzati in precedenza nei Presentation Model. I metodi delle classi corrisponderanno quindi ai task evidenziati nel Task Model associati alle schede CRC.
- **Entity**: contiene le classi associate alle schede CRC che descrivono informazioni persistenti.
- **Control**: contiene la classe di autenticazione per l'accesso a specifici task.
- **Boundary**: contiene le classi associate alle schede CRC relative a sistemi esterni.

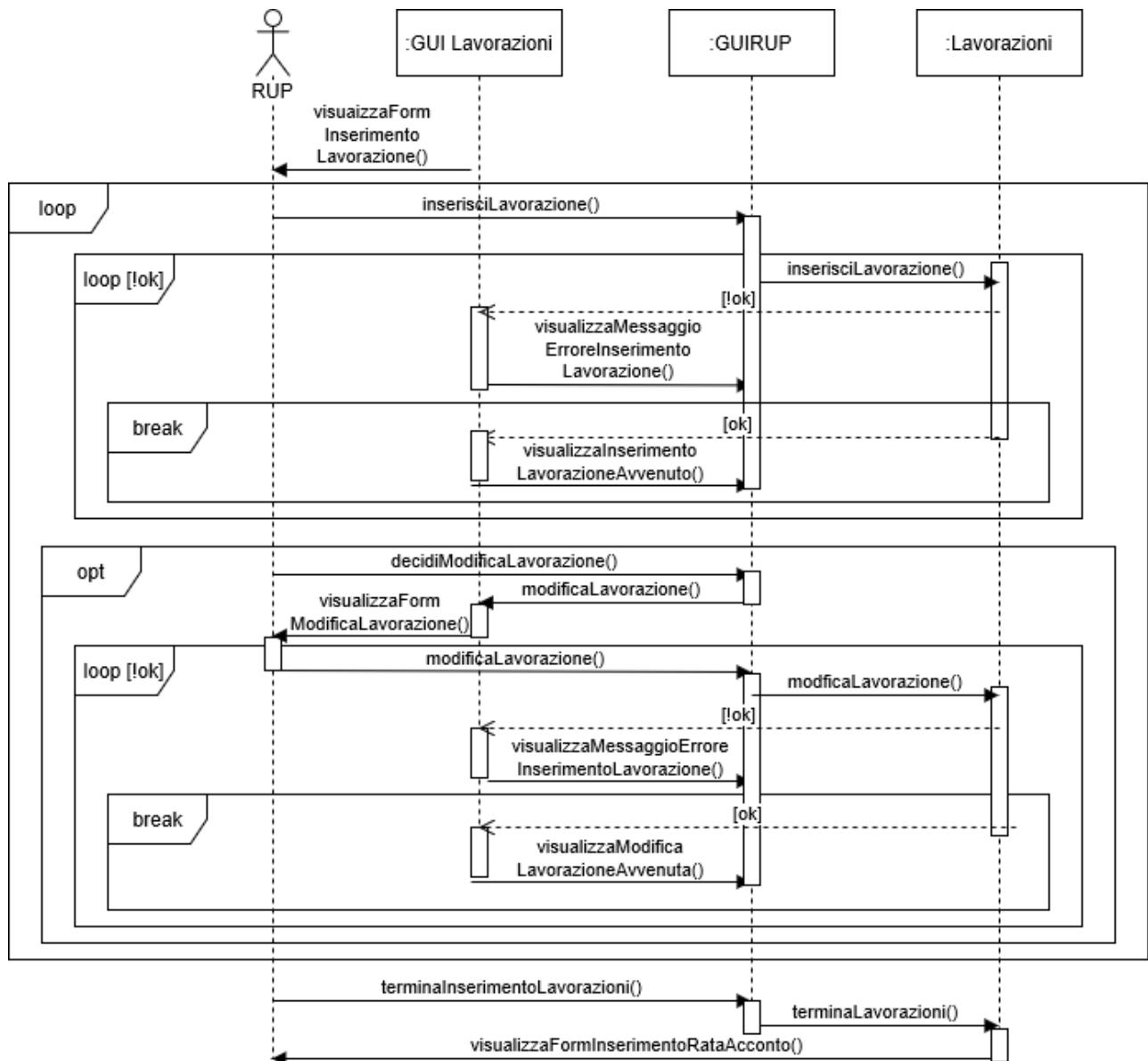


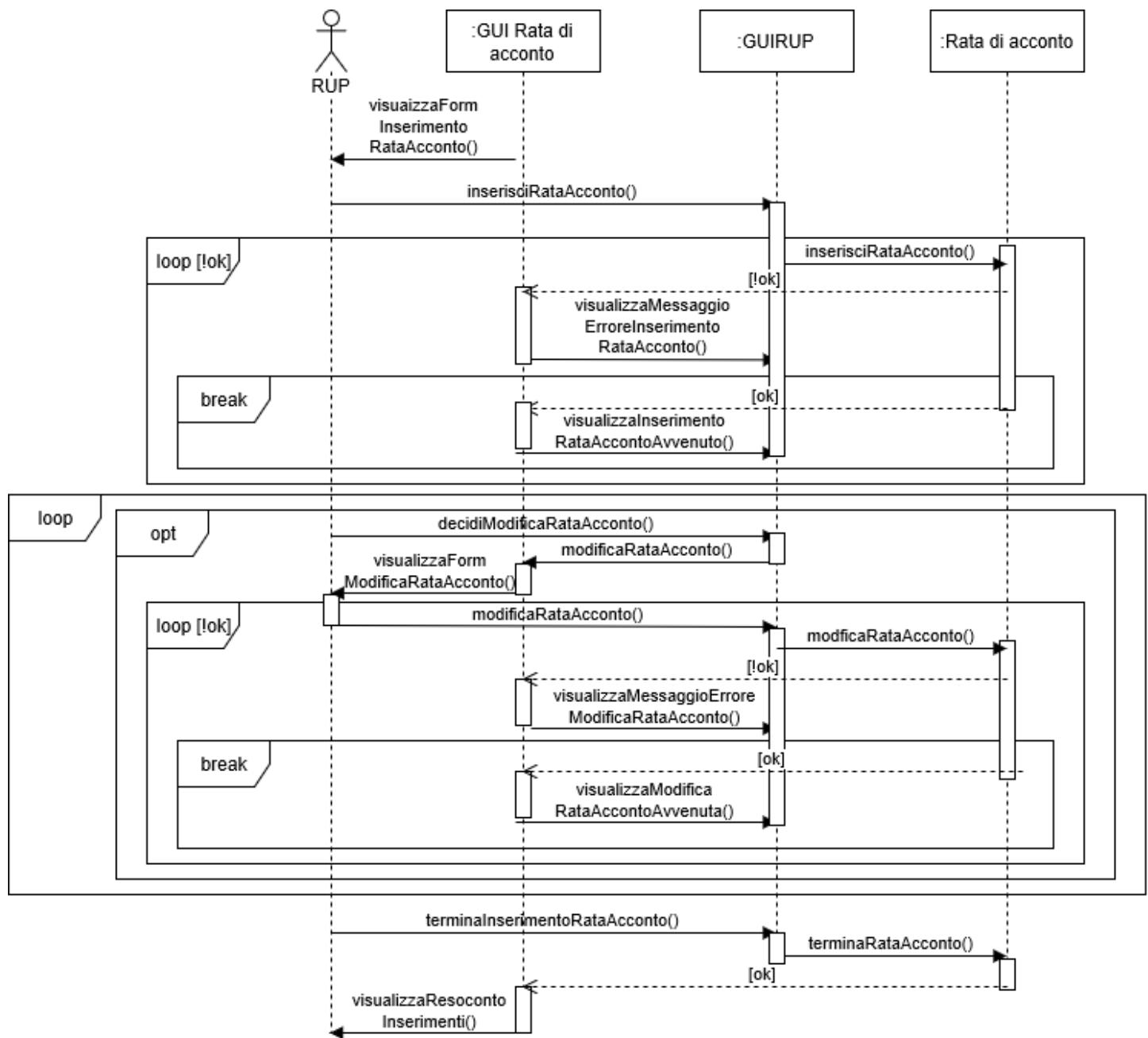
8. Analisi orientata ai comportamenti

Per quanto riguarda i diagrammi dei comportamenti siamo andati a realizzare i diagrammi delle sequenze e i diagrammi di stato.

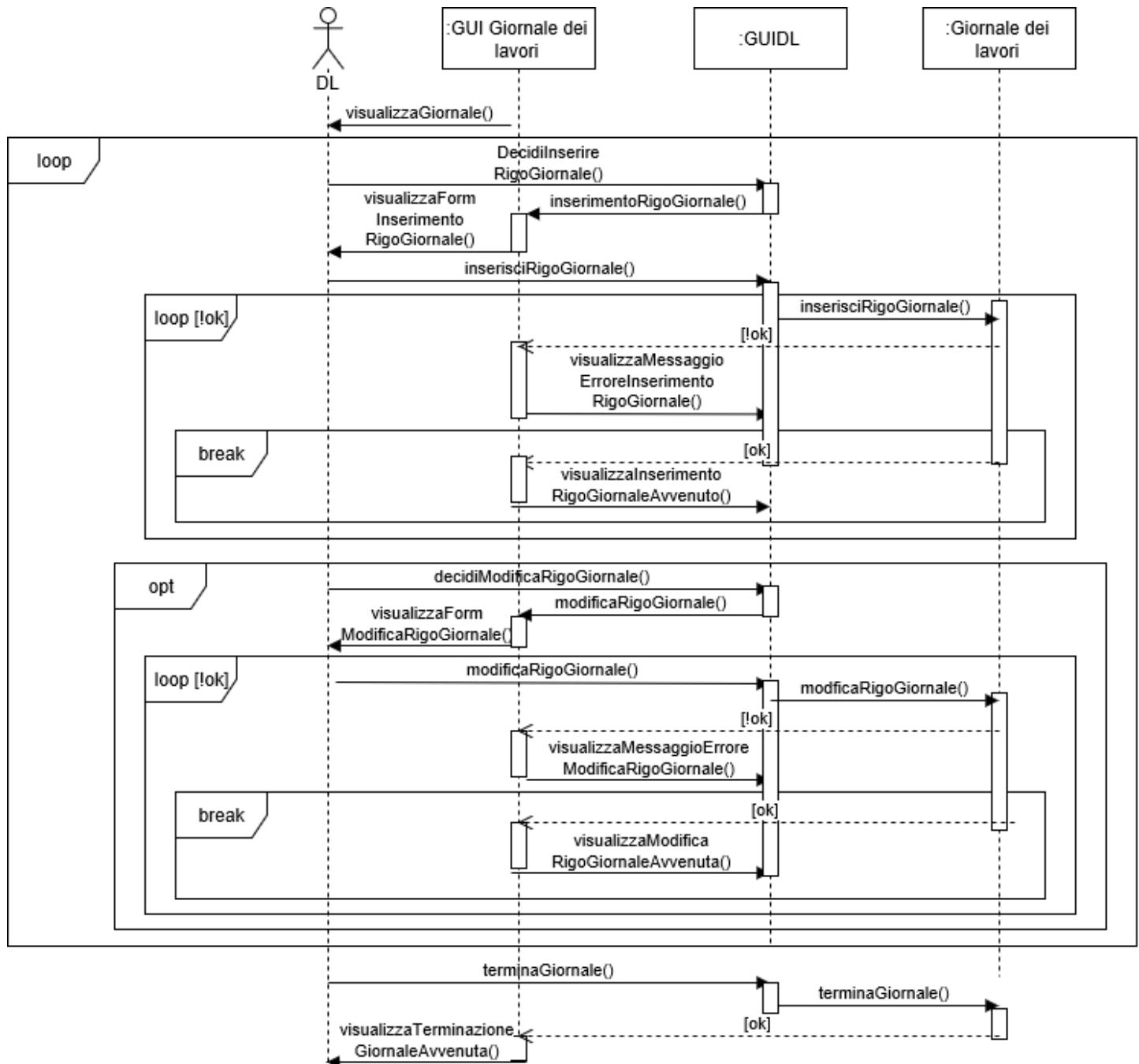
8.1. Diagramma delle sequenze

8.1.1. Story Card 1

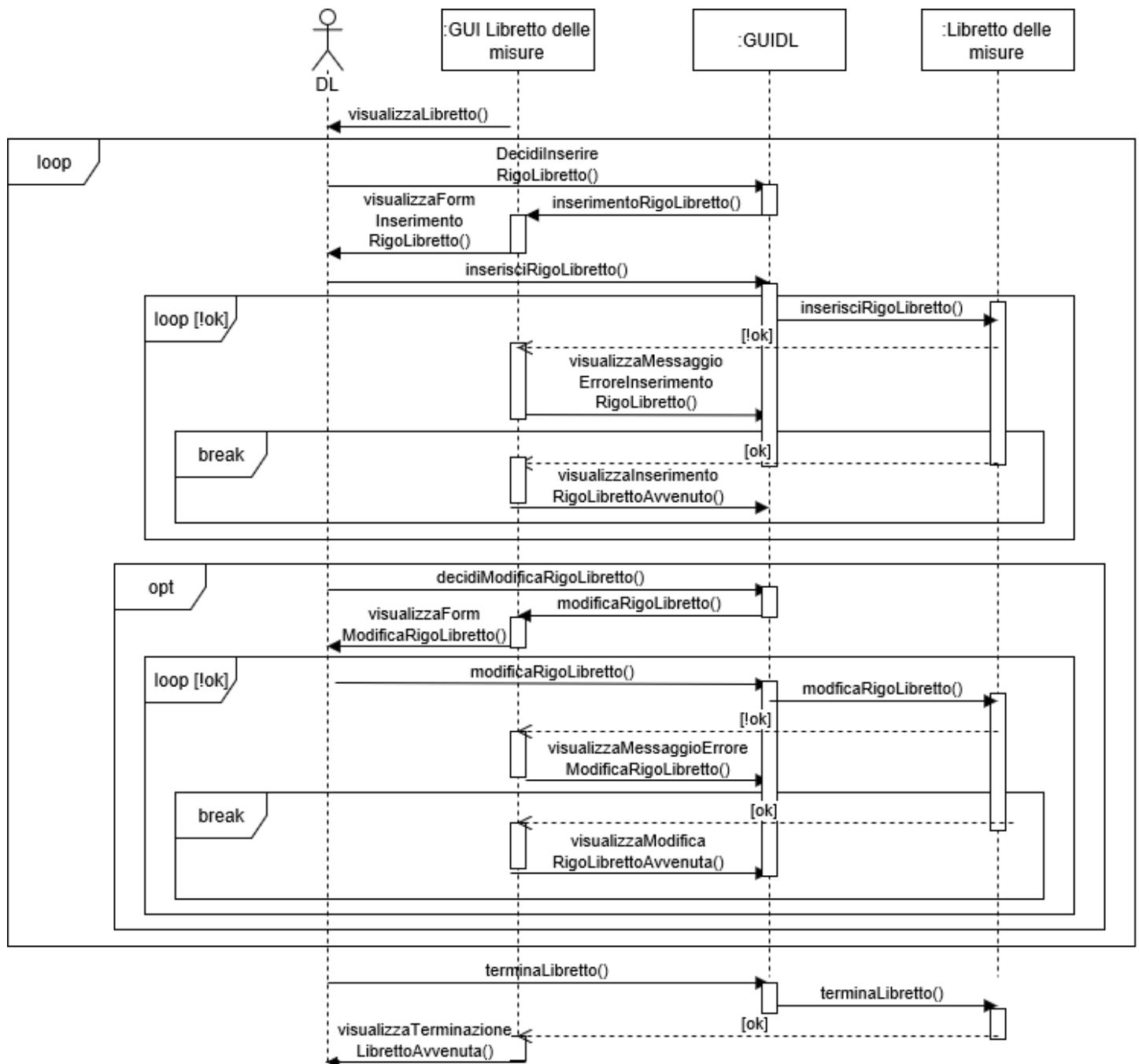




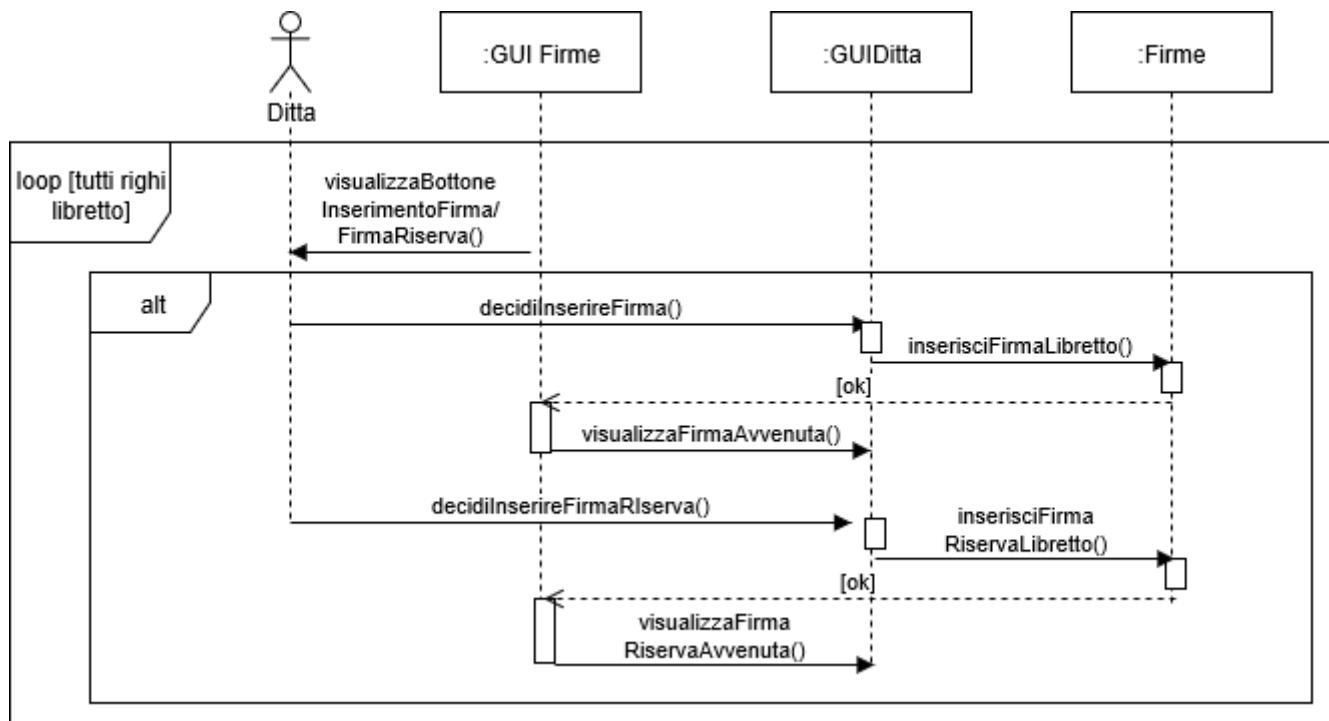
8.1.2. Story Card 2



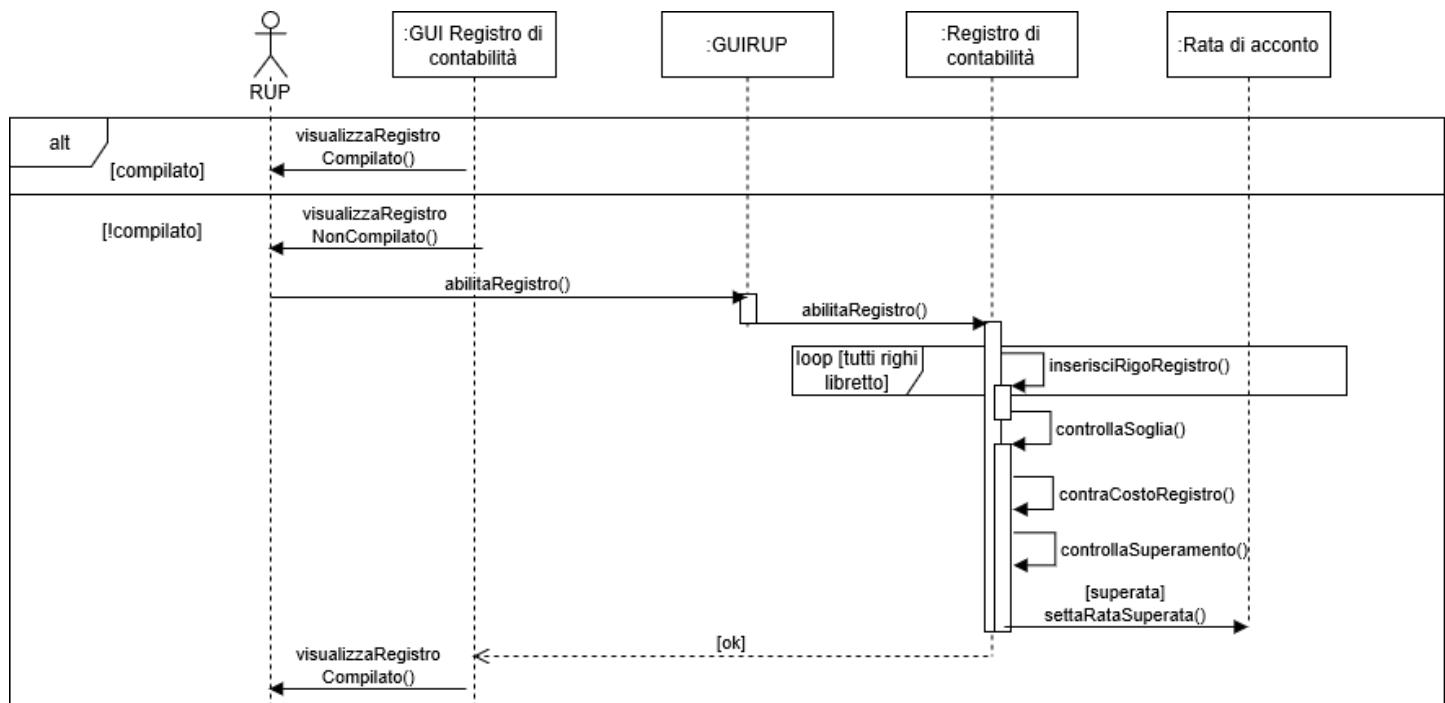
8.1.3. Story Card 3



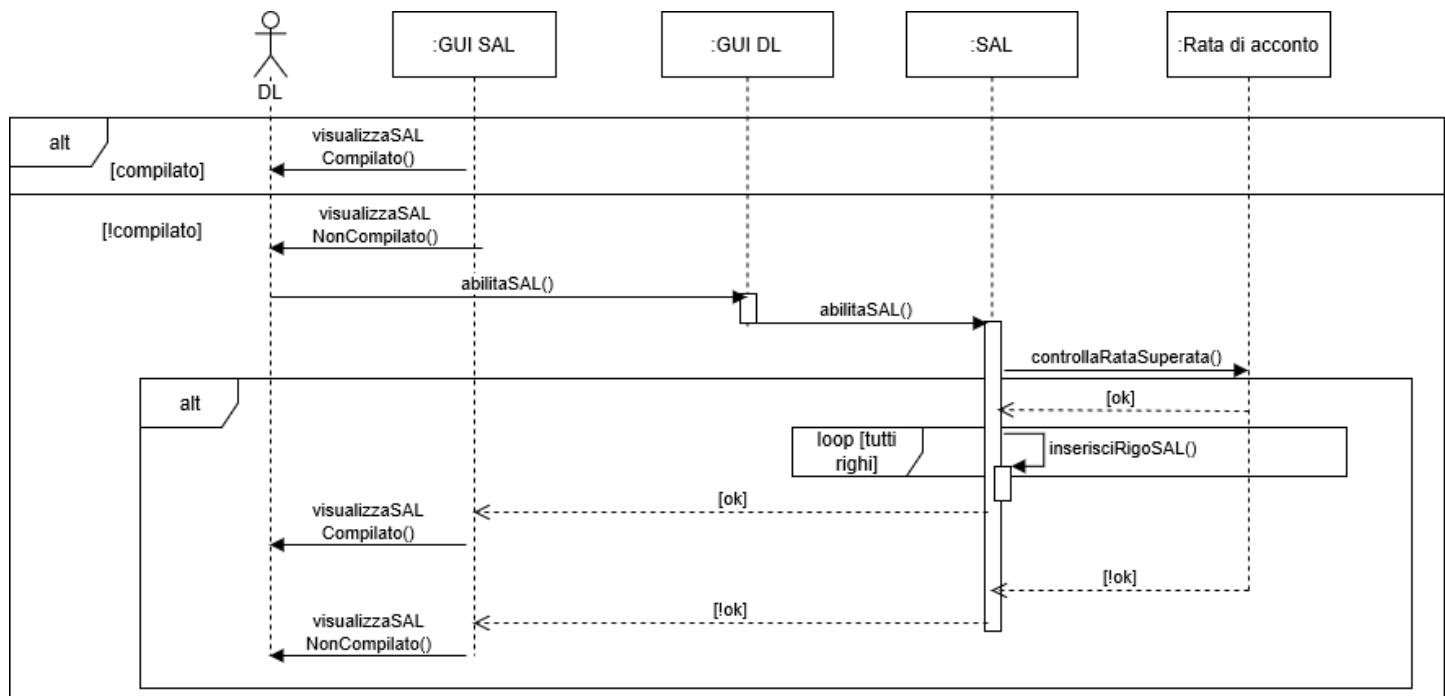
8.1.4. Story Card 4



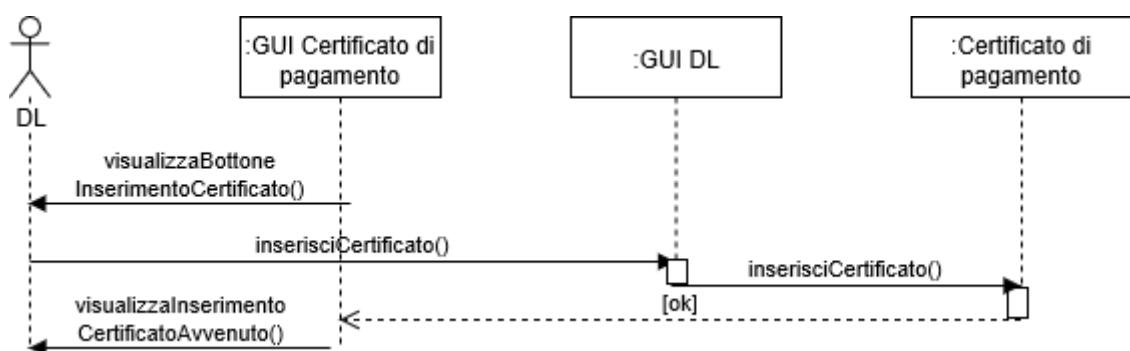
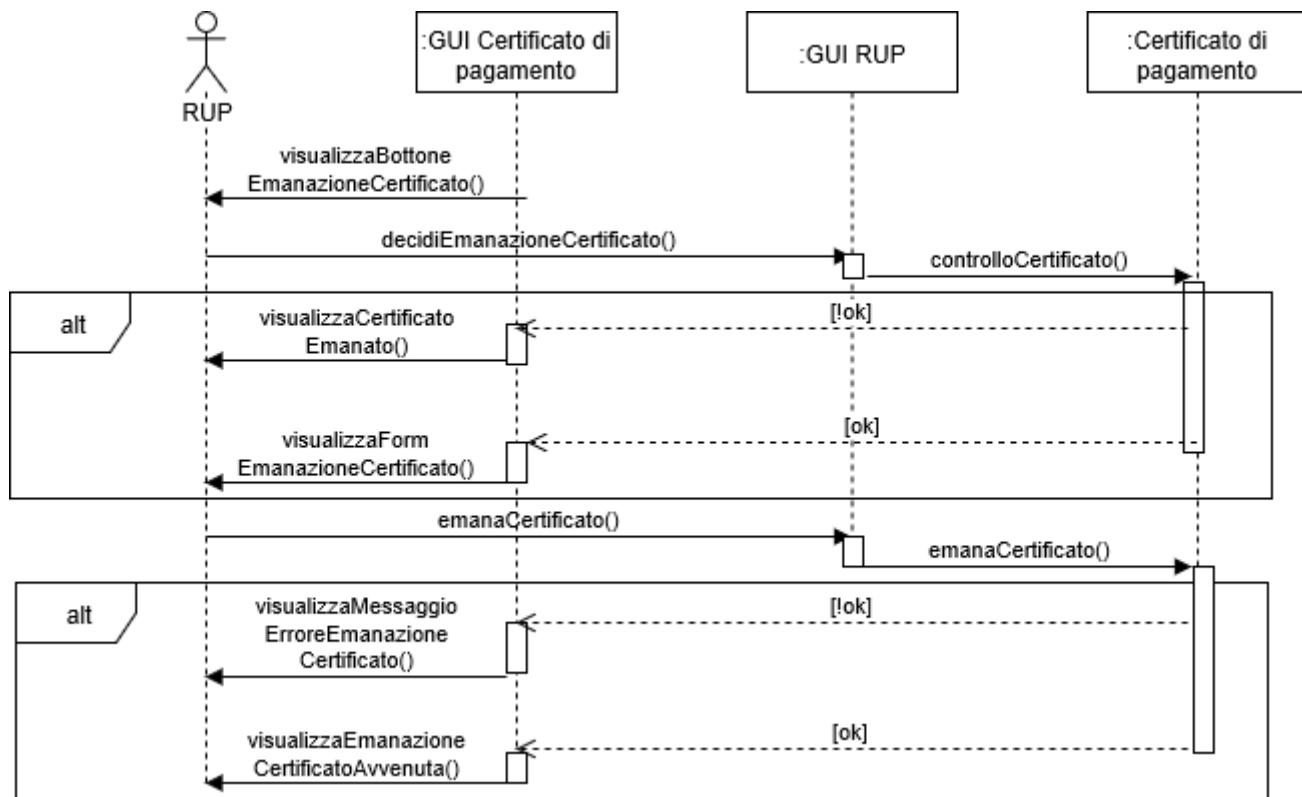
8.1.5. Story Card 5



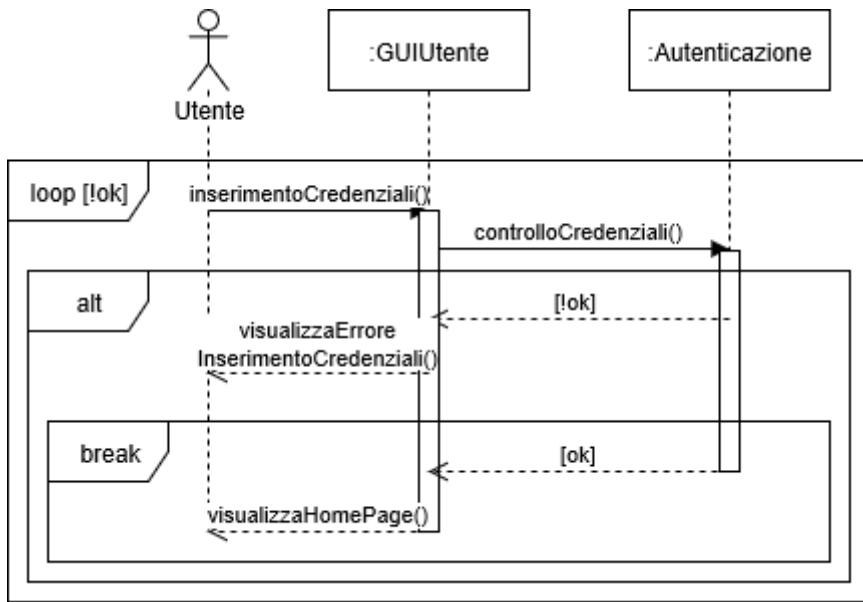
8.1.6. Story Card 6



8.1.7. Story Card 7



8.1.8. Story Card 8



8.1.9. Story Card 9

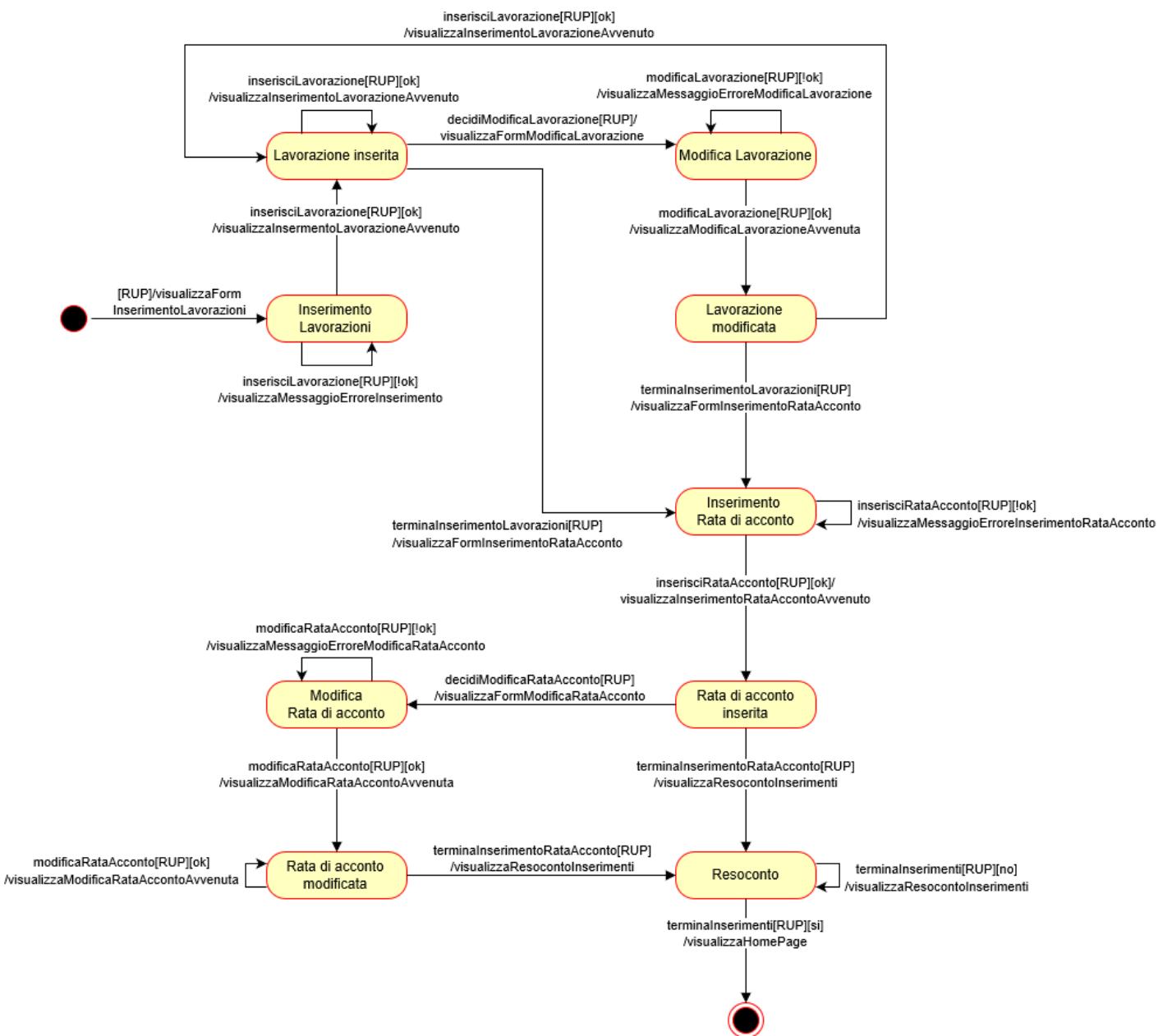
Per quanto riguarda la seguente storia utente non abbiamo ritenuto fosse necessario un diagramma delle sequenze, in quanto consiste in un'unica chiamata a funzione che restituisce la lista delle lavorazioni.

8.1.10. Story Card 10

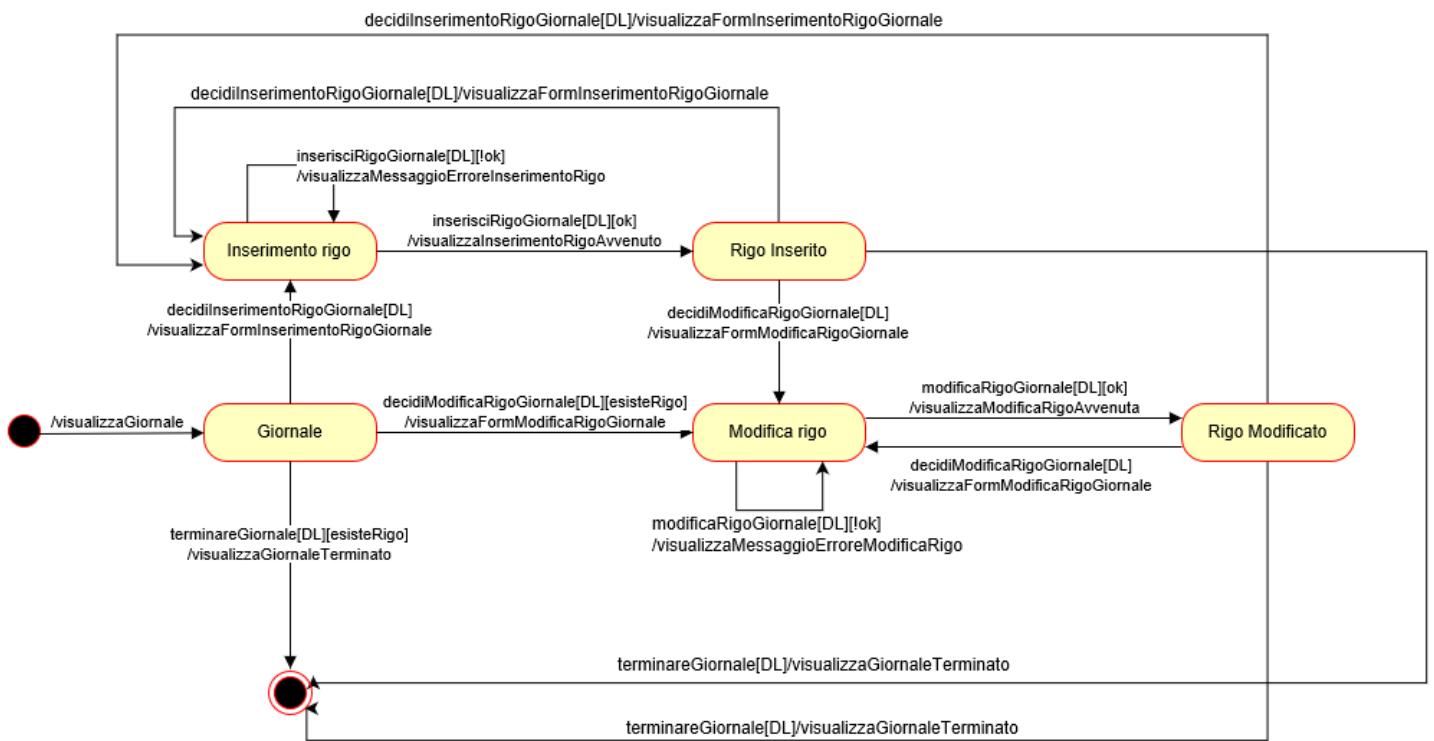
Per quanto riguarda la seguente storia utente non abbiamo ritenuto fosse necessario un diagramma delle sequenze, in quanto consiste in un'unica chiamata a funzione che restituisce il grafico riguardante l'andamento delle lavorazioni.

8.2. Diagramma degli stati

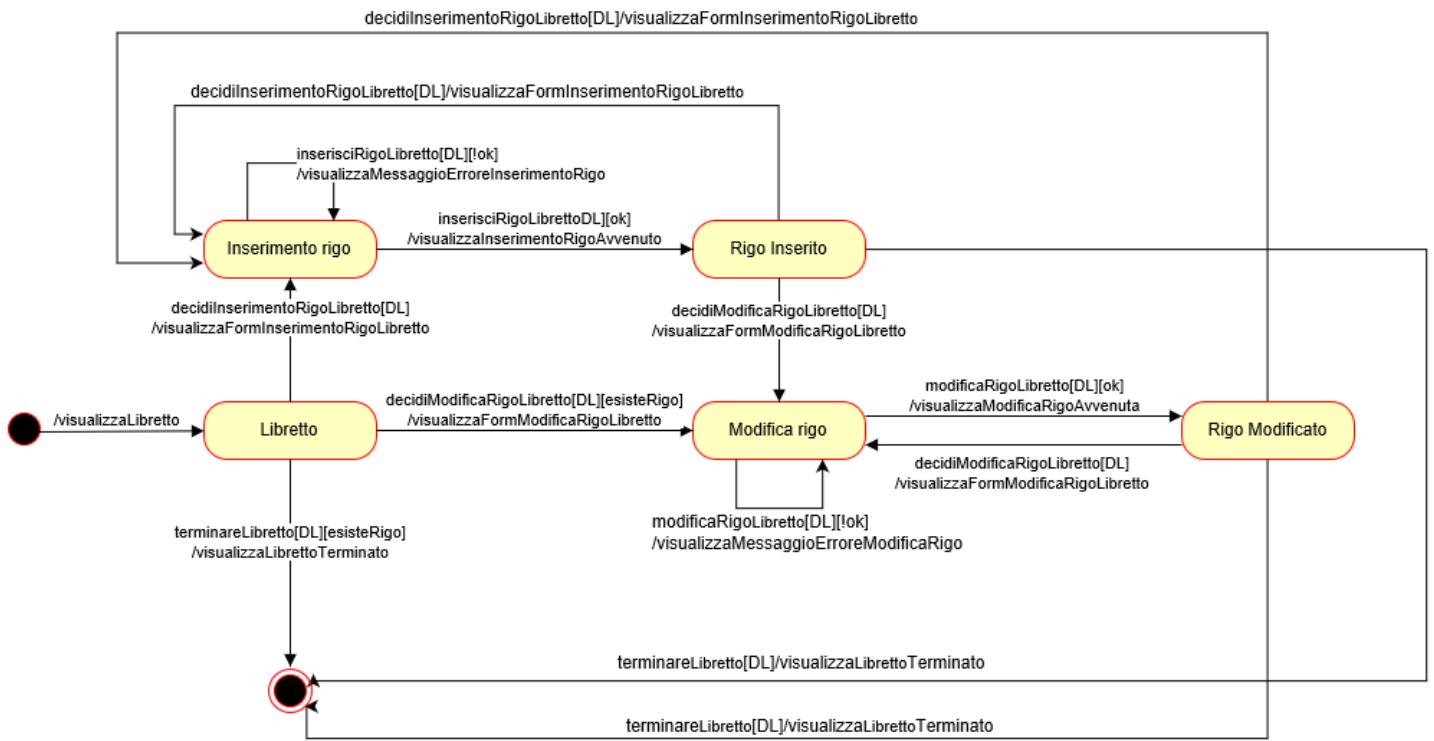
8.2.1. Story Card 1



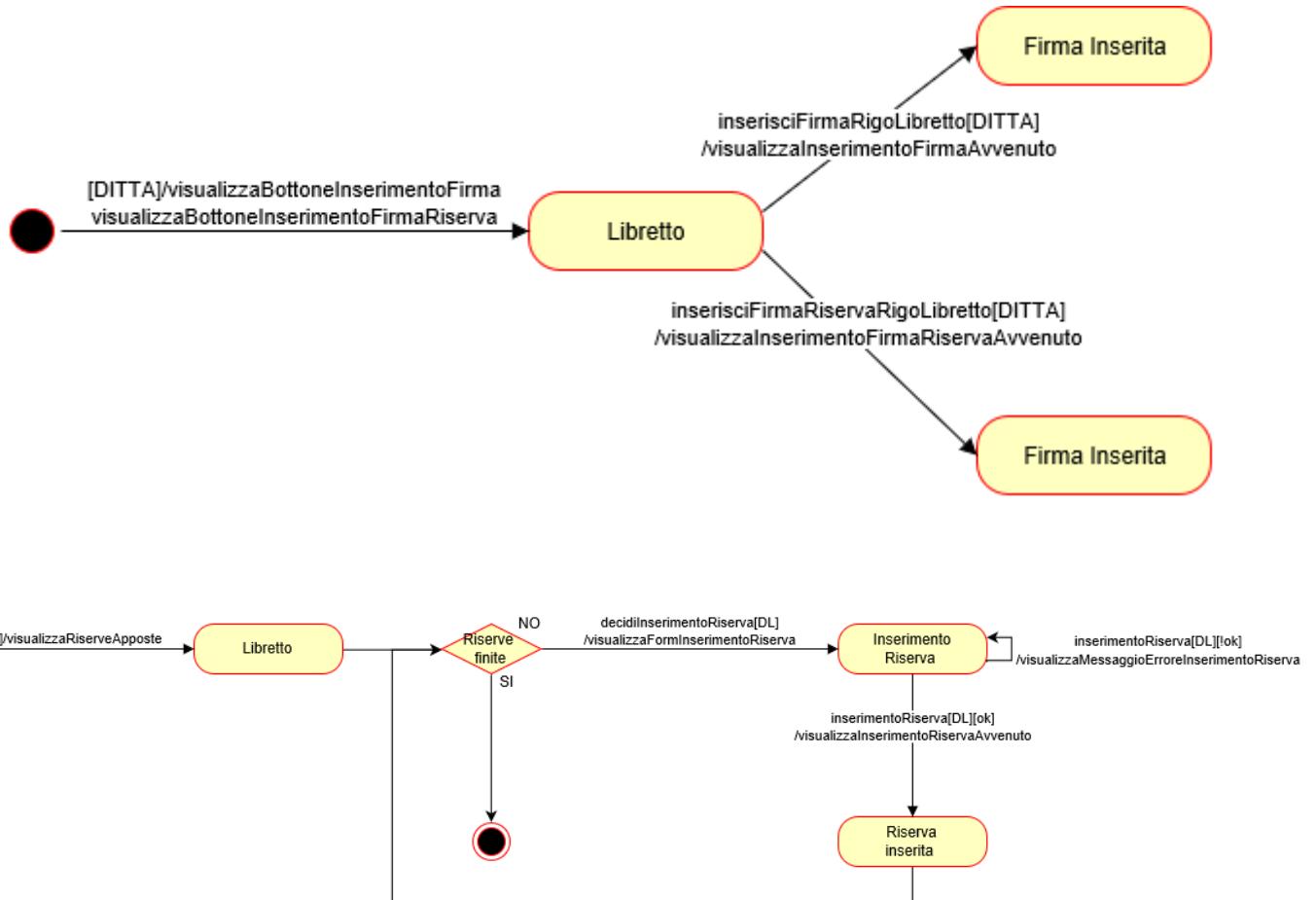
8.2.2. Story Card 2



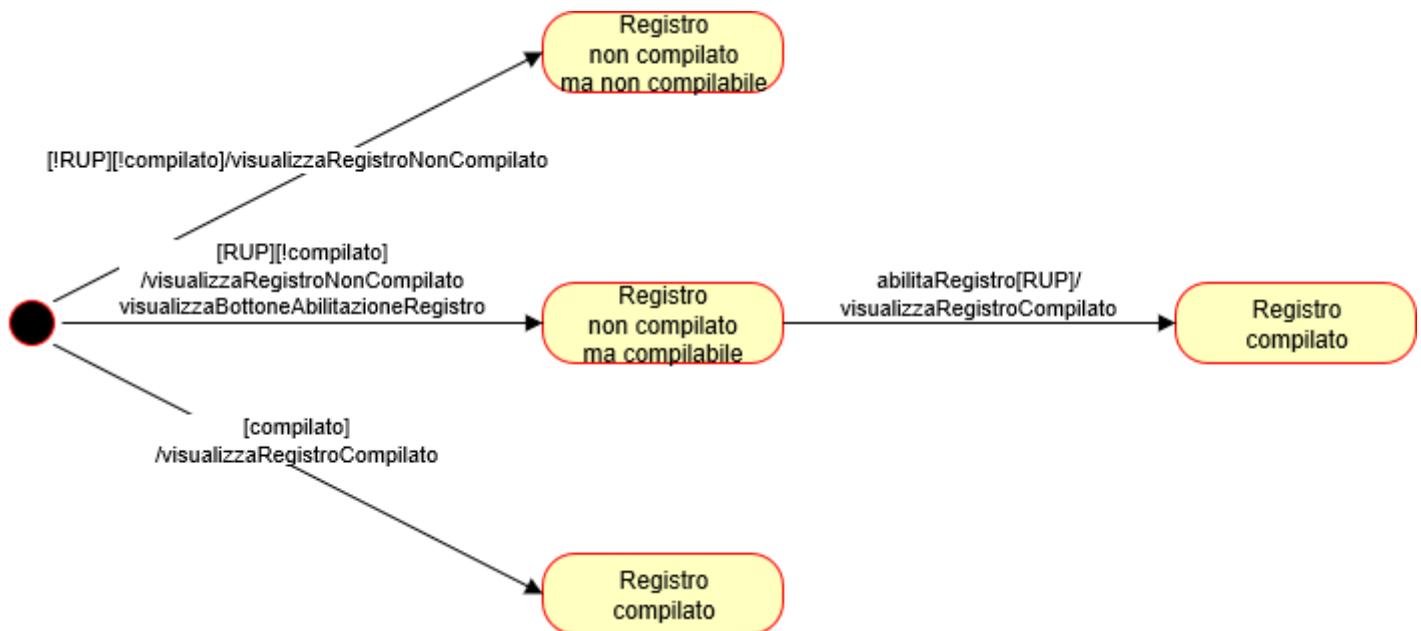
8.2.3. Story Card 3



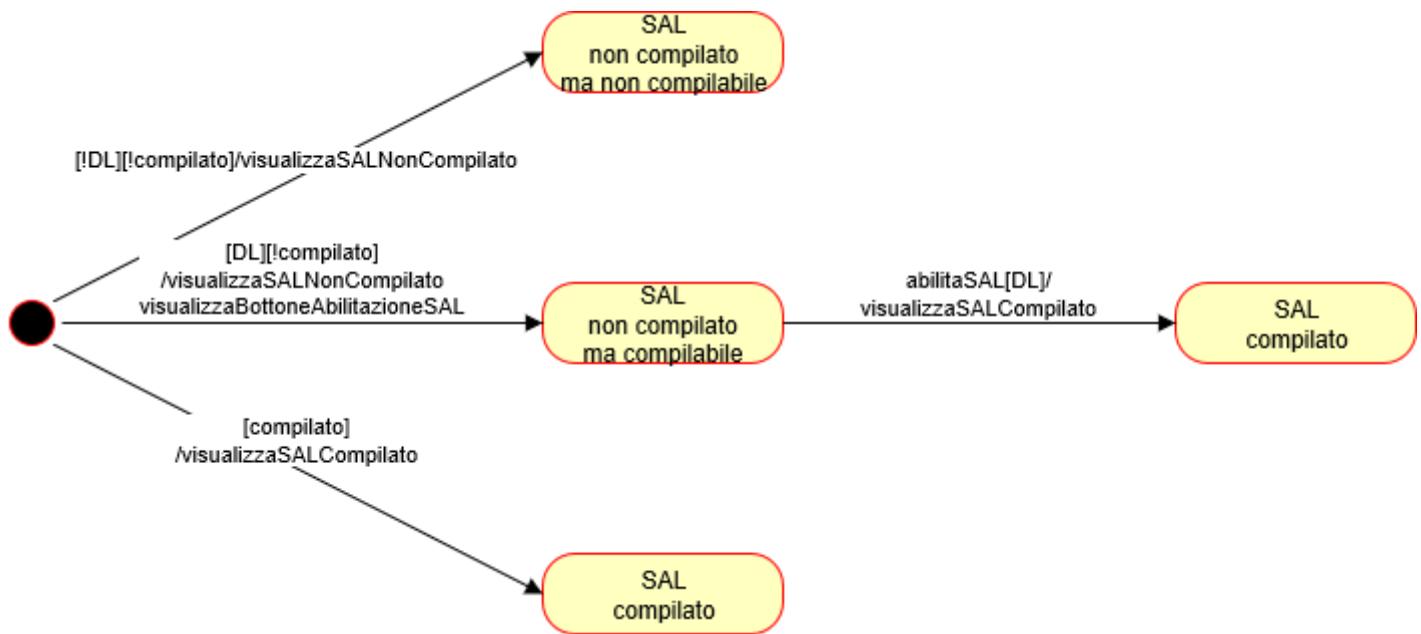
8.2.4. Story Card 4



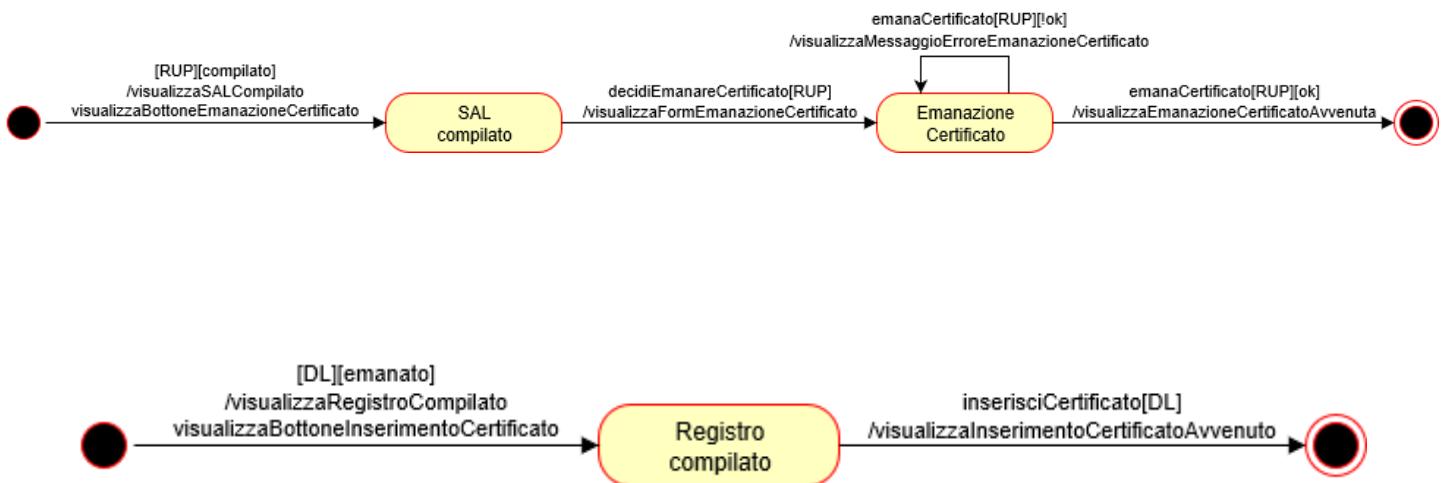
8.2.5. Story Card 5



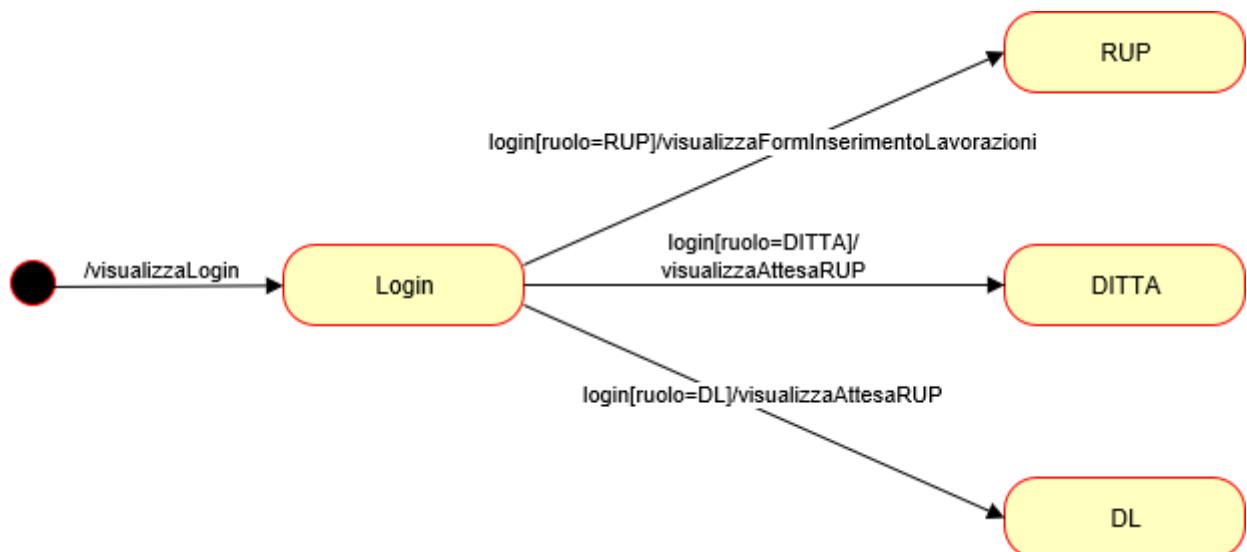
8.2.6. Story Card 6



8.2.7. Story Card 7



8.2.8. Story Card 8



8.2.9. Story Card 9

Per la story card 9 non abbiamo ritenuto di dover costituire un diagramma di stato in quanto l'operazione per ottenere la lista delle lavorazioni risulta essere piuttosto semplice.

8.2.10. Story Card 10

Anche per questa storia non abbiamo ritenuto di dover costituire un diagramma di stato in quanto l'operazione di visualizzazione del grafico che mostra l'andamento delle lavorazioni consiste in un'unica richiesta.

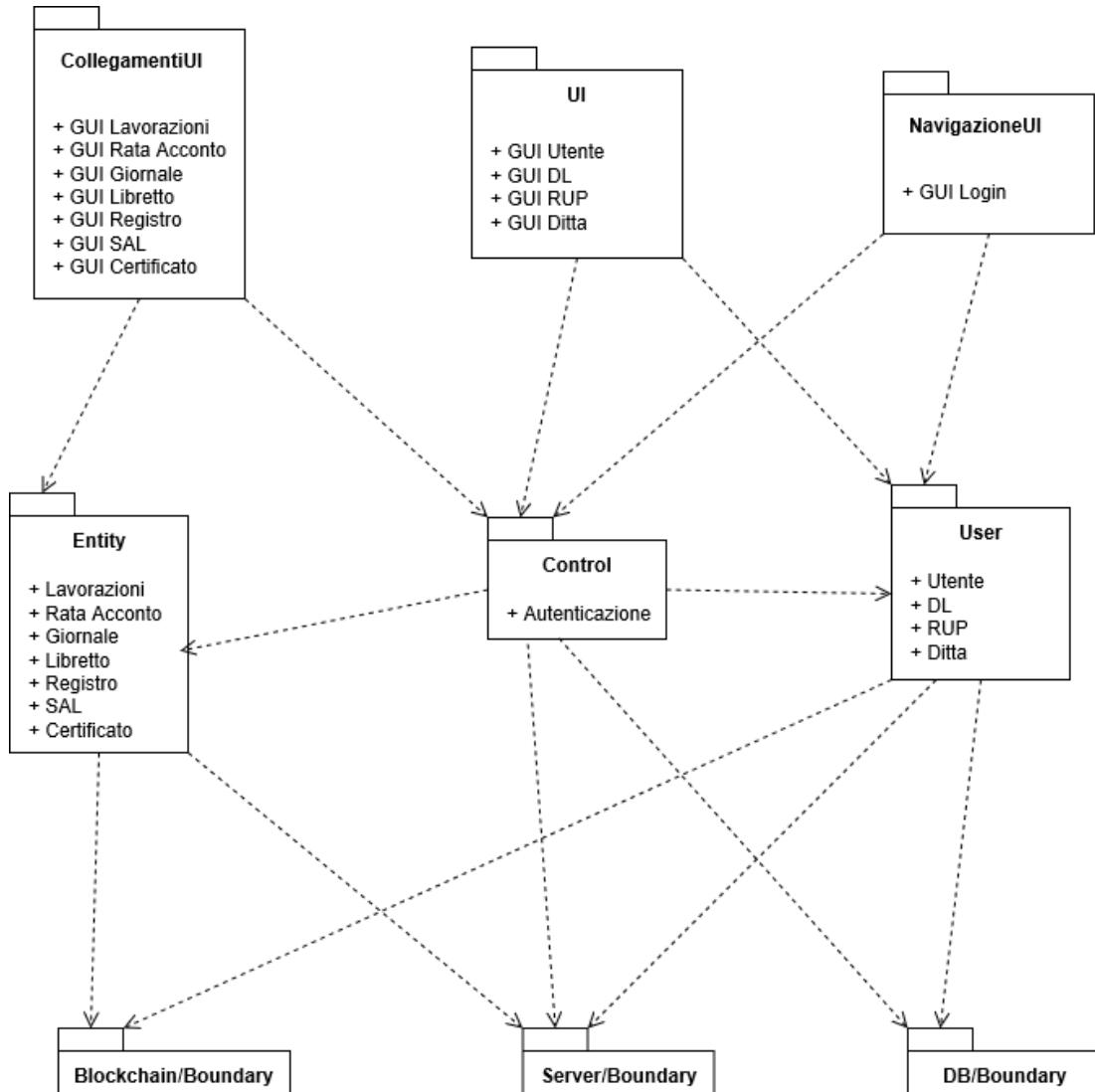
9. Progettazione

Conclusa la fase di analisi siamo passati alla fase di progettazione. I risultati di questa fase saranno i diagrammi dell'architettura e delle classi di progettazione.

Innanzitutto, per poter definire il diagramma dell'architettura abbiamo bisogno di raffinare il diagramma dei package presentato in precedenza. Il raffinamento deve poter produrre un diagramma in cui non si abbiano dipendenze circolari che, se presenti, dovranno essere eliminate tramite operazioni di eliminazione o fusione tra package.

Il diagramma dei package finale è rappresentato nella figura sottostante e non presenta alcuna dipendenza circolare.

Com'è possibile vedere, il package UI iniziale è stato separato in tre differenti package in modo da distinguere le diverse tipologie di UI che verranno realizzate.

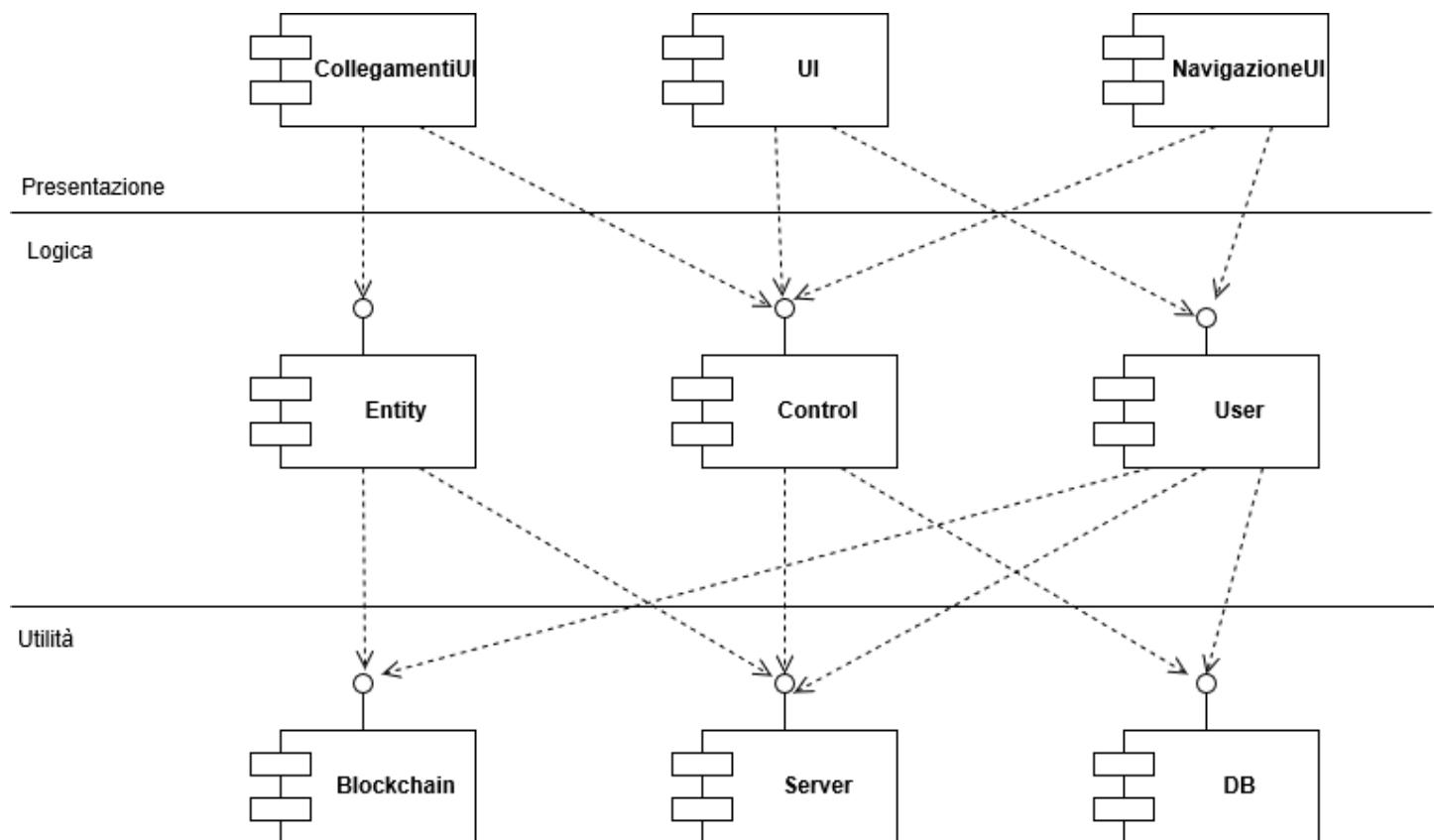


9.1. Diagramma dei componenti

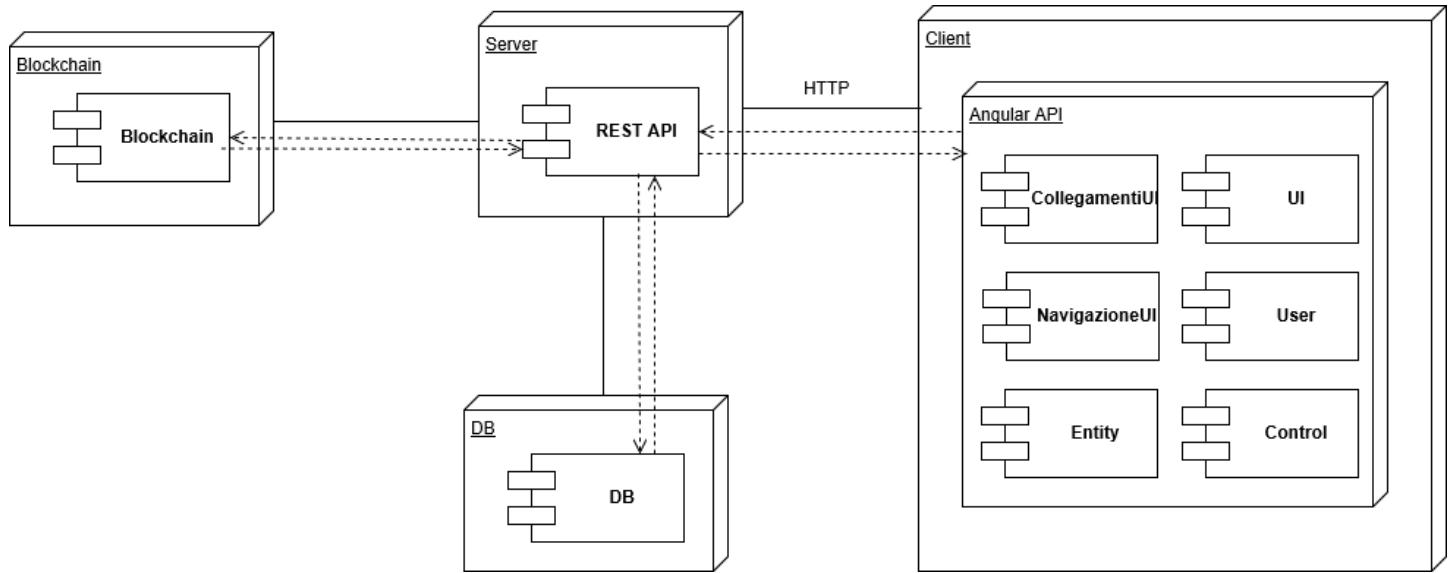
Il diagramma dei package finale, ci ha permesso di ricavare il diagramma dei componenti.

Questo presenta tre strati:

- **Presentazione:** permette l'interfacciamento con l'utente
- **Logica di business:** permette di gestire le operazioni richieste dall'utente, attraverso il modulo Control, per poter manipolare i dati dell'applicazione (Entity).
- **Utilità:** permette la comunicazione con il server, la Blockchain e il DB



9.2. Diagramma di dislocamento



9.3. Diagramma delle classi di progettazione

A partire dal diagramma delle classi siamo andati a costituire il diagramma delle classi di progettazione.

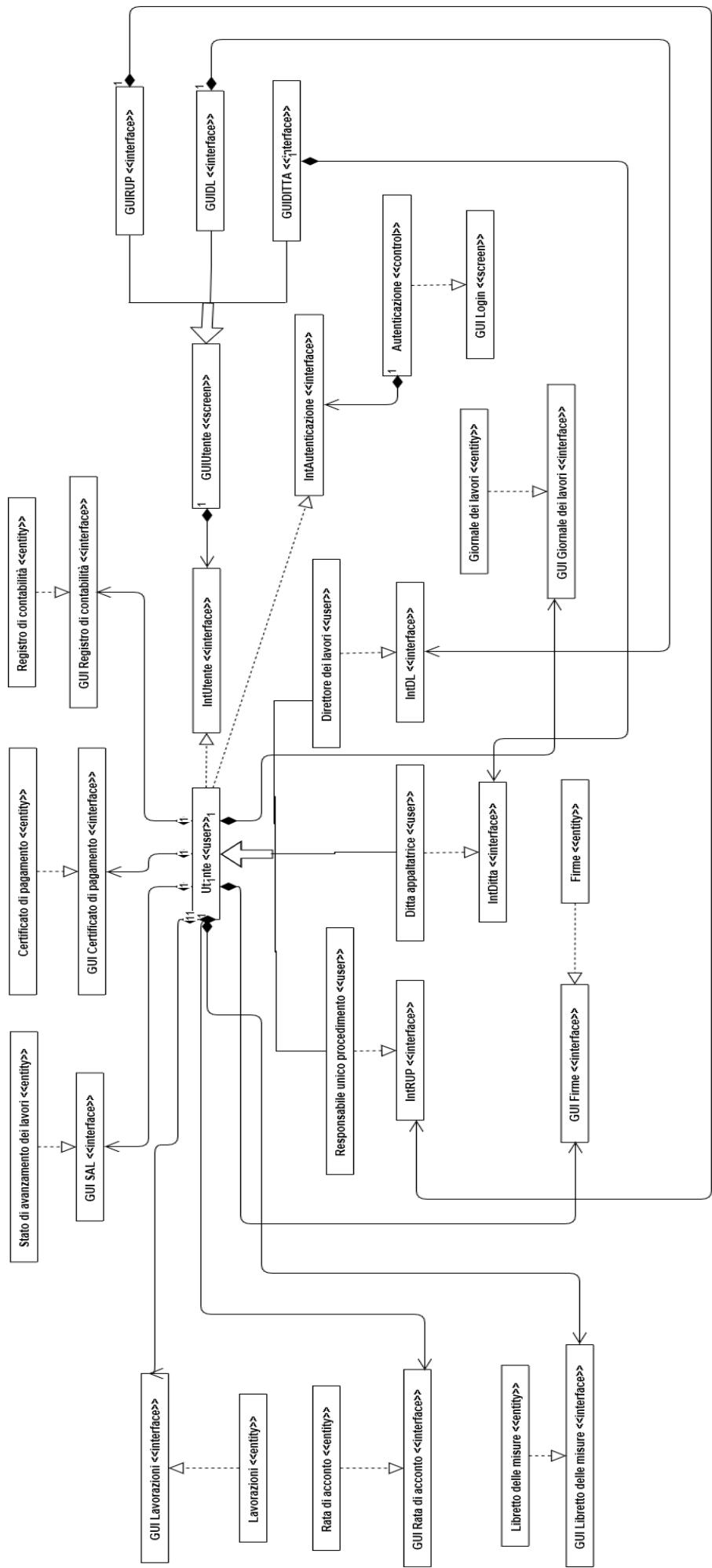
Innanzitutto, ogni classe di analisi è stata trasformata in una classe di progettazione. In seguito, siamo andati a valutare le dipendenze di una classe da altre classi di componenti esterni. In questo caso siamo andati ad inserire un'apposita interfaccia.

Infine, per ogni classe siamo andati a correggere attributi e metodi in modo da aggiungere elementi o eliminare possibili duplicati.

Com'è possibile osservare dal modello, ogni relazione rappresentata ha un preciso verso di navigazione.

In modo da semplificare la lettura abbiamo prima riportato le classi <> interface <> aggiunte al diagramma e poi il diagramma stesso.

<p>GUI Giornale dei lavori <<interface>></p> <ul style="list-style-type: none"> + visualizzaGiornale() + inserimentoRigoGiornale() + visualizzaBottoneInserimentoRigoGiornale() + visualizzaFormInserimentoRigoGiornale() + visualizzaBottoneModificaRigoGiornale() + visualizzaFormModificaRigoGiornale() + visualizzaBottoneTerminazioneGiornale() + visualizzaInserimentoRigoGiornaleAvvenuto() + visualizzaMessaggioErroreInserimentoRigoGiornale() + visualizzaModificaRigoGiornaleAvvenuta() + visualizzaMessaggioErroreModificaRigoGiornale() + visualizzaTerminazioneGiornaleAvvenuta() 	<p>GUI Libretto delle misure <<interface>></p> <ul style="list-style-type: none"> + visualizzaLibretto() + visualizzaBottoneInserimentoRigoLibretto() + visualizzaFormInserimentoRigoLibretto() + visualizzaBottoneModificaRigoLibretto() + visualizzaFormModificaRigoLibretto() + visualizzaBottoneTerminazioneLibretto() + visualizzaInserimentoRigoLibrettoAvvenuto() + visualizzaMessaggioErroreInserimentoRigoLibretto() + visualizzaModificaRigoLibrettoAvvenuta() + visualizzaMessaggioErroreModificaRigoLibretto() + visualizzaTerminazioneLibrettoAvvenuta()
<p>GUI Firme <<interface>></p> <ul style="list-style-type: none"> + visualizzaBottoneInserimentoFirma() + visualizzaBottoneInserimentoFirmaRIserva() + decidilnserireFirma() + decidilnserireFirmaRiserva() + visualizzalnserimentoFirmaAvvenuta() + visualizzalnserimentoFirmaRiservaAvvenuta() + visualizzaRiserveApposte() 	<p>GUI Certificato di pagamento <<interface>></p> <ul style="list-style-type: none"> + visualizzaBottoneEmanazioneCertificato() + visualizzaCertificatoEmanato() + visualizzaFormEmanazioneCertificato() + visualizzaEmanazioneCertificatoAvvenuto() + visualizzaMessaggioErroreEmanazioneCertificato() + visualizzaBottoneInserimentoCertificato()
<p>GUI SAL <<interface>></p> <ul style="list-style-type: none"> + visualizzaSAL() + visualizzaBottoneAbilitazioneSAL() 	<p>GUI Registro di contabilità <<interface>></p> <ul style="list-style-type: none"> + visualizzaRegistroNonCompilato() + visualizzaRegistroCompilato() + visualizzaBottoneAbilitazioneRegistro()
<p>GUI Lavorazioni <<interface>></p> <ul style="list-style-type: none"> + visualizzaFormInserimentoLavorazione() + visualizzalnserimentoLavorazioneAvvenuto() + visualizzaMessaggioErroreInserimentoLavorazione() + modificaLavorazione() + visualizzaFormModificaLavorazione() + visualizzaModificaLavorazioneAvvenuta() + visualizzaMessaggioErroreModificaLavorazione() + visualizzaBottoneTerminazioneInserimentoLavorazioni() 	<p>GUI Rata di acconto <<interface>></p> <ul style="list-style-type: none"> + visualizzaFormInserimentoRata() + visualizzalnserimentoRataAvvenuto() + visualizzaMessaggioErroreInserimentoRata() + visualizzaFormModificaRata() + modificaRataAcconto() + visualizzaModificaRataAvvenuta() + visualizzaMessaggioErroreModificaRata() + visualizzaBottoneTerminazioneInserimentoRata() + visualizzaResocontoInserimenti()



Dopo aver individuato le classi di progettazione, si è passati ad individuare le classi di tipo <>entity<> ed <>user<> con le relative associazioni. Queste classi saranno le classi che andranno a comporre il modello logico di un database.

Per quanto riguarda le classi <>user<> non abbiamo avuto grossi problemi nel ricavare il modello relazionale, in quanto è stato deciso di contenere i dati riguardanti gli utenti all'interno di un database locale.



Invece, per quanto riguarda le restanti classi, non siamo riusciti a creare delle vere e proprie classi da inserire nel modello relazionale, in quanto il linguaggio di programmazione Solidity, con cui andremo ad implementare queste classi, non predispone la possibilità di creare delle classi. Pertanto, abbiamo pensato di far matchare le classi <>entity<> con uno smart contract, creando delle struct per contenere gli attributi della classe e creare delle funzioni per implementare i metodi della classe. In realtà, per quanto riguarda le classi "Lavorazione" e "Rata di Acconto" abbiamo deciso di creare un unico smart contract, in quanto i due concetti rientrano all'interno di una stessa macro-operazione. Una cosa simile è stata fatta con le classi "Stato avanzamento dei lavori" e "Certificato di pagamento", infatti, dato che il certificato di pagamento è collegato con il concetto del SAL, sono stati inglobati in uno stesso smart contract.

Individuati i componenti da inserire in modo da garantire la persistenza delle informazioni, abbiamo inserito le classi DAO che modellano le interazioni CRUD sia con il DB che con la Blockchain.

DAO Lavorazioni
+ load(): Lavorazioni + find(): Lavorazioni + findAll(): Lavorazioni[] + update()

DAO Giornale
+ load(): Giornale + find(): Giornale + findAll(): Giornale[] + update()

DAO Libretto
+ load(): Libretto + find(): Libretto + findAll(): Libretto[] + update()

DAO Utente
+ load(): Utente + find(): Utente

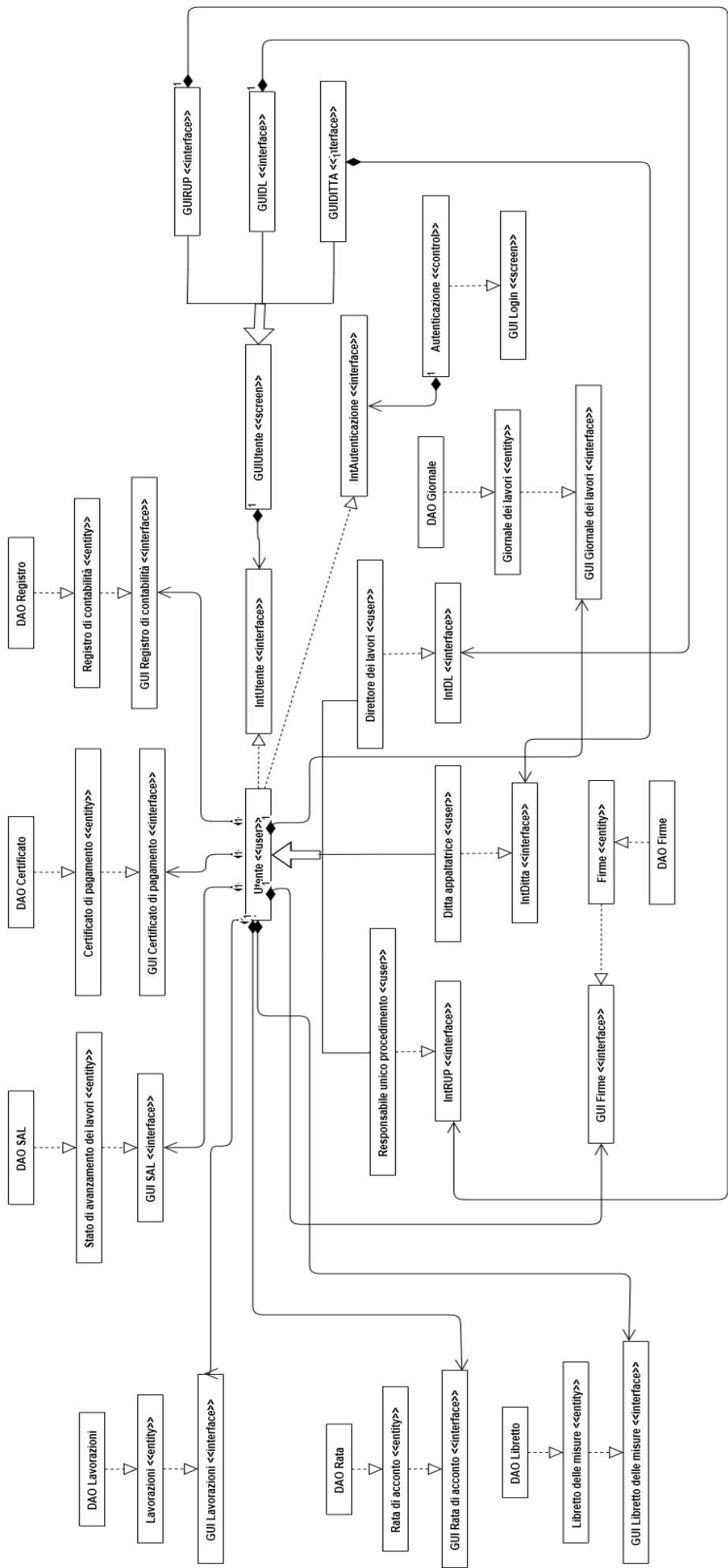
DAO Firme
+ load(): Firme + find(): Firme + findAll(): Firme

DAO Certificato
+ load(): Certificato + find(): Certificato + findAll(): Certificato[]

DAO Rata Acconto
+ load(): Rata + find(): Rata + update()

DAO Registro
+ load(): Registro + find(): Registro + findAll(): Registro[]

DAO SAL
+ load(): Sal + find(): Sal + findAll(): Sal[]



10. Risorse server

Prima di partire con l'implementazione siamo andati a creare uno schema delle risorse CRUD che siamo poi andati ad utilizzare nella progettazione server.

Risorsa	Metodo http	Descrizione	Parametri Input	Parametri Output
/login	POST	Restituisce un token per poter fare delle chiamate autenticate alle rotte sotto indicate.	Body della richiesta <pre>{ "token": string, "nome": string, "cognome": string, "ruolo": string }</pre>	<pre>{ "token": string, "nome": string, "cognome": string, "ruolo": string }</pre>
/info	GET	Restituisce le informazioni di un utente.		<pre>{ "nome": string, "cognome": string, "ruolo": string }</pre>
/check	GET	In base all'utente restituisce delle informazioni, non tutti i campi lì presenti vengono restituiti a tutti gli utenti.		<pre>{ "libretti": [{ "num_libretto": number, "timestamp": number }], "giornale": boolean, "inser_terminati": boolean, "lavori_terminati": boolean, "num_lavori": number, "soglia": number, "registri_da_compilare": [{ "num_libretto": number, "timestamp": number }], "soglia_superata": boolean, "num_sal": number }</pre>
/allegati	PUT	Richiesta per salvare sul server gli allegati che poi verranno inseriti in un rigo del libretto. Restituisce il nome del file con cui verrà salvato sul server.	Body in cui è presente un file binario.	<pre>{ "path": string }</pre>

/appalto	POST (solo RUP)	Inserimento di un lavoro da parte del RUP all'interno del contratto di appalto Funzioni di riferimento blockchain: - finishInserimenti		Risposta vuota con status code.
/appalto/terminaContratto	POST	Serve a fare una terminazione prematura del contratto d'appalto. Così da chiuderlo definitivamente. Funzioni di riferimento blockchain: killAvanzamento		Risposta vuota con status code.
/appalto/lavoro	POST (solo RUP)	Inserimento di un lavoro da parte del RUP all'interno del contratto di appalto Funzioni di riferimento blockchain: - addlavoro	Body della richiesta { "codLavoro": string, "nomeLavoro": string, "costoLavoro": number, } 	Risposta vuota con status code.
/appalto/lavoro	GET	Restituisce la lista dei lavori Funzioni di riferimento blockchain: - getlavori		{ "codLavoro": string, "nomeLavoro": string, "costoLavoro": number, "perc_completamento": number } []
/appalto/soglia	POST (solo RUP)	Inserimento valore di soglia Funzioni di riferimento blockchain: - addsoglia	Body della richiesta { "soglia": number, }	Risposta vuota con status code.
/appalto/soglia	GET	Restituisce il valore di soglia Funzioni di riferimento blockchain: - getsogli		{ "soglia": number, }

/contratto/giornale	POST (solo DIR)	<p>Inserisce un rigo nel giornale dei lavori ancora da terminare, o eventualmente, specificando dei parametri, sovrascrive un rigo del giornale ancora da terminare.</p> <p>I parametri</p> <pre>"num_giorn": number, "rigo": number</pre> <p>vanno esplicitati nel momento in cui si vuole sovrascrivere un determinato rigo del giornale corrente.</p> <p>Funzioni di riferimento blockchain:</p> <ul style="list-style-type: none"> - insertgiornale - updategiornale 	<p>Body della richiesta</p> <pre>{ "num_giorn": number, "rigo": number, "personale": [{ "quantita": number, "qualita": string, "ore": number }], "attrezzatura": [{ "nomeAtt": string, "quantitaAtt": number }], "descrizione": string }</pre>	Risposta vuota con status code.
/contratto/giornale	PUT (DIR)	<p>Termina l'inserimento dei righi del giornale corrente.</p> <p>Funzioni di riferimento blockchain:</p> <ul style="list-style-type: none"> - finishgiornale 		Risposta vuota con status code.
/contratto/giornale	GET	<p>Restituisce il giornale e le sue sovrascritture a partire dalla data, sulla blockchain.</p> <p>Se viene passato il timestamp in millisecondi di una certa data viene restituita una lista di giornali in tale data. Se non viene passato nulla viene restituito:</p> <ul style="list-style-type: none"> - DIR : giornale corrente da compilare e terminare; - RUP/DIT : l'ultimo giornale terminato dal DIR. <p>Se viene specificato il parametro mese viene restituita un array di numeri che indicano i giorni di quel mese dove è stato compilato il giornale dei lavori.</p> <p>Funzioni di riferimento blockchain:</p> <ul style="list-style-type: none"> - Get1 - Get2 - Get3 - Getgiornaledata 	<p>Parametri query string:</p> <ul style="list-style-type: none"> - timestamp : number - mese : number 	<p>Se specificato timestamp o se non viene specificato nessun parametro.</p> <pre>{ "num_giorn": number, "righi": [{ "personale": [{ "quantita": number, "qualita": string, "ore": number }], "attrezzatura": [{ "nomeAtt": string, "quantitaAtt": number }], "timestamp": number, "descrizione": string, "sovrascritto": number, "num_rigo": number }] }</pre> <p>Se specificato il parametro mese number []</p>

contratto/lib rettoMisure	POST (solo DIR)	<p>Inserisce una misura nel libretto corrente, non ancora terminato, sulla blockchain.</p> <p>I parametri</p> <pre>"num_libretto": "num_rigo":</pre> <p>vanno esplicitati nel momento in cui si vuole sovrascrivere un determinato rigo del libretto.</p> <p>Funzioni di riferimento blockchain:</p> <ul style="list-style-type: none"> - addlibretto - Updatelibretto 	<p>Body della richiesta</p> <pre>{ "descrizione": string, "num_rigo": number, "num_libretto": number, "codLavoro": string, "percentuale": number, "allegati": string [] }</pre>	Risposta vuota con status code.
contratto/lib rettoMisure	PUT (solo DIR)	<p>Terminazione del libretto, cioè lo si rende non più modificabile e si passa al successivo, come per il giornale dei lavori.</p> <p>Funzioni di riferimento blockchain:</p> <ul style="list-style-type: none"> - Finishlibretto 		Risposta vuota con status code.
/contratto/li brettoMisure	GET	<p>Restituisce il libretto e le sue sovrascritture a partire dalla data, sulla blockchain.</p> <p>Se viene passato il timestamp in millisecondi di una certa data viene restituito un array di libretti in tale data. Se non viene passato nulla viene restituito:</p> <ul style="list-style-type: none"> - DIR : libretto corrente da compilare e terminare, non sono presenti i campi <i>firmato</i>, <i>firmato_riserva</i>, <i>riserva_dir</i>, <i>percentuale_riserva</i>, <i>descrizione_riserva</i> ; - RUP/DIT : l'ultimo libretto terminato dal DIR. <p>Se viene specificato il parametro mese viene restituita un array di numeri che indicano i giorni di quel mese dove è stato terminato un libretto.</p> <p>Funzioni di riferimento blockchain:</p> <ul style="list-style-type: none"> - Getlibretti1 - Getlibretti2 - Getlibrettodata 	<p>Parametri query string:</p> <ul style="list-style-type: none"> - timestamp : number - mese : number 	<pre>{ "num_libretto": number, "righi": [{ "timestamp": number, "descrizione": string, "sovrascritto": number, "num_rigo": number, "codLavoro": string, "percentuale": number, "allegati": string [], "firmato": boolean, "firmato_riserva": boolean, "riserva_dir": boolean, "percentuale_riserva": number, "descrizione_riserva": string }], "timestamp": number }</pre>

/contratto/librettoMisure/riserve	POST (solo DIT)	<p>Se il campo riserva è true allora viene lanciata la funzione di firma con riserva da parte della Ditta, altrimenti la firma senza riserva.</p> <p>Funzioni di riferimento blockchain:</p> <ul style="list-style-type: none"> - insertfirmadittalibretto - insertfirmariservadittalibretto 	<p>Body della richiesta</p> <pre>{ "percentuale_riserva": number, "descrizione_riserva": string, "num_rigo": number, "num_libretto": string }</pre>	Risposta vuota con status code.
/contratto/librettoMisure/riserve	PUT (solo da parte del direttore e dei lavori)	<p>Serve per inserire la riserva da parte del direttore dei lavori, sulla misura sulla quale la ditta ha espresso la riserva</p> <p>Funzioni di riferimento blockchain:</p> <ul style="list-style-type: none"> - insertriserva 	<p>Body della richiesta</p> <pre>{ "riserva": boolean, "num_rigo": number, "num_libretto": string }</pre>	Risposta vuota con status code.
/contratto/registroContabilita	PUT (solo RUP)	<p>Serve ad abilitare il calcolo automatico del registro di contabilità,</p> <p>Funzioni di riferimento blockchain:</p> <ul style="list-style-type: none"> - enableregistrocontabilita 		Risposta vuota con status code.
/contratto/registroContabilita	GET	<p>Restituisce il registro a partire dalla data, sulla blockchain. Se viene passato il timestamp in millisecondi di una certa data viene restituito un array di registri in tale data. Se non viene passato nulla viene restituito l'ultimo registro compilato dal RUP.</p> <p>Se viene specificato il parametro mese viene restituito un array di numeri che indicano i giorni di quel mese dove è stato compilato un registro.</p> <p>Funzioni di riferimento blockchain:</p> <ul style="list-style-type: none"> - Getregistri1 - Getregistri2 - Getregistri3 - Getregistri4 - getregistroData 	<p>Parametri query string:</p> <ul style="list-style-type: none"> - timestamp : number - mese : number 	<pre>{ "num_registro": number, "righi": [{ "codLavoro": string, "percentuale_raggiunta_lavoro": number, "percentuale_lavoro_sul_totale": number, "nomeLavoro": string, "percentuale_lavoro_sul_totale_raggiunta": number, "costo_raggiunto": number, "costoLavoro": number }], "timestamp": number }</pre>

/contratto/statoAvanzamentoLavori	PUT (solo DIR)	Abilita il calcolo automatico del SAL. Funzioni di riferimento blockchain: enableSal		Risposta vuota con status code.
/contratto/statoAvanzamentoLavori	GET	Restituisce il sal a partire dalla data, sulla blockchain. Se viene passato il timestamp in millisecondi di una certa data viene restituito un array di sal in tale data. Se non viene passato nulla viene restituito l'ultimo sal compilato dal DIR. Se viene specificato il parametro mese viene restituito un array di numeri che indicano i giorni di quel mese dove è stato compilato un sal. Funzioni di riferimento blockchain: - getSal1 - getsal2 - getSalData	Parametri query string: - timestamp : number - mese : number	{ "num_sal": string, "righi": [{ "codLavoro": string, "percentuale_raggiunta_lavoro": number, "percentuale_lavoro_sul_totale": number, "percentuale_lavoro_sul_totale_raggiunta": number, "nomeLavoro": string, "costo_raggiunto": number, "costoLavoro": number }], "timestamp": number }
/contratto/statoAvanzamentoLavori/grafico	GET	Restituisce i dati per costruire un grafico che mostra l'andamento complessivo dei lavori nel corso del tempo.		{ "timestamp": number, "numeri_sal": number, "val_monetario": number }

In cui lo status code sarà:

- **200** se la richiesta è stata processata correttamente e viene restituita una risposta corretta.
- **201** se la richiesta è stata processata correttamente ed ha portato alla scrittura di qualche informazione sulla blockchain.
- **400** se la richiesta non è stata effettuata con i parametri corretti.
- **401** se la richiesta o non conteneva il token oppure il token non era valido.
- **403** la richiesta è corretta ma il server non risponde perché l'utente non è autorizzato per quella rotta.
- **500** se il server ha avuto qualche problema durante l'esecuzione.

11. Implementazione

Concluse le fasi di analisi e progettazione, siamo passati alla realizzazione della nostra web app. In una prima fase siamo andati a scegliere i vari componenti da utilizzare per l'implementazione della nostra applicazione, poi siamo passati alla realizzazione vera e propria della nostra applicazione.

Dato che la scelta, presa in accordo con gli stakeholder, è stata quella di una web application, si è deciso di utilizzare Angular per quanto riguarda il frontend e NodeJS per il backend.

L'IDE che si è scelto di utilizzare è Visual Studio Code, che permette l'installazione di molte librerie necessarie al nostro progetto. Permette anche l'installazione di librerie in modo da scrivere codice con il linguaggio di programmazione Solidity, con cui andremo ad implementare gli smart contract, però per quest'ultimo si è utilizzato un differente IDE. Infatti, la nostra scelta è ricaduta su Remix IDE, un IDE Browser-based che predispone di una serie di strumenti per interagire con gli smart contract e mette a disposizione un ambiente di testing e debugging molto accurato.

11.1. Frontend

```
C:\Users\Iaco>ng version
          _                         _     _ 
         / \   | |  | |  | |  | |  | |  | |  | |
        /   \  | |  | |  | |  | |  | |  | |  | |
       /     \ | |  | |  | |  | |  | |  | |  | |
      /       \| |  | |  | |  | |  | |  | |  | |
     /         \| |  | |  | |  | |  | |  | |  | |
    /           \| |  | |  | |  | |  | |  | |  | |
   /             \| |  | |  | |  | |  | |  | |  | |
  /               \| |  | |  | |  | |  | |  | |  | |
 /                 \| |  | |  | |  | |  | |  | |  | |
/                   \| |  | |  | |  | |  | |  | |  | |
Angular CLI: 7.3.3
```

Per quanto riguarda il frontend, come riferito sopra, si è deciso di utilizzare Angular, in particolare la versione 7. Inoltre, si è scelto di utilizzare Bootstrap per quanto riguarda la parte grafica del sito.

Dopo aver scaricato ed installato la versione corretta, seguendo le istruzioni presenti sul sito di Angular, siamo passati a creare un nuovo progetto. In questo, ci è stato molto utile l'IDE utilizzato, in quanto, nel caso di utilizzo di un linguaggio o librerie non installate, segnala il nuovo linguaggio (o libreria) e permette di installarla

in molto semplice ed automatico. Installate tutte le librerie necessarie, abbiamo avviato il vero e proprio progetto Angular. Infatti, posizionandoci nella directory scelta per il progetto tramite terminale, basta lanciare il comando “*ng new NOME_PROGETTO*” ed il tutto viene avviato in modo automatico.

A questo punto abbiamo potuto procedere con l’implementazione dell’applicazione, andando mano a mano a testare i vari componenti. Per poter visualizzare l’interfaccia è possibile lanciare il comando “*ng serve*” attraverso il quale è possibile accedere all’indirizzo <http://localhost:4200> e visualizzare la nostra applicazione.

11.2. Backend



Per la parte inherente al backend abbiamo utilizzato NodeJS, appoggiandoci su uno dei più famosi web framework, ovvero Express. Si tratta di un framework per NodeJS che permette di creare in modo molto più rapido applicazioni in Node.

Dopo aver installato Node direttamente dal sito, si è installato anche Express con il comando “*npm install express --save*”, che, oltre ad installare Express, permette di aggiungerlo all’elenco di dipendenze.

A questo punto, per poter avviare un progetto in Node, tramite terminale, ci è bastato digitare all’interno della directory del progetto Node, il comando “*node NOME_APP.js*”.

11.3. Blockchain



Tecnologia centrale al nostro progetto è stata ovviamente la Blockchain. Come spiegato nella parte introduttiva della relazione, la scelta è ricaduta su di una Permissioned Blockchain chiamata Quorum. Infatti, nel nostro caso, avere a che fare con documenti e dati sensibili, ci ha spinto all'utilizzo di una Blockchain privata, in cui si hanno maggiori meccanismi di sicurezza e privacy rispetto alle controparti pubbliche. Tutto ciò fa sì che le transazioni scambiate all'interno della nostra rete siano visibili solo ai peer che hanno accesso alla rete.

Per la sua installazione, la JP Morgan Chase (ideatrice di Quorum), ha messo a disposizione un repository su GitHub "<https://github.com/jpmorganchase/quorum>" dalla quale è possibile scaricare una rete Quorum già implementata che dispone di 7 nodi. Ovviamente, è possibile modificare a proprio piacimento i parametri della rete, infatti, abbiamo diminuito il numero di nodi a 5.

Come consigliato dal manuale di installazione, abbiamo deciso di far partire i vari nodi della rete su macchine virtuali. Per questo siamo andati ad installare Vagrant, ovvero un gestore di macchine virtuali che consente la creazione di più macchine virtuali che utilizzino una stessa configurazione. Ovviamente, per poterlo utilizzare, si deve disporre di un software che permetta l'esecuzione di macchine virtuali, ad esempio Virtual Box.

Quindi, dopo aver installato Virtual Box, Vagrant ed infine il repository di Quorum, basta spostarsi all'interno della directory di Quorum e lanciare in sequenza i comandi "`vagrant up`" e "`vagrant ssh`" in modo da connettersi tramite un client SSH alle macchine virtuali. Il primo comando serve per configurare l'accesso SSH utilizzando i parametri interni al *VagrantFile*, contenuto nella directory di Quorum, mentre il secondo serve per connettersi alla macchina virtuale. Infatti, al termine di questi passaggi ci troveremo, tramite riga di comando, all'interno di una macchina virtuale Linux.

Per quanto riguarda i protocolli di consenso messi a disposizione da Quorum, abbiamo scelto l'utilizzo di Raft in quanto è il più utilizzato dei tre disponibili e ha un meccanismo molto rapido per il processamento e la conferma di una transazione. Infatti, dato che gli attori della nostra applicazione possono considerarsi "fidati", è molto più vantaggioso avere un meccanismo di convalida delle transazioni che sia molto rapido e che si riesca quindi a mostrare i dati inseriti in modo immediato.

11.4. Smart contract



Attualmente, Solidity, rappresenta il più importante linguaggio di programmazione contract-oriented per l'implementazione di smart contract su piattaforme Ethereum ed Ethereum-based. Inoltre, Remix IDE, scelta per la scrittura e testing dei contratti intelligenti, disponeva sino all'ultima versione, solamente la possibilità di utilizzare Solidity come linguaggio di programmazione. Per questo motivo abbiamo scelto proprio Solidity come linguaggio.

Inoltre, i suoi vantaggi rispetto ai maggiori concorrenti come Serpent sono:

- Ha a disposizione variabili membro complesse, come mapping e struct
- Introduce un ABI (Application Binary Interface) per controllare il codice ed impedire errori di tipo durante la fase di compilazione

Per poter utilizzare Solidity, non abbiamo avuto bisogno di alcun'installazione, in quanto Remix IDE ha già presenti al suo interno tutte le librerie built-in necessarie.

Durante l'implementazione degli smart contract si è rivelata fondamentale una libreria, ad oggi, sperimentale chiamata ABIEncoderV2. Infatti, in Solidity non è ancora possibile passare struct o array come parametri di una funzione, mentre questa libreria permette questo, utilizzando un nuovo tipo di parametro che chiama *tuple*.

11.5. Truffle



Una volta iniziati ad implementare i contratti, si è resa necessaria la migrazione di questi smart contract all'interno della Blockchain.

Un primo metodo è sicuramente quello di inserire in modo manuale i contratti all'interno di Quorum, ma sicuramente la soluzione più pulita ed utilizzata è il deploy del contratto attraverso l'utilizzo del software Truffle.

Per poter utilizzare Truffle, è necessario avviare il comando “`npm install -g truffle`”, che andrà ad installare i pacchetti necessari per l'utilizzo. In seguito, si dovrà andare a creare l'ambiente Truffle. Per farlo è possibile eseguire il comando “`truffle init`”, che produrrà come risultato uno scheletro completo di un progetto Truffle.

A questo punto sarà possibile inserire all'interno della directory `contracts/` i vari smart contract da migrare. Da notare la presenza di un contratto creato in modo automatico dal processo di inizializzazione dell'ambiente Truffle, che è il contratto `Migration.sol`. Questo, in combinazione con il file di migrazione `1_deploy_contracts.js` (file creato anch'esso in modo automatico e contenuto all'interno della cartella `migrations/`), servirà per poter iniziare l'operazione di deploying.

Non resta far altro che andare a creare e poi inserire i file di migrazione necessari al deploying all'interno della cartella `migrations/` ed eseguire i comandi per la migrazione. In particolare, dovrà essere lanciato il comando `“truffle compile”` che andrà a compilare i contratti e salvare gli artefatti all'interno della directory `build/contracts` ed il comando `“truffle migrate”`, che invece andrà a fare la migrazione vera e propria. Solamente a questo punto sarà possibile richiamare le funzionalità degli smart contract attraverso l'API REST che andremo a creare.

11.6. Database



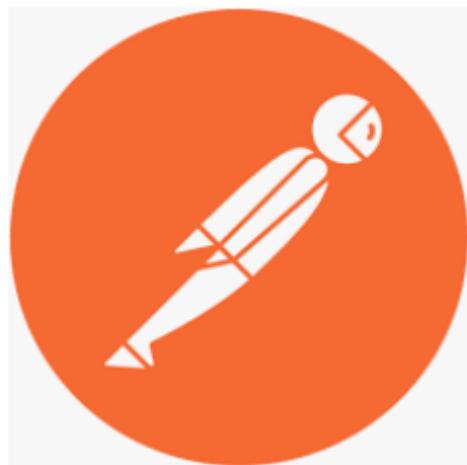
In modo da mantenere alcune delle informazioni chiave riguardante utenti, smart contract e nodi della Blockchain, ci siamo serviti di un database. In particolare, abbiamo fatto affidamento a MySQL.

Innanzitutto, abbiamo utilizzato MySQL per poter salvare le credenziali di accesso ad AppAlto. Infatti, abbiamo ritenuto che questi dati siano definiti a priori dell'inizializzazione del contratto d'appalto e quindi da salvare all'interno di un DB.

Altri dati molto importanti da dover salvare nel DB sono le chiavi pubbliche dei nodi autorizzati ad accedere alle informazioni presenti sulla Blockchain e gli indirizzi degli account che corrispondono ai nostri tre diversi attori.

Infine, abbiamo ritenuto di dover salvare gli indirizzi in cui vengono deployati gli smart contract tramite Truffle e l'ABI dei contratti, ovvero uno scheletro del contratto che riporta la signature dei metodi dei contratti.

11.7. Postman



Altro strumento utilizzato è Postman. Quest'applicazione consente di costruire e testare API REST molto velocemente.

Quindi una volta create le REST su NodeJS, si è passato a testarle con Postman, per verificarne l'esattezza. In questo modo non si ha bisogno di dover lanciare tramite terminale le diverse chiamate, ma è possibile farlo tramite un'interfaccia semplice ed intuitiva.

12. Testing

Per quanto riguarda i test da compiere, ci siamo soprattutto attenuti ai test case proposti nelle task card, ma abbiamo anche compiuto alcuni test sul momento. Come spiegato nei paragrafi precedenti, il testing è stato svolto a diversi livelli:

- Partendo dal livello più basso, ovvero quello degli smart contract, dopo aver creato i vari contratti, abbiamo utilizzato Remix IDE per poter testare ogni funzione. Infatti, è possibile inizializzare una Blockchain di prova e poter utilizzare, tramite interfaccia grafica, le funzioni dei vari contratti.
- Successivamente, dopo la creazione delle API REST, siamo andati a testarle le varie funzioni di POST, GET o PUT tramite Postman, che come descritto sopra permette di andare a testare le API sempre tramite un'interfaccia grafica.
- Infine, l'ultima tipologia di test, l'abbiamo fatta lato frontend, utilizzando, questa volta, la nostra interfaccia per andare a verificare che le chiamate a funzioni producessero l'effetto desiderato.

13. Sicurezza

13.1. Sicurezza nel frontend

In modo da garantire un maggior isolamento, siamo andati ad evitare punti di accesso inattesi e siamo andati a limitare i punti di accesso.

Infatti, la maggior parte dei punti di accesso della nostra applicazione sono dati da buttoni, che vengono controllati da specifici controlli sul frontend, sul backend e a livello di smart contract.

Inoltre, siamo andati anche ad evitare possibili SQL Injection. Infatti, la nostra applicazione presenta, ovviamente, anche form di input, come ad esempio nel login. Quindi, siamo andati a controllare le stringhe immesse all'interno di ogni form di input, in modo da bloccare comportamenti malevoli da parte dell'utente.

13.2. JWT

Effettuando una richiesta POST alla pagina /login con username e password corretti il server restituirà un JWT (JSON Web Token), diverso per ogni accesso, che verrà utilizzato come autenticazione per le successive richieste.

Tale token è composto da tre stringhe divise da un punto e codificate in Base64, per una idonea codificazione dei caratteri binari; queste stringhe sono rispettivamente **l'header**, **il payload** e **la firma** del token.

L'header trasporta le informazioni riguardanti l'algoritmo utilizzato nella codifica della firma, mentre il payload contiene dati (non codificati) che possono essere utili al server nel processo di autenticazione (inclusa la durata della validità del token).

La firma, invece, è generata tramite la concatenazione dei due elementi precedenti, utilizzando una chiave conosciuta solamente dal server. Ciò permette al server di riconoscere univocamente la provenienza di un token, senza alcuna possibilità di falsificazione in mancanza della chiave.

Quindi, una volta effettuato il login il client dovrà esibire il token in ogni successiva richiesta che effettuerà. Esso verrà inserito nell'header HTTP chiamato "Authorization" con la tipologia "Bearer", e in assenza di tale voce o in caso di credenziali non valide, il server risponderà con uno stato HTTP 401, "Unauthorized".

All'interno del token sono inserite inoltre informazioni riguardanti il ruolo dell'utente, per permettere un monitoraggio basato su **Role Based Access Control** (RBAC): ogni utente potrà infatti accedere solamente alle risorse per cui il suo ruolo ha la giusta autorizzazione. Tutti questi controlli vengono effettuati in primis nell'interfaccia utente, e in secondo luogo nel back-end da un apposito middleware. Il controllo nel lato back-end è necessario ed è estremamente più importante, poiché ogni controllo effettuato univocamente sul front-end è facilmente aggirabile da un utente con un minimo di esperienza in termini di sicurezza.

13.3. Sicurezza nella Blockchain

L'utilizzo di una Blockchain come database distribuito garantisce già da sé aspetti fondamentali per la sicurezza. Infatti, questa gode di diverse proprietà, quali:

- **Distribuita:** avere dati distribuiti su più nodi esula dal problema del single point of failure
- **Inviolabile:** il protocollo di consenso interno ad una rete garantisce che un attacco possa avere successo solo se l'attaccante possiede il consenso del 50% +1 dei nodi della rete
- **Privata:** l'utilizzo di una Blockchain Permissioned come Quorum ci ha permesso di restringere l'accesso alla rete ai soli peer autorizzati. Questo è reso possibile grazie all'utilizzo di transazioni e contratti privati che permettono di interagire ai soli nodi specificati nelle transazioni.

Inoltre, una blockchain soddisfa alcuni dei requisiti fondamentali della sicurezza (Modello CIA e Modello AAA), tra cui:

- **Confidenzialità:** grazie al meccanismo di offuscamento delle transazioni, i dati contenuti all'interno del corpo di una transazione saranno accessibili ai soli utenti autorizzati. Un attaccante potrà al massimo accedere ad una transazione, ma vedrà i dati completamente criptati.

- **Integrità:** un certo attore può accedere e modificare solamente i dati a cui è autorizzato ad accedere.
- **Disponibilità:** i dati sono accessibili e disponibili in qualsiasi momento e per qualunque peer facente parte della rete autorizzata. Questa proprietà è ovviamente possibile grazie alla natura distribuita della Blockchain.
- **Autenticazione:** si riferisce all'autenticazione di ciascun nodo, che è resa possibile dalla coppia di chiavi pubbliche-private.
- **Autorizzazione:** si riferisce al processo di autorizzazione ad accedere alla rete e ai dati al suo interno. Questo è ovviamente garantito dalla logica permissioned di Quorum.
- **Contabilità/Non ripudio:** è uno tra i maggiori vantaggi dell'utilizzo delle Blockchain, in quanto qualsiasi azione intrapresa da un utente è immutabile.
- **Autenticità:** in modo da garantire l'autenticità dei dati, sono utilizzati meccanismi di cifratura con una coppia di chiavi pubbliche-private per ogni nodo interno alla rete.

13.4. Sicurezza negli smart contract

Per quanto riguarda gli aspetti di sicurezza e di buona programmazione del software, abbiamo inserito alcuni dei design pattern e pattern di programmazione che sono di comune utilizzo nella scrittura degli smart contract, con l'utilizzo del linguaggio di programmazione Solidity.

Questi pattern sono stati pensati per Ethereum, quindi la maggior parte riguardano aspetti legati al trasferimento sicuro di Ether, in modo da salvaguardare da comportamenti malevoli di utenti, come il double spending. Tuttavia, siamo riusciti ad utilizzarne qualcuno, legato ad aspetti più generali della sicurezza e dell'ottimizzazione degli smart contract.

13.1.1. Behavioral Pattern Guard Check

È un design pattern comportamentale che ha come scopo controllare l'inserimento di dati non corretti in modo da garantire che la logica dello smart contract funzioni come voluto.

Il controllo viene fatto tramite specifiche funzioni che, nel caso di circostanze non volute, ripristinano lo stato precedente, eliminando tutte le modifiche apportate durante la transazione che genera l'errore (infatti, ogni transazione è atomica per l'Ethereum Virtual Machine, ovvero la parte che si occupa di gestire le transazioni).

In particolare, il pattern Guard Check viene utilizzato per:

- Controllare e validare input provenienti dall'utente
- Verificare lo stato del contratto prima di eseguire la logica del contratto
- Escludere condizioni che non sono possibili

Per implementare questo design pattern, abbiamo utilizzato soprattutto la funzione require di Solidity.

`require(bool condition, string message)` prende in input due parametri:

- Il primo che è la condizione che è richiesta
- Il secondo è l'eventuale messaggio di errore che deve essere restituito nel caso in cui la condizione non sia verificata

Nel caso in cui la condizione non sia verificata viene lanciata in automatico la funzione di revert che elimina le ultime modifiche apportate e ripristina lo stato immediatamente precedente.

Alcuni esempi di utilizzo del require sono:

- Richiedere che il codice del lavoro inserito dall'utente sia già presente all'interno della Blockchain, dovendo essere stato inserito in precedenza da parte del RUP.

```
1. require(  
2.   lavori[i].codice_lavoro != code,  
3.   "Il codice del lavoro che è stato inserito non corrisponde ai codici possibili"  
4. );
```

- Richiedere che un certo indirizzo passato sia valido. Infatti, in presenza di un qualche errore, l'EVM andrebbe a passare alla funzione che richiede un certo indirizzo, l'indirizzo nullo. In questo modo, se l'indirizzo passato è proprio 0, si va a generare un errore.

```

• require(
•     _indirizzo != address(0),
•     "L'indirizzo passato non è valido"
• );
```

In ottica di verificare lo stato del contatto prima di eseguire la logica del contratto, oltre ad utilizzare vari require, abbiamo anche implementato alcune funzioni che vadano a controllare specifiche variabili.

Ad esempio, in modo da controllare il superamento di una certa soglia (data dalla rata di acconto assegnata), abbiamo definito una variabile di soglia che va ad assumere tre differenti valori:

- Superata: nel caso in cui la soglia corrente sia stata superata
- Non superata: nel caso in cui non sia invece stata superata
- Da compilare: nel caso in cui sia stata superata una certa soglia, quindi sia possibile al direttore dei lavori emanare uno stato di avanzamento, ma questo non l'abbia ancora emanato

Grazie a diverse funzioni, si va a controllare e settare di conseguenza il valore di questa variabile, in modo da avere sempre un adeguato controllo dello stato del contratto e quindi non permettere azioni non possibili.

13.1.2. Behavioral pattern State Machine

Attraverso il pattern State Machine, si vuole consentire ad uno smart contract di passare attraverso diverse fasi durante il suo ciclo di vita e di poter esporre differenti funzionalità nelle differenti fasi.

Nel nostro caso, abbiamo avuto bisogno di questo pattern, in quanto all'interno del contratto Misure.sol abbiamo due differenti fasi:

- La prima in cui il Responsabile Unico del Procedimento può solamente inserire le lavorazioni da dover compiere durante la costruzione dell'opera edilizia e la rata di acconto utile al calcolo della soglia monetaria da dover superare per poter emanare un SAL. Durante questa fase né il RUP, né gli altri attori hanno accesso alle altre funzioni del contratto.

- La seconda fase in cui, in seguito alla terminazione dell'inserimento dei dati da parte del RUP, ora sono accessibili le altre funzioni.

Quindi per poter far utilizzare le funzioni di inserimento e modifica sia delle lavorazioni che della rata di acconto, abbiamo definito la variabile inserimenti_terminati, inizialmente settata a false, che in seguito alla conclusione dei vari inserimenti verrà invece settata a true.

Abbiamo poi definito il modifier onlyFaseUno, ovvero un controllore che se posto nella definizione di una funzione, va ad eseguire prima il controllo che trova al suo interno e se il controllo risulta corretto, chiama la funzione chiamata. Nel nostro caso, il modifier va a controllare che la variabile inserimenti_terminati sia ancora posta a false, ovvero ci si trovi ancora nella fase iniziale e in tal caso va a chiamare la funzione per cui è stato definito.

```

1. modifier onlyFaseUno() {
2.     require(
3.         inserimenti_terminati == false,
4.         "Non è possibile utilizzare questa funzione in questa fase del contratto"
5.     );
6.     _; //indica dove deve essere posta la funzione chiamata
7.

```

Un esempio di funzione in cui è stato definito è la funzione di inserimento di una lavorazione, in cui si controlla prima questo modifier, poi il modifier OnlyRup (ovvero che l'utente che esegue la funzione sia il RUP) e soltanto in caso di successo, vengono svolte le operazioni all'interno della addLavoro.

```

1. function addLavoro (string memory _codice, string memory _nome, uint _costo) public onlyFaseU
   no onlyRup {}

```

13.1.3. Security Pattern Access Restriction

Access Restriction serve per restringere l'accesso di alcune funzionalità e dati, appunto, solamente al verificarsi di alcune condizioni o per alcuni utenti.

Ad esempio, una variabile, se definita come pubblica, può essere letta da chiunque, quindi qualsiasi attore all'interno di una rete potrebbe accedere allo stato del contratto e nel caso modificare in modo malevolo lo stato del contratto. Tutto ciò viene risolto andando a definire ogni variabile come privata ed è ciò che abbiamo realizzato in ognuno dei nostri smart contract. Stessa cosa riguarda per le funzioni che implementano la logica del contratto. Se rese pubbliche, chiunque potrebbe accidentalmente chiamare quelle funzioni. Se invece vengono definite come private, solamente le funzioni interne ad uno stesso contratto possono utilizzarle.

D'altra parte, alcune funzioni devono essere invece protette dal possibile utilizzo malevolo di utenti non autorizzati all'uso di quelle funzionalità. Un esempio, nel nostro caso, sono le funzioni di inserimento di dati all'interno di un certo documento. Infatti, alcuni documenti sono compilabili da un solo attore. Se lasciate pubbliche, quelle funzioni sarebbero invece utilizzabili da chiunque. Per questo motivo abbiamo definito dei modifier per ogni attore coinvolto nel contratto d'appalto e li abbiamo assegnati alle funzioni che gli competono.

13.1.4. Economic Pattern String Equality Comparison

Questo pattern si è rivelato molto importante all'interno dei nostri smart contract per andare a comparare due stringhe. Infatti, inizialmente avevamo preferito l'utilizzo della libreria `StringUtils`, implementata dalla Ethereum Foundation, che contiene al suo interno funzioni utili proprio a controllare l'uguaglianza tra due stringhe, dato che non è possibile ovviamente controllarle con un operatore di uguaglianza. La funzione della libreria va a fare un controllo a coppie di caratteri, quindi per stringhe molto lunghe, il consumo di gas e quindi la complessità computazionale del contratto aumenta di molto. Per questo motivo, dovendo andare a

fare molte comparazioni tra stringhe in modo da controllare valori già inseriti, ci è risultato conveniente l'utilizzo di String Equality Comparison.

Il pattern in questione utilizza un trick che si compone di due step:

- Inizialmente trasforma le stringhe in bytes e riesce in questo modo a confrontare la lunghezza dei due bytes. Se la lunghezza è diversa, restituisce errore, altrimenti passa allo step due.
- Nello step due va ad utilizzare la funzione crittografica built-in Kekkak-256, che prende un input una certa stringa e va a generare un hash di questa. A questo punto i due hash sono comparabili tra di loro e in caso di un matching completo tra gli hash, viene restituito true.

```
1. function compareStrings(string a, string b) public returns (bool) {  
2.     if(bytes(a).length != bytes(b).length) {  
3.         return false;  
4.     } else {  
5.         return keccak256(a) == keccak256(b);  
6.     }  
7. }
```