POLITECNICO DI TORINO

Corso di Laurea Magistrale
in Ingegneria Informatica

Tesi di Laurea Magistrale

RESOURCES MANAGEMENT

L'agenda per l'organizzazione aziendale

Relatore
Prof. Maurizio Morisio

Candidato
Giorgia Ricotta

Anno Accademico 2018/2019

1. INTRODUZIONE

1.1 L'azienda

SAN S.r.l. è un'impresa che opera in Piemonte nel settore ICT da circa 25 anni.
La sua area di competenza principale è la progettazione e lo sviluppo di software per le imprese, alle quali offre la possibilità di ottimizzare i propri processi gestionali interni per guadagnare efficienza, ridurre gli errori e di conseguenza aumentare la competitività sul mercato.
Per SAN l'attività di ottimizzazione dei processi aziendali inizia sempre con un'analisi della condizione esistente, mappando il percorso delle informazioni e delle loro trasformazioni ad opera dei vari attori e delle varie funzioni coinvolte; per questo SAN dispone di alcune figure professionali senior che hanno la capacità di eseguire questa analisi, tramite le quali proporre poi soluzioni migliorative che nella maggior parte dei casi si traducono in:

- Ottimizzazione dei processi (quindi miglioramenti organizzativi)
- Adozione di tecnologia ICT per l'informatizzazione dei processi (quindi miglioramenti tecnologici e di integrazione)

La tecnologia per SAN è quindi strumento per il raggiungimento dei due obiettivi esposti ai punti precedenti.
Pur avendo le necessarie certificazioni da parte di alcune aziende nel settore (Apple, IBM, Microsoft, etc.) SAN afferma di scegliere sempre la migliore applicabile allo specifico caso in esame, prendendo ogni volta in considerazione anche le soluzioni offerte dal mondo open-source che negli ultimi anni sta occupando un ruolo crescente nei diversi progetti, sia come strumento di sviluppo che come insieme di oggetti funzionali già pronti ed integrabili.

SAN lavora quindi producendo progetti in ambito esclusivamente B2B; i settori in cui si è maggiormente specializzata negli anni sono quello sanitario (*healthcare*), delle *utilities* (acquedotti in particolare) e dell'industria aeronautica. L'inevitabile acquisizione di competenza specializzata nella gestione dei processi in questi settori ha portato SAN a ricevere incarichi di IT management presso aziende clienti, con responsabilità gestionali che vanno oltre l'offerta di soluzioni proprie ma anche di analisi, confronto ed integrazione con prodotti e servizi di altre aziende presenti sul mercato.

Da circa un paio di anni, SAN sta cercando di valorizzare la propria esperienza affiancando alla sua usuale produzione di progetti anche l'offerta di uno o più prodotti, introducendo al proprio interno una nuova modalità organizzativa in grado di gestire tutte le problematiche relative alla produzione (controllo, manutenzione, evoluzione, CRM[1]).
Questa tesi riguarda un progetto che per SAN ha la possibilità di diventare un prodotto, avendo incontrato tre clienti che, operando nello stesso ambito, presentano le stesse macro-esigenze.

[1] CRM: Customer Relationship Management, gestione delle relazioni con la Clientela

1.2 Lo scopo della tesi

Il lavoro consiste nell'affiancamento ad un gruppo di lavoro SAN che si è occupato di sviluppare un progetto (nello specifico, nell'ambito del Facility Management) con la prospettiva di elevarlo a prodotto.
L'ambito di applicazione del lavoro si scosta da quello usuale dell'azienda, principalmente per due ragioni:

- È in un ambito diverso dal quello usuale, cioè al di fuori del contesto sanitario, industriale, utilities.
- Offre la possibilità di sperimentare la produzione di un prodotto spendibile su più clienti.

Il lavoro è consistito nell'ideare un applicativo di gestione del lavoro per aziende facenti parte dell'ambito del Facility Management.
Sul mercato odierno esistono già molti applicativi in grado di fornire questo tipo di servizio, ma, come abbiamo potuto constatare (sotto informazione e indagine dei clienti), risultano incompleti dal punto di vista degli utilizzatori intervistati (rispetto a quello che si ricerca nei servizi che dovrebbe offrire), con ristrettezze e vincoli che ne rendono l'uso poco fluido (si devono integrare più applicativi tra loro), poco dinamico e flessibile rispetto alle esigenze presenti.
Per questi motivi è stata avanzata la suddetta proposta: progettare e ideare un applicativo ad hoc che superasse le criticità emerse dall'uso delle soluzioni presenti sul mercato, che offrisse tutti i servizi necessari a migliorare e rendere più veloce il lavoro dell'azienda. Una soluzione customizzata quel tanto da differenziarsi dalle altre ma che non diventasse troppo mirata per un solo acquirente.
Quindi è stato ideato un applicativo che potesse essere raggiungibile ovunque e in qualsiasi momento, che potesse contenere tutte le informazioni necessarie alla pianificazione e gestione del lavoro all'interno dell'azienda, dinamico e flessibile ai cambiamenti che avvengono nella riorganizzazione del lavoro.
Dopo aver eseguito le dovute analisi, è stata sviluppata l'applicazione web "Resources Management".

2. AMBITO DI OPERATIVITÀ DELLE AZIENDE COINVOLTE

2.1 Facility management

Il facility management nasce per indirizzare tutte quelle attività che non riguardano specificatamente il core business di un'azienda. Occuparsi e preoccuparsi di tutte quelle attività accessorie rispetto al business dell'azienda comporta uno sgravio di risorse (sia a livello mentale, sia a livello fisico, sia a livello di costi) che così possono essere focalizzate sull'unico centro di interesse dell'organizzazione. L'azienda, che offre questi servizi, li progetta e li personalizza per soddisfare le esigenze dell'azienda che li richiede.
Il Facility Management è il processo di progettazione, implementazione e controllo attraverso il quale le facility (ovvero gli edifici e i servizi necessari a supportare e facilitare l'attività dell'azienda) sono individuate, specificate, reperite ed erogate allo scopo di fornire e mantenere quei livelli di servizio in grado di soddisfare le esigenze aziendali, creando un ambiente di lavoro di qualità con una spesa il più possibile contenuta.

Il facility management integra i principi della gestione economica e finanziaria d'azienda, dell'architettura e delle scienze comportamentali e ingegneristiche.
Essendo un approccio integrato, presuppone lo sviluppo e l'implementazione di politiche, standard e processi che supportano le attività primarie, rendendo l'organizzazione in grado di adattarsi ai cambiamenti e di migliorare l'efficacia.

I tre aspetti principali che caratterizzano la disciplina del Facility Management sono quello strategico, quello analitico e quello gestionale-operativo.
L'Aspetto Strategico concerne ogni decisione relativa alla politica di gestione e reperimento dei servizi, di distribuzione delle risorse da impiegare per supportare gli obiettivi corporate (predisposizione e gestione del budget, ripartizione dei costi, ecc.), di scelta del fornitore, ecc.
L'Aspetto Analitico è relativo alla comprensione delle necessità dei Clienti Interni relative ai servizi, al controllo dei risultati della gestione e dell'efficienza nell'erogazione del servizio, all'individuazione di nuove tecniche e tecnologie che supportino il business aziendale. Si tratta quindi di un aspetto fondamentale per far sì che il Facility Management contribuisca, con i fatti, al conseguimento degli obiettivi dell'azienda.
L'Aspetto Gestionale-Operativo riguarda la gestione e il coordinamento di tutti i servizi complessivamente intesi (non dei singoli servizi) e include la definizione di sistemi, procedure, l'implementazione e reingegnerizzazione dei processi di erogazione.

2.2 Le aziende coinvolte e le esigenze raccolte

L'analisi dei processi e lo sviluppo dell'applicazione hanno visto come fonte principale tre aziende facenti parte di questo settore ma con caratteristiche diverse tra di loro, interessate a rendere più efficienti i processi gestionali interni.
Il primo cliente, *A*, è una piccola azienda, con sede a Grugliasco: essa offre un insieme di servizi mirati sia alle aziende che ai privati. Questo cliente è stato fondamentale, ci ha fornito un feedback utile sui prodotti disponibili sul mercato, sulle criticità riscontrate che l'hanno portato a chiedersi se ci fosse un'azienda disponibile a creare un prodotto completo che superasse queste criticità. I dati forniti ci hanno permesso di scostarci dalle soluzioni sul mercato in modo da offrire una soluzione vantaggiosa.
Il secondo cliente, l'azienda *B*, è situata in Lombardia: è una società cooperativa di servizi che fornisce risorse operative per tutte le attività logistiche e manodopera generica o qualificata.
Il terzo cliente, l'azienda *C*, ha giocato un ruolo importante: ci ha permesso di etichettare nel modo corretto (così da comprendere più approfonditamente il lavoro) l'ambito applicativo di queste aziende; il cliente ha sede a Favria (TO) e fornisce gli stessi servizi delle altre due aziende ma su scala maggiore.

La parte più complessa è risultata essere l'analisi del modus operandi di tutti e tre i clienti e dei loro processi per trovare il filo conduttore che portasse ad una soluzione sia comune ma anche vantaggiosa per ognuna di esse e per i futuri clienti.
Da questa analisi sono emerse diverse esigenze:

- possibilità di inserire e consultare tutti i tipi di dati riguardanti il lavoro in un unico applicativo, in modo che il lavoro potesse diventare più efficiente e veloce;
- poter raggiungere l'applicativo da qualsiasi dispositivo in qualsiasi momento;
- offrire l'accesso customizzato in base al ruolo;
- monitorare l'attività dell'operatore;
- poter programmare gli appuntamenti in modo dinamico;
- poter modificare/spostare gli appuntamenti in base alle necessità del momento;
- assegnare il lavoro in maniera efficiente avendo una panoramica aggiornata della situazione attuale;
- avere uno strumento che non fosse troppo restrittivo e non avesse troppi vincoli.

2.3 Analisi di mercato

Nella prima fase di analisi, un team si è occupato di redigere un *Business plan* dettagliato ed accurato.
Nella sezione relativa all'analisi di mercato sono stati analizzati prodotti potenzialmente adatti, già inseriti sul mercato, ad informatizzare i processi di queste aziende; ne sono stati evidenziati punti di forza e punti di debolezza sia dal punto di vista lavorativo che economico.
Nella tabella di seguito è stato sintetizzato il risultato dell'analisi di alcuni prodotti già presenti sul mercato, evidenziandone punti di forza e di debolezza:

PRODOTTO	PUNTI DI FORZA	PUNTI DI DEBOLEZZA
Trello (Atlassian)	- è uno strumento gratuito - condivisione semplice con i membri del team - creazione veloce di un board per il progetto con le liste relative alle fasi - semplicità nello spostamento dei tasks all'interno delle liste e personalizzazione di essi	- supporto limitato - non c'è modo di ordinare le attività per scadenza o stato di avanzamento - le attività possono essere inserite sul calendario ma non vi è la distinzione per ore
Wrike (Wrike)	- lungo elenco di caratteristiche, incluse le scadenze visivi, editing di file, moduli di richiesta automatizzati, ecc. - tempo di monitoraggio e moduli di raccolta dati dinamici funzionano bene per la gestione delle attività di customer support - controllo accesso basato sui ruoli e	- pianta libera molto limitato - complesso da usare - contiene molte funzionalità, ma non rispecchia a pieno le esigenze emerse

	autenticazione dei dati	
Microsoft Project (Microsoft)	- ottime funzionalità sia per la gestione dei progetti che per la gestione delle risorse - di semplice utilizzo	- costo elevato per ottenere più funzionalità e l'uso simultaneo su più dispositivi - non adatto alle esigenze presenti, non totalmente adatto ad aziende che forniscono questo tipo di attività

Tabella 1: Prodotti

Questi prodotti offrono funzionalità differenti tra loro; inoltre tutti quanti presentano molti punti di forza e rendono l'organizzazione del lavoro molto più efficiente, tuttavia non è stato trovato un prodotto che racchiudesse tutte le caratteristiche cercate.
Nello specifico, un prodotto avrebbe potuto, potenzialmente, essere valido per un aspetto del lavoro ma non per altri e, allo stesso tempo, un altro prodotto avrebbe potuto esserlo per altri aspetti ancora; quindi non è stato trovato un prodotto che potesse soddisfare tutte le esigenze segnalate dalle aziende.

Raccolti questi dati e messi a confronto con le esigenze delle aziende, si è deciso di intraprendere la strada dello sviluppo ad hoc, non effettuando un *Tailor made*[2], bensì creando un prodotto da poter essere inserito sul mercato con quelle determinate caratteristiche avendo come base il filo conduttore.

[2] Prodotto customizzato per un unico cliente

3. ANALISI

Il primo passo fatto è stato quello di analizzare nel dettaglio i processi che avvengono all'interno delle aziende, quindi abbiamo chiesto un'analisi dettagliata di come svolgono il loro lavoro, quali sono gli attori coinvolti nei vari processi: dall'acquisizione di un possibile nuovo cliente, fino all'organizzazione del lavoro accettato.

3.1 GLI ATTORI

In questo scenario, gli attori coinvolti – nel senso di ruolo – comuni tra le aziende, sono:

1. Il cliente;
2. Il coordinatore;
3. L'ufficio tecnico;
4. Il capocantiere;
5. L'approvatore;
6. L'assegnatore;
7. L'operatore.

Naturalmente ciascun ruolo può essere assunto da più persone o utenti utilizzatori.

Nome ruolo	**Descrizione**
Cliente (CLI)	Il cliente è la persona che richiede uno o più servizi all'azienda
Coordinatore (CO)	Il coordinatore è la persona che ha il compito di interagire con il cliente per discutere dei lavori che sono da svolgere, gestirli; gestire gli acquisti e gli ordini
Ufficio tecnico (UT)	L'ufficio tecnico svolge attività di supervisione e può creare preventivi per i clienti
Capocantiere (CC)	Il capocantiere è la persona che si occupa di creare la RDA (in alternativa al coordinatore) a fronte di un preventivo approvato.
Approvatore (AP)	L'approvatore è la persona che si occupa di approvare o meno le RDA.
Assegnatore (AS)	Si occupa di creare e assegnare i clienti al coordinatore

Operatore (OP)	Si occupa di svolgere i lavori presso i clienti

Tabella 2: Ruoli

Questa suddivisione in ruoli, o identificazione di attori, si rende necessaria perché ciascuno ha uno o più compiti differenti da svolgere e quindi bisogna prevedere uno strumento informatico che deve presentarsi a ciascun ruolo offrendo le funzionalità a lui riservate.

3.2 I PROCESSI

I processi identificati sono i seguenti:

1. Processo di creazione di un nuovo preventivo: a fronte di una nuova richiesta da parte di un cliente (nuovo o già presente a sistema) il coordinatore o l'ufficio tecnico creerà il preventivo;
2. Processo di creazione di un sopralluogo: il coordinatore può decidere di eseguire un sopralluogo prima di creare un preventivo (in modo da verificare tutto il necessario) oppure a preventivo fatto per definire i lavori;
3. Processo di creazione e pianificazione del/dei lavori: collegato al primo, tratta la creazione della commessa cliente (a partire dal preventivo approvato) con successiva suddivisione in commesse interne e creazione della/e RDA associata/e;
4. Processo di assegnazione dei lavori: collegato ai primi due, tratta l'assegnazione delle commesse interne agli operatori e la pianificazione temporale definitiva;
5. Processo di check-in e check-out di un operatore: l'operatore esegue il check-in e check-out giornaliero a distanza (simulazione dell'uso del badge nell'azienda).

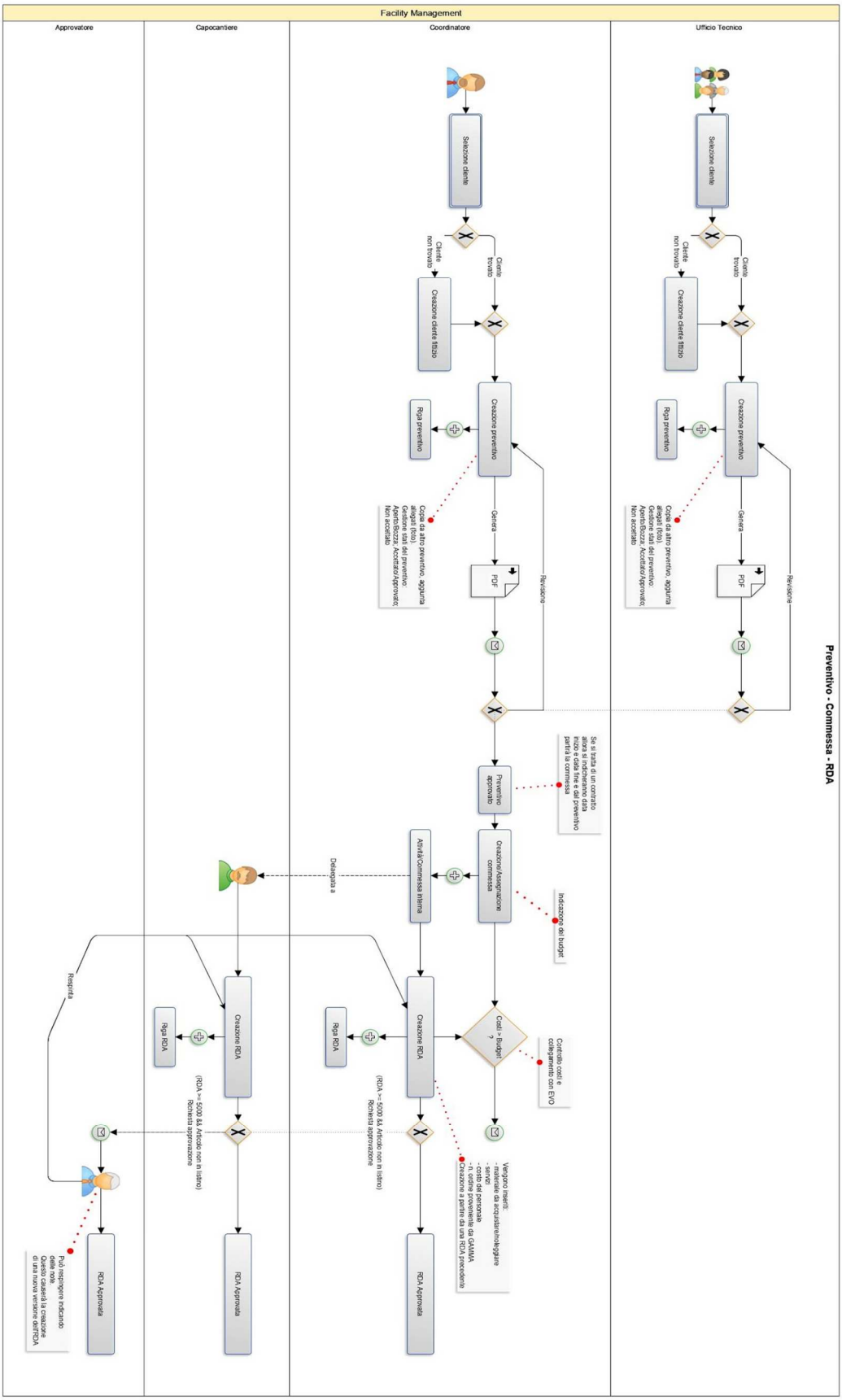

Figura 1: Primo e terzo processo

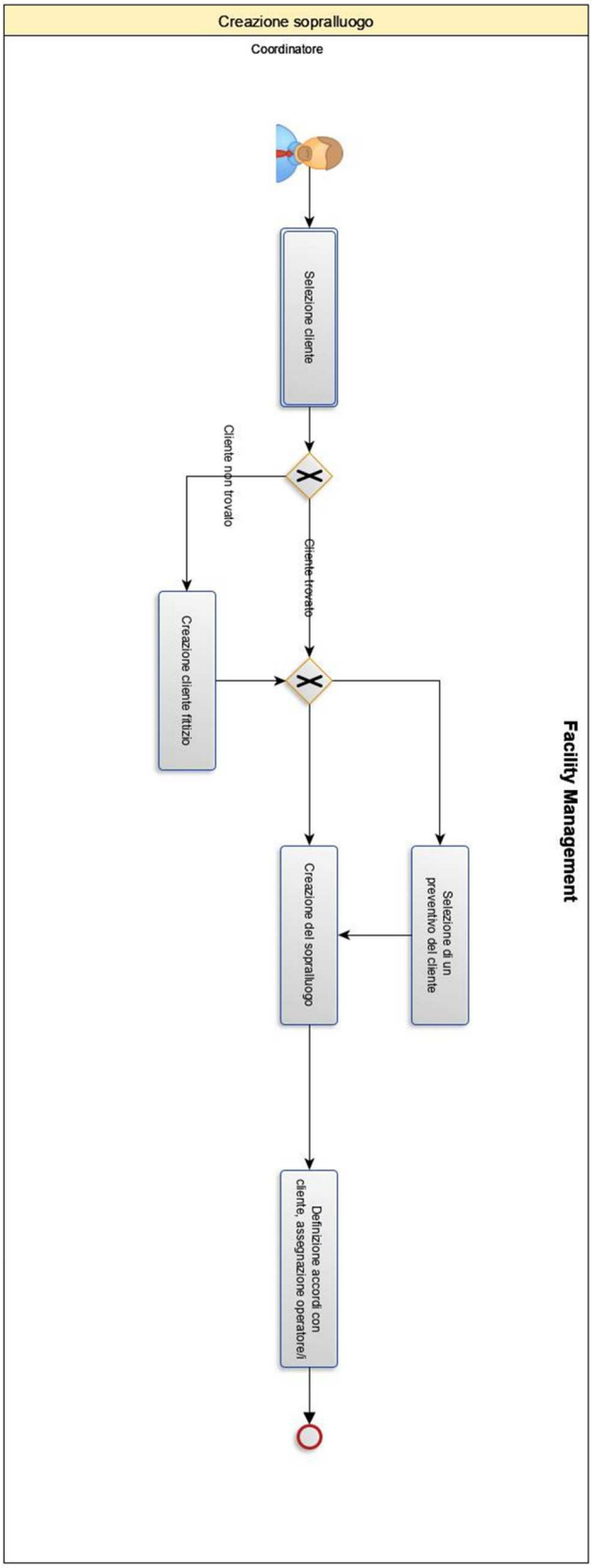

Figura 2: Secondo processo

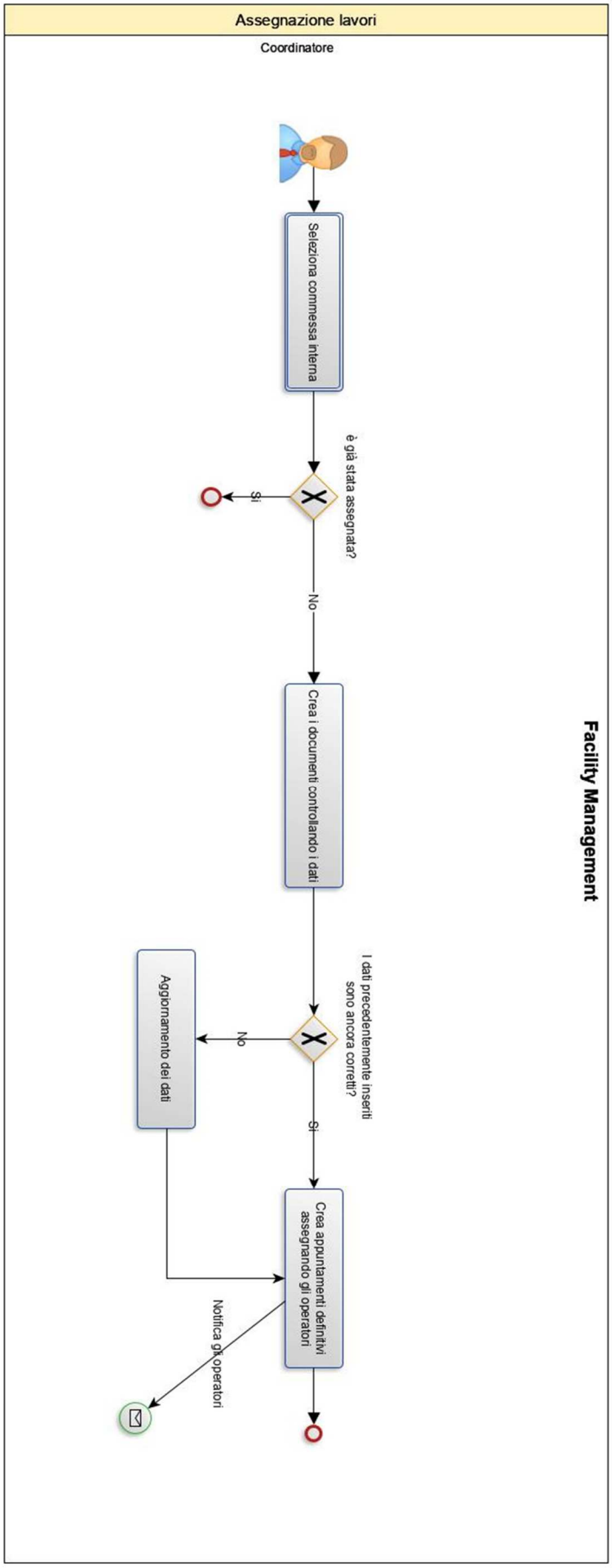

Figura 3: Quarto processo

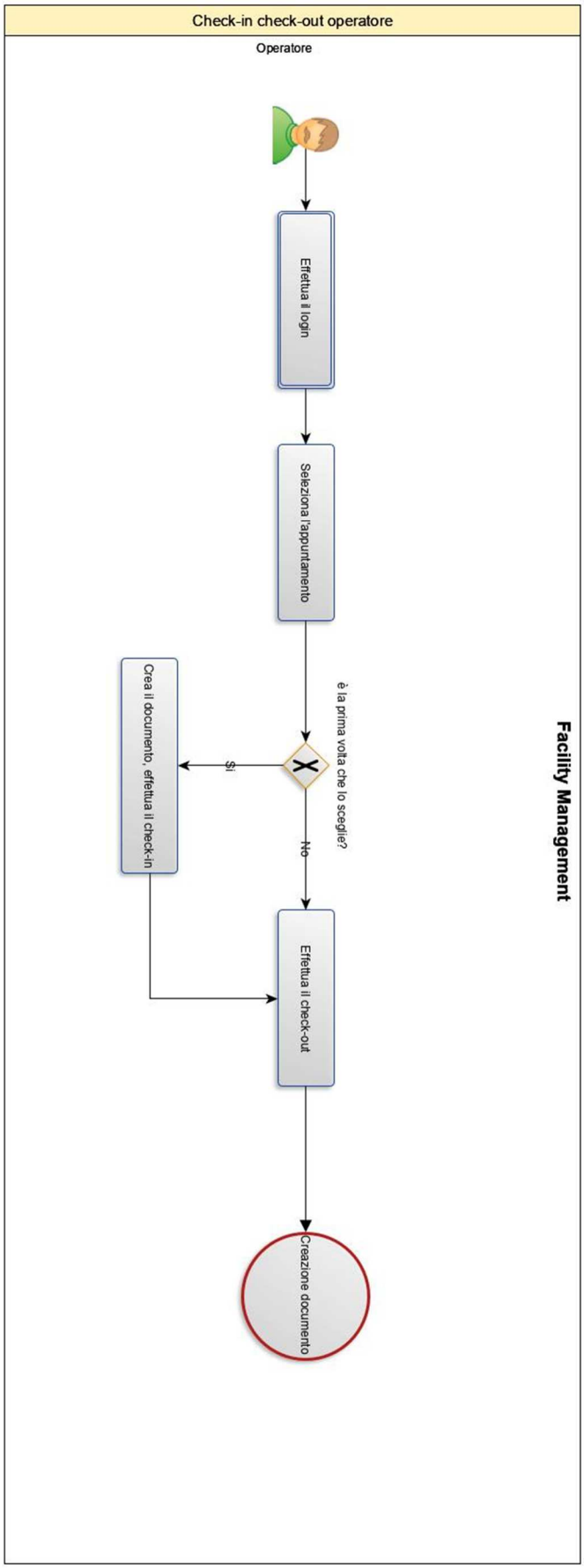

Figura 4: Quinto processo

3.2.1 Processo di creazione di un nuovo preventivo (1)

- Il coordinatore/ufficio tecnico (CO) riceve una nuova richiesta da parte di un cliente: innanzitutto, egli deve verificare se il cliente è nuovo (e quindi deve prima essere inserito nel sistema creando un cliente fittizio) o se è già presente nel sistema.
- Una volta verificato, il CO procede con l'analisi della richiesta per poter formulare un preventivo da presentare al cliente (eventualmente copiando i dati da un preventivo già creato precedentemente); successivamente procede con la creazione del PDF di esso per inviarlo al cliente.
- Il cliente, valutato il preventivo, può non accettarlo e richiederne una revisione (e quindi si ritorna ai punti precedenti) oppure può accettarlo.
- Se il preventivo viene accettato, viene creata la commessa cliente oppure viene assegnata una commessa già esistente.

3.2.2 Processo di creazione di un sopralluogo (2)

- Il coordinatore sceglie il cliente, se è già inserito a sistema, lo seleziona altrimenti lo crea e lo inserisce.
- Se il sopralluogo deve essere eseguito prima di creare un preventivo, il CC semplicemente crea il sopralluogo dal cliente.
- Se il sopralluogo viene eseguito post creazione preventivo, il CC va a prendere il preventivo in questione e crea il sopralluogo.

3.2.3 Processo di creazione e pianificazione del/dei lavori (3)

- Creata la commessa cliente, viene indicato il budget, vengono create tante commesse interne quanti sono i lavori da svolgere che sono stati concordati.
- Per ogni commessa interna, il CO o il Capocantiere (CC) (a cui, eventualmente, viene delegato il lavoro al posto del CO) creano la RDA associata, indicando: il materiale da acquistare/noleggiare e i servizi. Nel caso in cui il costo supera una certa cifra e sono presenti articoli non in listino, la RDA necessiterà di approvazione da parte dell'Approvatore (AP), che eventualmente può respingere la richiesta.
- Se la RDA viene respinta, automaticamente viene creata una nuova versione della RDA.

3.2.4 Processo di assegnazione dei lavori (4)

- Il CO o il CC reperisce ogni commessa interna che deve ancora essere assegnata.
- Per ogni commessa interna, il CO o il CC crea tutti gli appuntamenti necessari.
- Per ogni appuntamento, il CO o il CC, innanzitutto, conferma o modifica i dati temporali trascinati dalla commessa interna (essenziali per poter creare gli appuntamenti); poi va a selezionare gli operatori a cui assegnerà questo lavoro (tanti quanti pensa siano necessari).

3.2.5 Processo di check-in e check-out di un operatore (5)

- L’operatore si collega alla piattaforma ed effettua il login con username e password.
- Accede all’area dedicata e, se si tratta della prima volta che esegue questa operazione nella giornata, creerà il documento di check-in in uno specifico luogo; per quanto riguarda invece la data e l’ora, saranno presi in automatico in modo che non possano essere manomessi.
- Alla fine della giornata/lavoro, l’operatore rieffetturà il login e andrà a creare il documento di check-out (altro non è che reperire i documenti di check-in a cui verrà apportata la data e ora di check-out).

4. TECNOLOGIA UTILIZZATA

È stato costituito un team di sviluppo per decidere le soluzioni software più adatte e procedere con lo sviluppo della soluzione stessa: l'idea, dalla quale si è partiti e che è rimasta alla base dell'intero progetto, è fornire una soluzione multipiattaforma e accessibile ovunque. Per cui si è scelto di svilupparla come applicazione web.

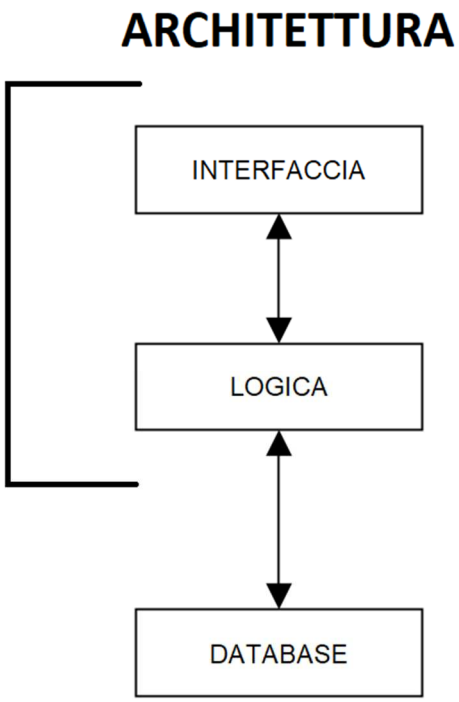

Figura 5: Architettura

4.1 Framework

È stato scelto di utilizzare il framework Ruby On Rails per sviluppare la parte logica dell'applicazione e l'interfaccia lato utente. Questo framework è completamente open source ed è disponibile con la Licenza MIT permissiva, cioè permette il riutilizzo nel software proprietario sotto la condizione che la licenza sia distribuita con tale software.
Il framework viene aggiornato periodicamente in quanto è utilizzato da alcuni siti noti ed inoltre, lato server è facilmente installabile e di facile utilizzo.
Rails deve gran parte del suo successo al suo design elegante e compatto. Sfruttando l'elasticità del linguaggio Ruby sottostante, di fatto Rails crea un linguaggio specifico di dominio per la scrittura di applicazioni web.
Ruby On Rails si basa sul modello MVC e un design RESTful, infatti è stato uno dei primi framework ad assimilare e implementare completamente lo stile architettonico REST

(REpresentational State Transfer) per strutturare le applicazioni web.
REST è un tipo di architettura per lo sviluppo di sistemi di rete distribuiti e di applicazioni software come internet e le applicazioni web. Anche se la teoria REST è piuttosto astratta, nel contesto delle applicazioni Rails, essa significa che la maggior parte dei componenti delle applicazioni sono modellati come risorse che possono essere create, lette, aggiornate ed eliminate. Tali operazioni corrispondono alle operazioni CRUD dei database relazionali e ai quattro metodi di richiesta HTTP fondamentali: POST, GET, PATCH e DELETE.
Lo stile RESTful dello sviluppo aiuta uno sviluppatore di applicazioni Rails a prendere decisioni sui controller e sulle azioni da scrivere: si crea semplicemente la struttura dell'applicazione utilizzando risorse che vengono create, lette, aggiornate ed eliminate.

Un accenno ai metodi di richiesta fondamentali di HTTP sopra citati:

- GET è l'operazione HTTP più comune, usata per *leggere* dati sul web: significa soltanto "prendi una pagina", e ogni volta che visiti un sito come http://www.google.com/ o http://www.wikipedia.org/ il tuo browser sta sottoponendo una GET request.
- POST è la successiva operazione più comune: è la richiesta inviata dal tuo browser nel momento in cui presenti un form. Nelle applicazioni Rails, le POST request sono solitamente usate per creare cose (nonostante HTTP permetta anche a POST di effettuare aggiornamenti). Per esempio, la POST request inviata nel momento in cui viene presentato un form di registrazione, porta alla creazione di un nuovo user sul sito remoto.
- Gli altri due verbi, PATCH and DELETE, sono progettati per effettuare l'update e il destroy di cose sul server remoto. Queste request sono meno comuni rispetto a GET ed a POST perché i browsers sono incapaci di inviarli spontaneamente, ma alcuni web framework (compreso Ruby on Rails) dispongono di mezzi per farlo, come se i browsers stessero emettendo questo tipo di request. Di conseguenza, Rails supporta tutti e quattro i tipi di request.

Le applicazioni sviluppate con Rails hanno una particolarità: sono tutte organizzate secondo una struttura comune. Questo è una conseguenza del fatto che il comando rails genera automaticamente una serie di directory e file che forniscono una certa linea guida nello sviluppo, linea che, se rispettata, permette a Rails di effettuare molte azioni automaticamente. Questa struttura comune permette anche di comprendere con semplicità il codice di progetti realizzati da altri, in quanto sono organizzati nella stessa maniera.

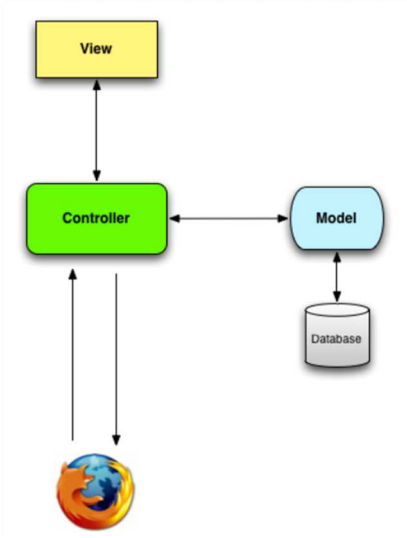

Figura 6: MVC

L'approccio Model-View-Controller, o MVC, è, invece, un metodo per organizzare il codice nei software che interagiscono con delle persone. Tale approccio è diventato ormai comune anche per la realizzazione di applicazioni web. Esso permette infatti di raggiungere un ottimo isolamento tra il codice che gestisce i dati e quello che li presenta all'utente, permettendo di estendere e modificare facilmente un'applicazione in qualsiasi momento.
In altre parole, esso si basa sulla separazione dei compiti fra i componenti software che interpretano tre ruoli principali: il model, la view ed il controller.
Il model fornisce i metodi per accedere ai dati utili all'applicazione. La view visualizza i dati contenuti nel model e si occupa dell'interazione con utenti e agenti. Il controller riceve i comandi dell'utente (in genere attraverso il view) e li attua modificando lo stato degli altri due componenti.

Il model (la M in MVC) è il nucleo dell'applicazione, che incapsulandone lo stato definisce i dati e le operazioni che possono essere eseguite su questi. Esso, quindi, definisce le regole di business per l'interazione con i dati, esponendo alla View ed al Controller rispettivamente le funzionalità per l'accesso e l'aggiornamento. Per lo sviluppo del Model quindi è vivamente consigliato utilizzare le tipiche tecniche di progettazione object oriented al fine di ottenere un componente software che astragga al meglio concetti importati dal mondo reale. Il Model può inoltre avere la responsabilità di notificare ai componenti della View eventuali aggiornamenti verificatisi in seguito a richieste del Controller, al fine di permettere alle View di presentare agli occhi degli utenti dati sempre aggiornati.

La view (la V in MVC) gestisce la logica di presentazione dei dati. Ciò implica che questa deve fondamentalmente gestire la costruzione dell'interfaccia grafica (GUI) che rappresenta il mezzo mediante il quale gli utenti interagiranno con il sistema. Ogni GUI può essere costituita da schermate diverse che presentano più modi di interagire con i dati dell'applicazione.

Il controller (la C in MVC) ha la responsabilità di trasformare le interazioni dell'utente della View in azioni eseguite dal Model: non rappresenta un semplice "ponte" tra View e Model ma realizza la mappatura tra input dell'utente e processi eseguiti dal Model.

Questo schema, fra l'altro, implica anche la tradizionale separazione della *logica di dominio* (chiamata anche *logica di business*) dalla *logica di input* e presentazione associata a un'interfaccia grafica utente (GUI).
Nel caso delle applicazioni web, la *logica di dominio* normalmente consiste in modelli di dati per elementi (come utenti, articoli e prodotti), mentre la GUI è solo una pagina web in un browser web. Quando interagisce con un'applicazione web, un browser invia una richiesta, che viene ricevuta da un server web e inoltrata a un controller Rails, che si occuperà delle azioni successive. In alcuni casi, il controller esegue immediatamente il rendering di una vista, che è un modello che viene convertito in HTML e inviato nuovamente al browser. Più spesso, per i siti dinamici, il controller interagisce con un modello, che è un oggetto di Ruby che rappresenta un elemento del sito e che si occupa della comunicazione con il database. Dopo aver richiamato il modello, il controller esegue il rendering della vista e restituisce la pagina web completa al browser come HTML.

Per accelerare lo sviluppo dell'interfaccia lato utente, inoltre, è stato acquistato un template sviluppato con Ruby On Rails contenente molte funzionalità utili nello sviluppo di un'applicazione web: questo per concentrarsi maggiormente sullo sviluppo della logica dell'applicazione e meno sulla parte grafica.
Questo template si chiama SmartAdmin, è stato scelto in quanto risulta il più completo nel fornire soluzioni moderne e meccanismi reattivi.

4.2 Database

La soluzione di default adottata da Rails per la persistenza dei dati è di utilizzare un database per un'archiviazione di dati a lungo termine, mentre la libreria di base per le interazioni con il database è chiamata *Active Record*.
Active Record comprende una serie di metodi per creare, salvare e recuperare oggetti dati senza utilizzare lo SQL (structured query language) utilizzato dai database relazionali.
Rails, inoltre, presenta una caratteristica chiamata migrazioni che consente di scrivere i *data definition* in puro linguaggio Ruby, eliminando così la necessità di apprendere un linguaggio SQL di *data definition* (DDL). Il risultato è che Rails isola lo sviluppatore quasi completamente dai dettagli dell'archiviazione dati.
Per far sì che questa mappatura tra database ed oggetti funzioni è sufficiente seguire alcune semplici convenzioni, senza la necessità di descrivere esplicitamente le corrispondenze. Il concetto chiave in questo caso, applicato sistematicamente anche nel resto del framework, è compreso nell'acronimo DRY, che sta per 'don't repeat yourself' ('non ripeterti'). Quello che si intende è che ogni informazione deve essere espressa, per quanto possibile, una sola volta,

quindi niente file XML dedicati ad associare tabelle e classi, via le istruzioni per il caricamento dei file, e nessuna descrizione del modello in forma di codice, in quanto è già presente una rappresentazione dello stesso nel database.

È utile spendere due parole sulla nomenclatura: come è già stato detto, Rails preferisce le convenzioni alle configurazioni, e quindi cerca di evitare allo sviluppatore il peso di dover specificare l'associazione tra tabelle e classi. Ovviamente però, *Active Record*, la libreria che gestisce il database, deve avere un modo per capire da solo questa relazione, e questo modo è la lingua inglese.
Infatti, per far sì che *Active Record* trovi da solo le tabelle, è sufficiente che esse siano chiamate con il plurale del nome della classe e che siano scritte in minuscolo.

Vista la scelta del framework, che richiede un database basato su SQL, si è scelto di usare postgreSQL: è open source e più robusto, rispetto agli altri e presenta tabelle grandi. I dati sono conservati come una serie di tabelle con chiavi esterne che servono a collegare i dati correlati. La programmabilità di PostgreSQL è il suo principale punto di forza ed il principale vantaggio verso i suoi concorrenti: PostgreSQL rende più semplice costruire applicazioni per il mondo reale, utilizzando i dati prelevati dal database.
PostgreSQL ha la possibilità di eseguire domande complesse e presenta trigger che vengono attivati automaticamente in ingresso. Questi trigger controllano, confermano, modificano, eliminano o in alternativa inseriscono i dati di riferimento. PostgreSQL inoltre utilizza il metodo MVCC (Multiversion Concurrency Control) per eseguire in modo efficiente l'accesso simultaneo al database.
La flessibilità di PostgreSQL non è solo evidente in termini di funzionalità, espandibilità e personalizzazione: il database offre buon gioco anche per l'installazione di software e hardware; inoltre si basa sul tipico modello Client-Server.

Dopo l'analisi dei processi ed il confronto con i clienti, sono state identificate le informazioni che potessero essere inserite nell'applicativo e consultabili per lo svolgimento del lavoro.
Ne è stato creato il diagramma ER per avere un quadro generale di quali informazioni sarebbero state memorizzate nel database:

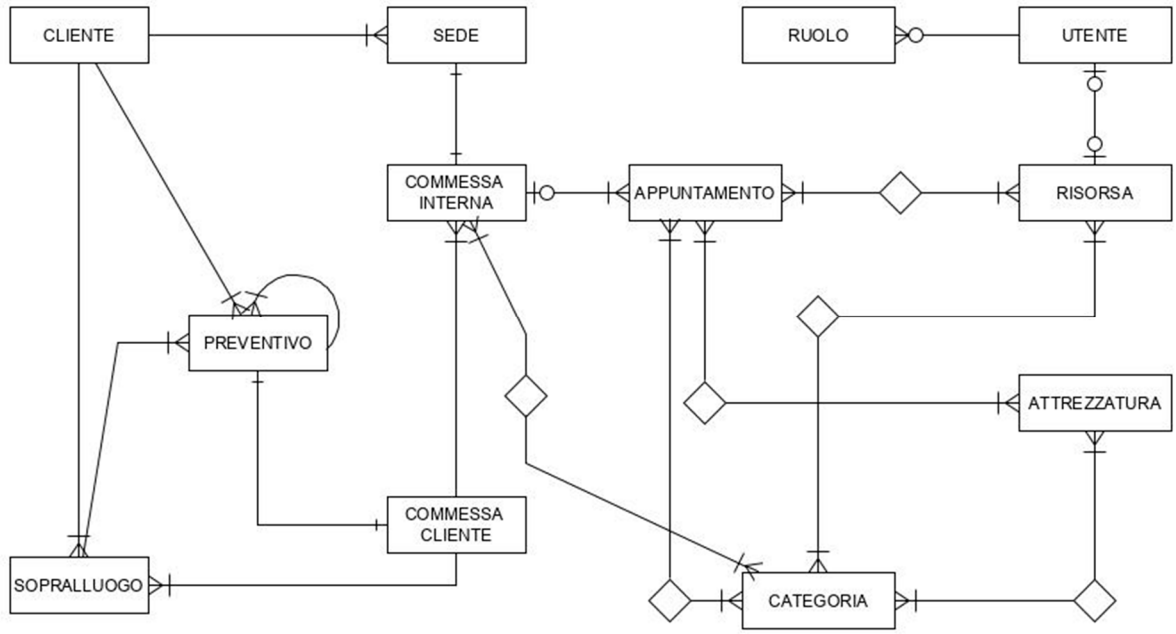

Figura 7: modello ER (1)

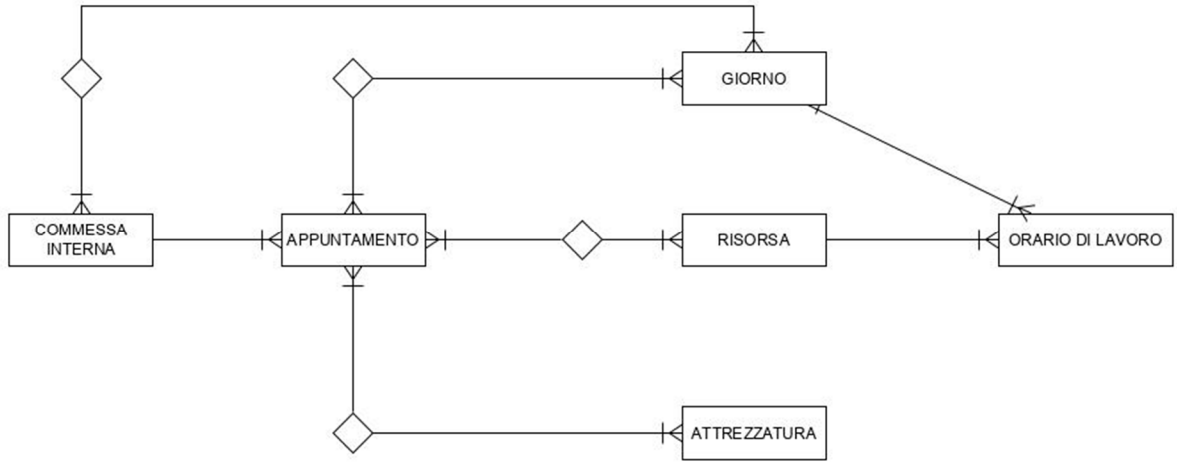

Figura 8: modello ER (2)

Come si può notare nei diagrammi e come è già stato evidenziato nell'analisi dei processi, vi sono collegamenti tra i modelli delle entità.
Essi sono stati studiati e definiti approfonditamente nella fase di analisi in modo da progettare nel modo corretto il database, creando tutte le tabelle, così da avere una solida base per la realizzazione degli altri aspetti dell'applicativo.

Le associazioni rendono le operazioni comuni più facili e veloci.
Il database è stato popolato tramite i file *Migrations*: sono un modo conveniente per modificare il database in modo strutturato e organizzato. I *Migrations* consentono di descrivere le trasformazioni usando Ruby, in modo da essere indipendente dal database.
Ogni *migration* può essere considerata come una nuova "versione" del database: infatti modifica lo schema aggiungendo o rimuovendo tabelle, colonne o voci. La libreria *Active Record* sa come aggiornare lo schema nell'arco temporale, potendo tornare allo stato precedente.
All'interno delle tabelle possono essere aggiunti vincoli di chiave esterna per garantire l'integrità referenziale. L'unico vincolo da ricordare per inserire chiavi esterne è: la tabella alla quale la chiave fa riferimento deve essere già stata creata.

Rails supporta sei tipi di associazioni:

- *belongs_to*
- *has_one*
- *has_many*
- *has_many : through*
- *has_one : through*
- *has_and_belongs_to_many*

Usando queste associazioni si istruisce Rails a mantenere l'informazione ChiavePrimaria-ChiaveEsterna tra due istanze di due modelli, nonché ottenere metodi aggiuntivi che potrebbero tornare utili.

Nell'applicativo sono state usate tre di queste associazioni: *belongs_to*, *has_many* e *has_many : through.*

L'associazione *belongs_to* fissa una relazione uno-a-uno con un altro modello, tale che ciascuna istanza del modello dichiarante "appartiene a" un'istanza dell'altro modello.
All'interno dell'applicativo, relazioni di questo tipo sono presenti tra:

- sopralluogo e commessa cliente: il sopralluogo può essere creato dalla commessa cliente;
- preventivo e sopralluogo: il preventivo può essere creato dal sopralluogo;
- commessa cliente e preventivo: la commessa cliente può essere creata soltanto quando è stato creato il preventivo;
- commessa interna e commessa cliente: come per la commessa cliente, lo stesso tipo di relazione la troviamo tra la commessa interna e la commessa cliente;
- appuntamento e commessa interna: l'appuntamento può avere il riferimento alla commessa interna da cui è stato creato.

Un'associazione *has_many* indica una connessione uno-a-molti con un altro modello.

Questo tipo di associazione la si ritrova dall'altra parte di un'associazione *belongs_to*. Questa associazione indica che ogni istanza del modello ha zero o più istanze di un altro modello.
Per cui, troviamo questo tipo di relazione negli stessi legami citati per l'associazione *belongs_to* ma invertiti, ossia:

- una commessa cliente può avere più sopralluoghi;
- un sopralluogo può avere più preventivi;
- una commessa cliente può avere più commesse interne;
- una commessa interna può avere più appuntamenti.

Il preventivo è un caso particolare rispetto alle due associazioni appena citate: all'interno del proprio modello ha un riferimento a sé stesso, perché se gli viene assegnato come stato "non accettato" da parte di un preventivo, può essere creata la revisione tenendo il riferimento del preventivo "padre".

Has_many : through è l'associazione che viene spesso utilizzata per impostare una connessione molti-a-molti con un altro modello. Questa associazione indica che il modello dichiarante può essere associato a zero o più istanze di un altro modello procedendo attraverso un terzo modello. Il terzo modello altro non fa che memorizzare le chiavi esterne degli altri due modelli che sono legati tra loro.
Associazione di questo tipo, all'interno dell'applicativo, viene usata tra:

- categoria e attrezzatura;
- categoria e risorsa;
- appuntamento e risorsa;
- appuntamento e attrezzatura.

Siccome queste relazioni sono di tipo molti-a-molti, per esempio una risorsa può avere più categorie di lavoro che svolge, il terzo modello archivierà tante tuple nel database quante categorie sono state assegnate alla risorsa. La stessa operazione avverrà per le altre relazioni.

5. L’APPLICAZIONE WEB

L’applicazione web è stata progettata per essere intuitiva e di facile utilizzo. Per accedervi è necessario autenticarsi con username e password.
Gli utenti possono essere creati da altri utenti che possiedono un ruolo specifico e possono assegnare a ciascun utente uno o più ruoli. I ruoli, all’interno dell’applicazione, vengono usati per classificare quali operazioni l’utente è autorizzato o meno ad eseguire e a quali informazioni ha accesso, quindi ogni utente potrà accedere solo alle informazioni che gli servono per svolgere il proprio lavoro e potrà creare/modificare solo ciò che gli è consentito.
L’applicazione è navigabile tramite il menù laterale, cliccando sulle varie sezioni si possono consultare le informazioni già presenti, modificarle o aggiungerne di nuove.
All’interno dell’applicazione sono contenute tutte le anagrafiche necessarie allo svolgimento del lavoro, si trovano sia i dati relativi all’azienda (risorse, attrezzature, orari, etc.) che i dati relativi ai clienti.
In essa è possibile creare tutti i documenti che sono stati citati nei processi analizzati, ossia tutte le entità presenti nel diagramma ER.
I documenti possono essere esportati così da poter condividere con il cliente gli accordi presi.
L’operatore può eseguire il *check-in* e il *check-out* di un lavoro assegnatogli, cosicché il supervisore può controllare in tempo reale lo svolgimento dei lavori.
Inoltre, possono essere fissati e creati gli appuntamenti con i clienti.
Gli appuntamenti possono essere di più tipologie (ripetitivi, spot, etc.).
Tramite l’algoritmo realizzato si ha, in tempo reale, una panoramica dell’occupazione delle risorse dell’azienda e la loro compatibilità con il lavoro, come un’istantanea della situazione attuale. Tramite questa soluzione è più efficiente organizzare e assegnare il lavoro.
Gli appuntamenti possono essere modificati, spostati e cancellati in qualsiasi momento, così da favorire la riorganizzazione del lavoro in base ai cambiamenti che si verificano.
Il calendario è l’elemento centrale dell’applicativo, in esso sono presenti gli appuntamenti.
Il calendario è consultabile da tutti gli utenti, customizzato in base all’utente attualmente connesso ed è filtrabile per poter cercare più rapidamente e chiaramente le informazioni desiderate.

5.1 Overview

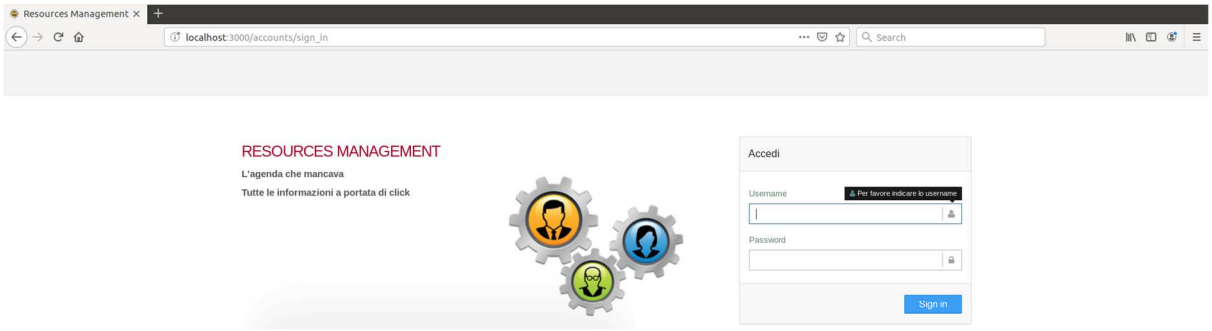

Figura 9: pagina di login

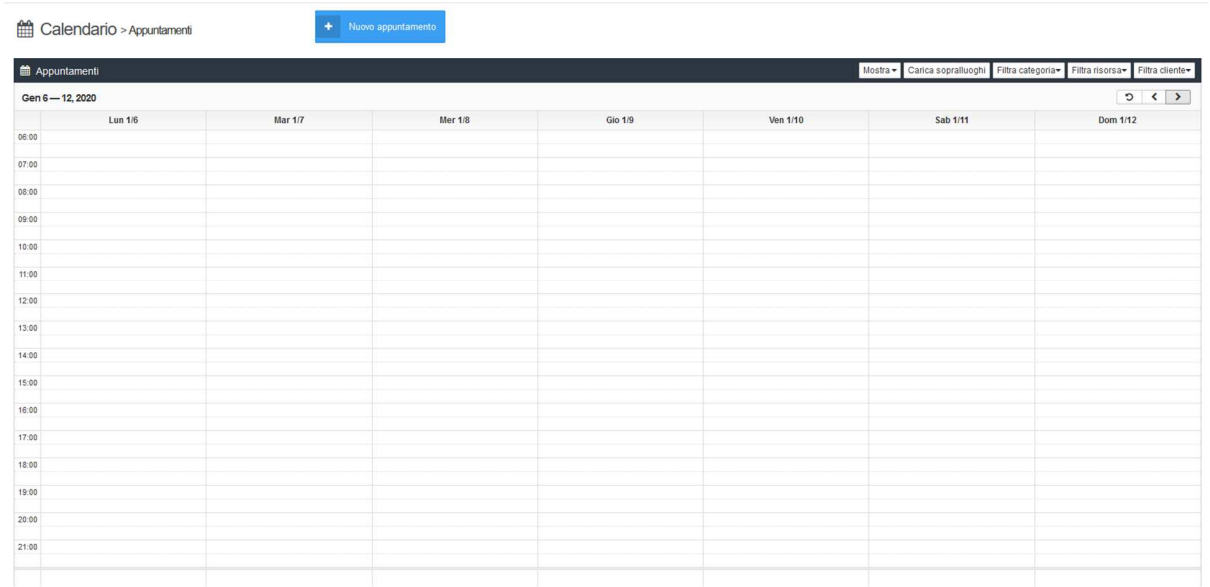

Figura 10: Home page

Nella home page è presente il calendario: è stato inserito in questo posto in modo da richiamare il focus dell'utente non appena effettua l'accesso, un reminder degli appuntamenti della settimana.
Il calendario rende più facile l'organizzazione del lavoro e fornisce un quadro completo della situazione attuale.
La visualizzazione del calendario può essere: giornaliera, settimanale o mensile.
Il calendario può essere filtrato e per farlo vi sono tre filtri: "per categoria", "per risorsa" e

“per cliente”, così è più semplice la visualizzazione quando sono presenti molti appuntamenti. Inoltre, è stato inserito un tasto “Carica sopralluoghi”: come può essere intuito dal nome, cliccando sul tasto, al posto degli appuntamenti, vengono caricati i sopralluoghi, così da poter avere una visione completa anche di essi.
La particolarità che presenta questo calendario risiede nel fatto che gli appuntamenti vengono caricati di periodo in periodo, cioè vengono caricati gli appuntamenti dei giorni che sono visualizzati in quel momento. Quindi, se viene cambiato periodo, gli appuntamenti vengono caricati a runtime.

Cliccando sull’icona con il proprio username, in alto nel menù, l’utente può modificare i propri dati: potrà modificare la password, la e-mail, il proprio nome e il cognome in qualsiasi momento.

Tutte le voci del menù sono state sviluppate seguendo la stessa linea guida: cliccando sopra di esse, come prima pagina, viene caricato l’indice contenente tutti i dati che sono stati già inseriti. Dall’indice di ogni dato (in base alle dipendenze dei dati) è possibile inserirne di nuovi tramite il tasto apposito. Per ogni dato già presente vi sono altri tasti per consultarlo o modificarlo.

Figura 11: Menù (1)

In questa figura si visualizzano le voci corrispondenti alle informazioni anagrafiche dell’azienda.
Questi dati sono consultabili e modificabili solo dagli utenti che hanno il ruolo per farlo, ovvero coloro che gestiscono e organizzano il lavoro.

Sono informazioni relative all'organico dell'azienda, alle attrezzature in possesso ad essa ed altre informazioni utili per organizzare e classificare il lavoro.

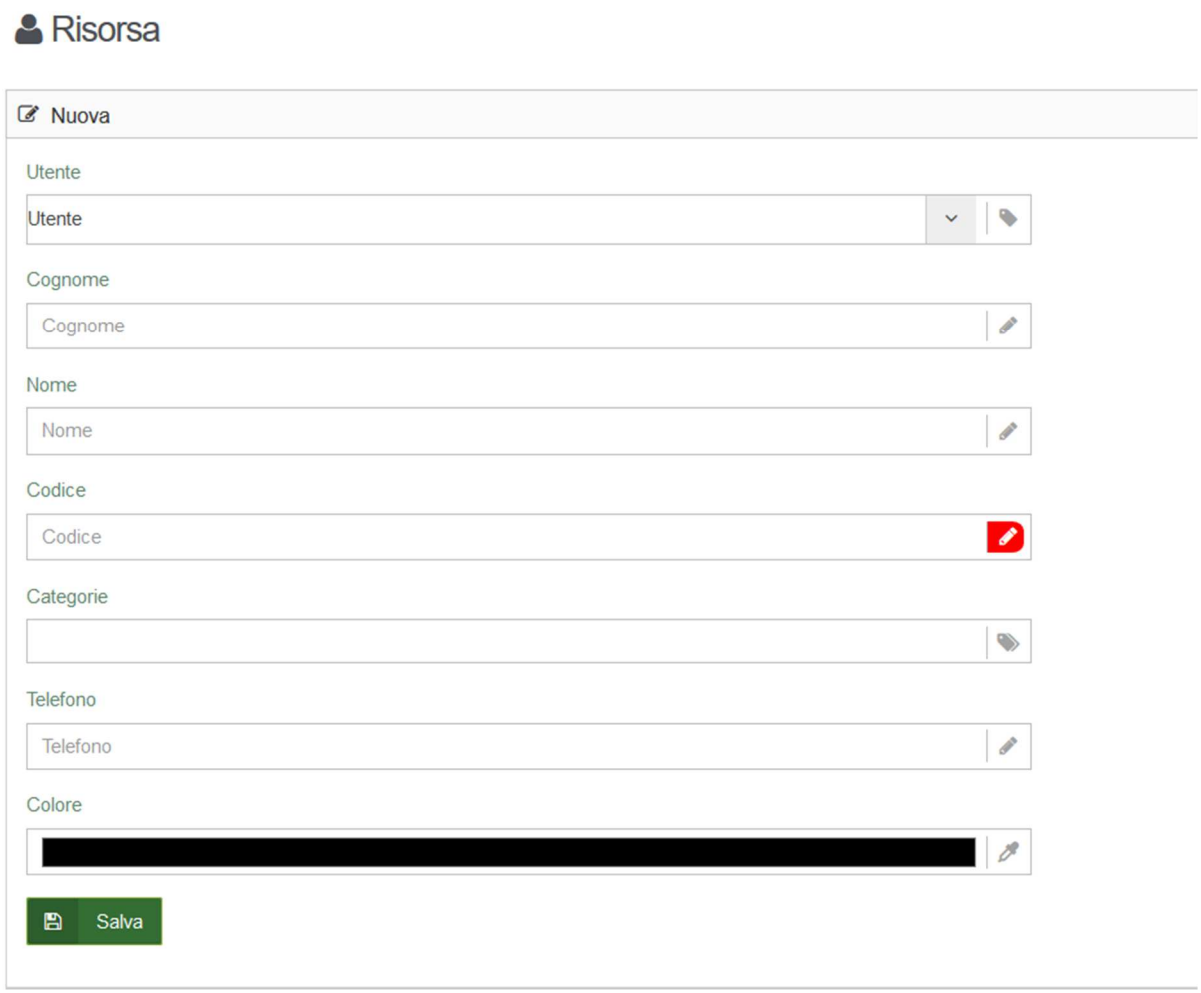

Figura 12: form risorsa

In foto si può vedere il form di creazione di una nuova risorsa nell'applicativo: oltre ai campi puramente anagrafici, è presente il campo *Categorie*, dove si possono selezionare una o più categorie di lavoro da associargli (ossia i lavori che la risorsa è in grado di svolgere).
L'assegnazione di/delle categoria/e non è obbligatoria ma è necessaria al fine della corretta esecuzione dell'algoritmo.
Nello specifico, l'informazione relativa alle *categorie* serve per indicare se quella determinata risorsa è adeguata ad eseguire il lavoro che si sta per assegnare.
È possibile assegnare un colore preciso alla risorsa, sarà usato sul calendario per distinguere gli appuntamenti di quella risorsa rispetto a quelli delle altre risorse (ovviamente è modificabile ogni qual volta è necessario).
Oltre a tutto ciò, è possibile ma non obbligatorio associare questa determinata risorsa ad un utente che è stato creato, così da tenere traccia del legame.

Figura 13: form attrezzatura

Come per la risorsa, anche per l'attrezzatura è possibile indicare una o più categorie per quale viene usata.

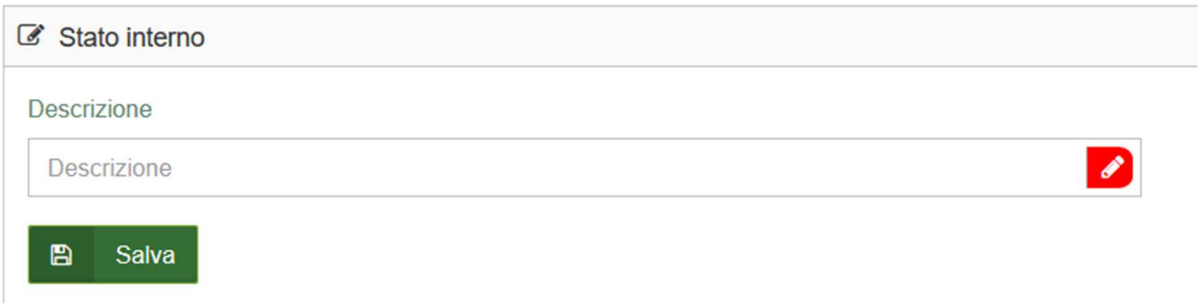

Figura 14: form stato interno

Lo *Stato interno* è un'informazione che viene usata per classificare la commessa interna all'interno dell'azienda.

Figura 15: form tipo di contratto

La *Tipologia di contratto* è un'informazione che riguarda la commessa interna.

Figura 16: form categoria

Questo form riguarda le *categorie* di lavoro che vengono svolte all'interno dell'azienda, sono le stesse voci che possono essere associate alle risorse e alle attrezzature, nonché al lavoro stesso.

Figura 17: form tipologia di sospensione

La *tipologia di sospensione* serve per classificare l'orario di lavoro, ossia se l'orario si sta inserendo è "lavorativo" o di altra natura (le nature che possono essere scelte vengono inserite qua).

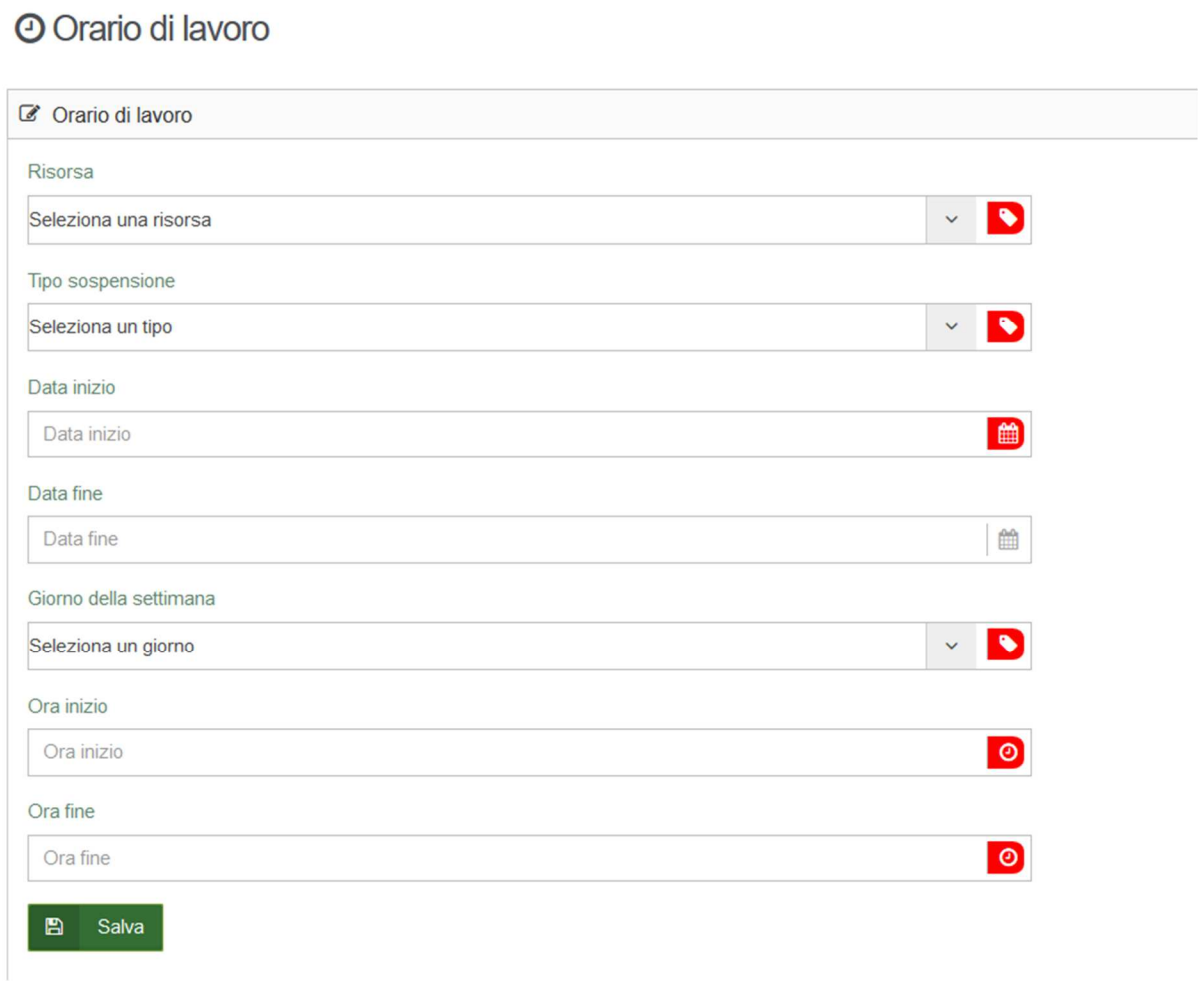

Figura 18: form orario di lavoro

In questa sezione si inserisce l'*orario di lavoro* delle risorse.
L'orario di lavoro è così composto: si inserisce per ogni giorno della settimana uno o più orari (in base alle ore stipulate da contratto). La *tipologia di sospensione* è un riferimento esterno all'entità appena sopra citata.
Queste informazioni risultano necessarie al fine dell'esecuzione dell'algoritmo: servono per indicare se una risorsa, in un dato periodo di tempo prestabilito, è disponibile (lavorativamente), ossia può essere associata al lavoro perché rientra nelle ore stabilite da contratto, oppure se non è disponibile e quindi non presente/occupata per svolgere il lavoro.

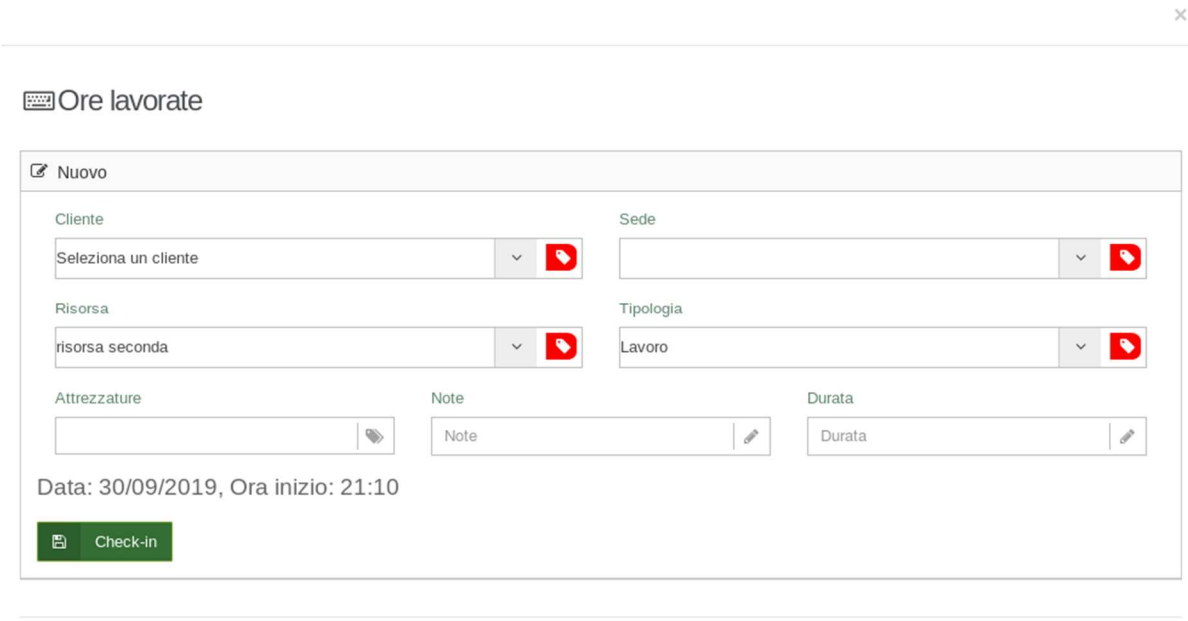

Figura 19: form ore lavorate

Questa parte riguarda la richiesta di poter "monitorare" la risorsa quando esegue un lavoro. È nata in sostituzione all'uso del badge che di solito avviene in un'azienda, ossia emula *l'entrata* e *l'uscita* della risorsa da un lavoro.
Per poter ovviare al fatto che la risorsa potrebbe inserire un orario diverso rispetto a quello in cui sta effettuando l'inserimento, la data e l'ora vengono presi in automatico al momento del click sul tasto *Nuovo* (questa opzione avviene solo in base ad uno specifico ruolo dell'utente) e non possono essere modificati.
Il supervisore può inserire le ore di tutte le risorse andando a modificare sia data che ora.
La risorsa può anche effettuare il *check-in* dall'elenco dei suoi appuntamenti, in questo modo, i dati relativi al *cliente* e alla *sede* saranno compilati in automatico e l'operazione risulterà più veloce.

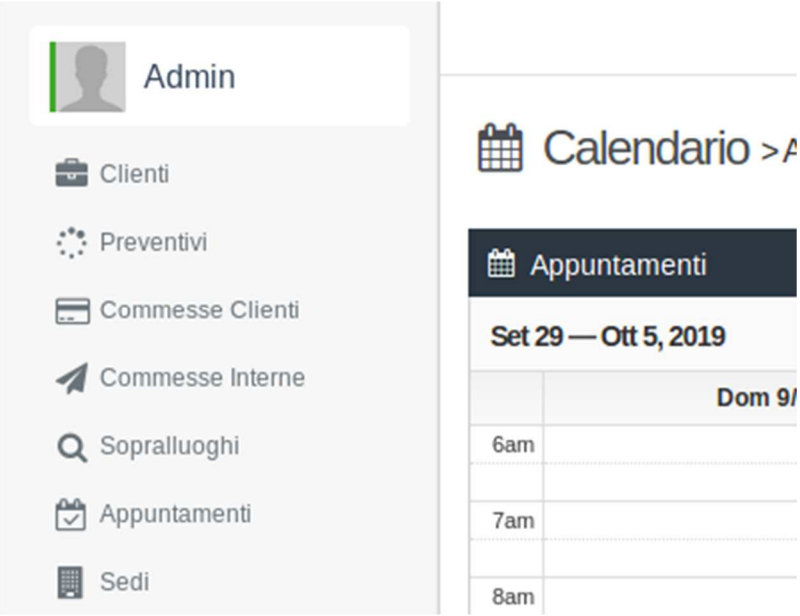

Figura 20: menù (2)

Nella sezione del menù, raffigurata in foto, si trovano le informazioni che vengono modificate e inserite quotidianamente per svolgere il lavoro.
Inoltre, esse sono le informazioni con cui si interagisce di più all'interno dell'applicativo.

Figura 21: form cliente

All'inserimento di un nuovo *cliente*, i dati da inserire sono pochi ma sufficienti per identificarlo.

Se almeno il campo *indirizzo* è compilato, al primo salvataggio viene creata automaticamente l'informazione relativa alla prima *sede* (questo per avvantaggiare l'utente nel caso in cui il cliente possegga una sola sede).
A differenza dei form visti fino ad ora, visualizzando un *cliente* si possono anche visualizzare i dati ad esso collegati, ossia: le *sedi*, i *preventivi* e i *sopralluoghi*. Da questi indici, inoltre, è possibile creare nuovi dati di questi tipi.
Questa opzione è stata pensata per rendere fluido e intuitivo il processo di creazione e consultazione delle informazioni.

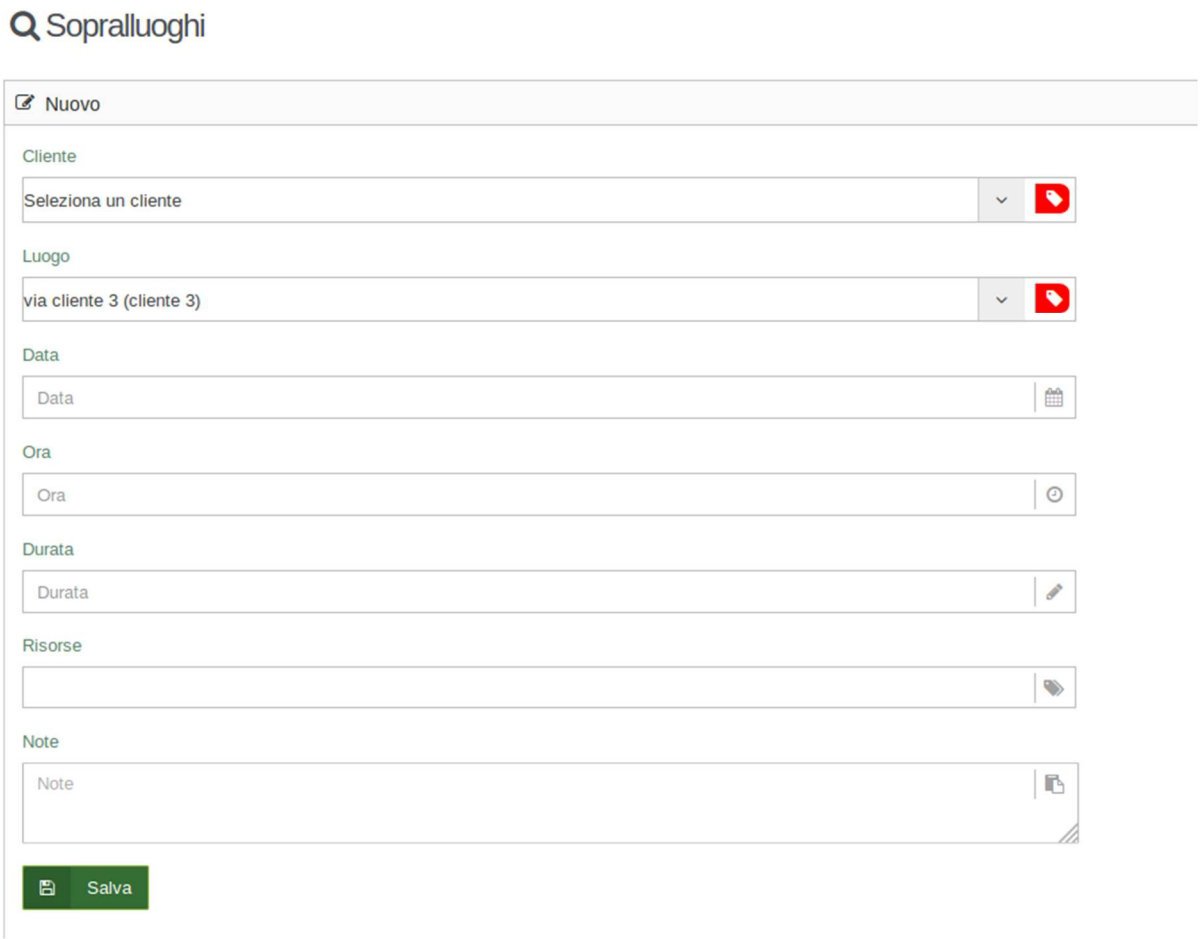

Figura 22: form sopralluogo

Il documento relativo ad un *sopralluogo* può essere creato in più punti: come appena visto, può essere creato all'interno di un *cliente*, dall'indice generale con tutti i sopralluoghi oppure a fronte di un *preventivo* o *commessa cliente*, quindi dopo aver già definito le condizioni del lavoro insieme al cliente.
Se il *sopralluogo* non viene creato a fronte di *preventivo*, una volta salvato è possibile da esso generare il preventivo associato.

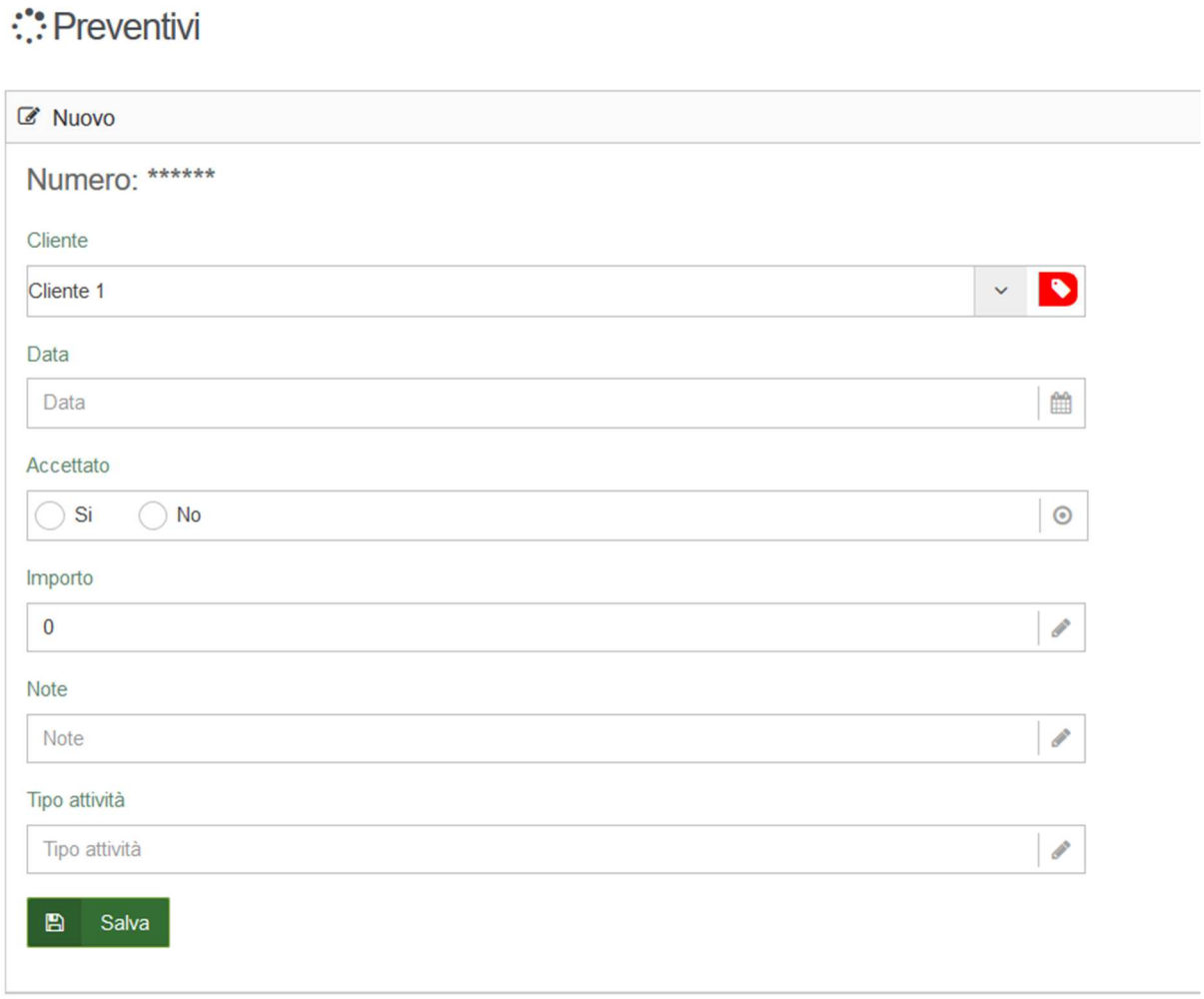

Figura 23: form preventivo

Il documento relativo al *preventivo* fa parte di uno dei processi più importanti ed è stato analizzato.
Finché non gli è stato attribuito uno stato (ossia non è stato né *accettato* né *non accettato*), il documento può essere modificato tutte le volte che è necessario.
Associate al *preventivo* vi sono le *righe*: sono righe vere e proprie, di descrizione dettagliata di un lavoro, che vengono generate automaticamente ogni qualvolta se ne salva una e non vi è un numero massimo.
Le righe possono essere eliminate tramite l'apposito tasto che si trova di fianco. Inoltre, ad ogni salvataggio di una di esse si aggiorna l'importo totale del preventivo.
Al momento del cambio di stato del documento compaiono due azioni:

- se il preventivo non viene accettato da parte del cliente è possibile generarne una nuova revisione; la revisione si porta dietro i dati presenti nel documento padre con la possibilità di essere modificati (la numerazione delle revisioni avviene così: x.1, x.2, etc. con la *x* che rappresenta il numero del preventivo iniziale);
- se il preventivo viene accettato si può procedere e quindi creare la commessa cliente associata.

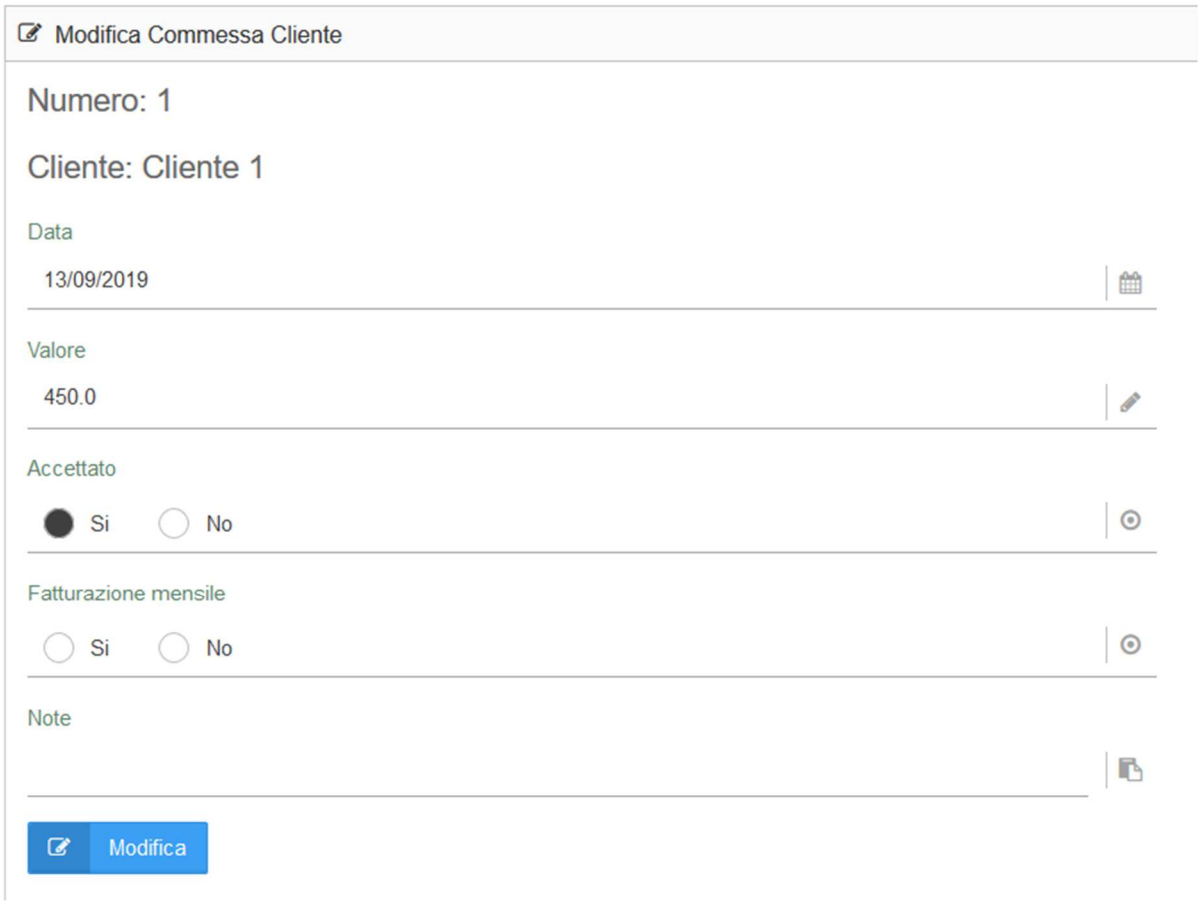

Figura 24: form commessa cliente

Il ruolo della *commessa cliente* consiste nell'inglobare tutte le *commesse interne* relative a quel contratto. All'interno della visualizzazione possono essere consultate le *commesse interne* associate (nonché crearle), come anche i *sopralluoghi* (come già detto).
Nella *commessa cliente* può essere inserita l'informazione relativa alla fatturazione, così da organizzare anche il lavoro amministrativo dell'azienda.

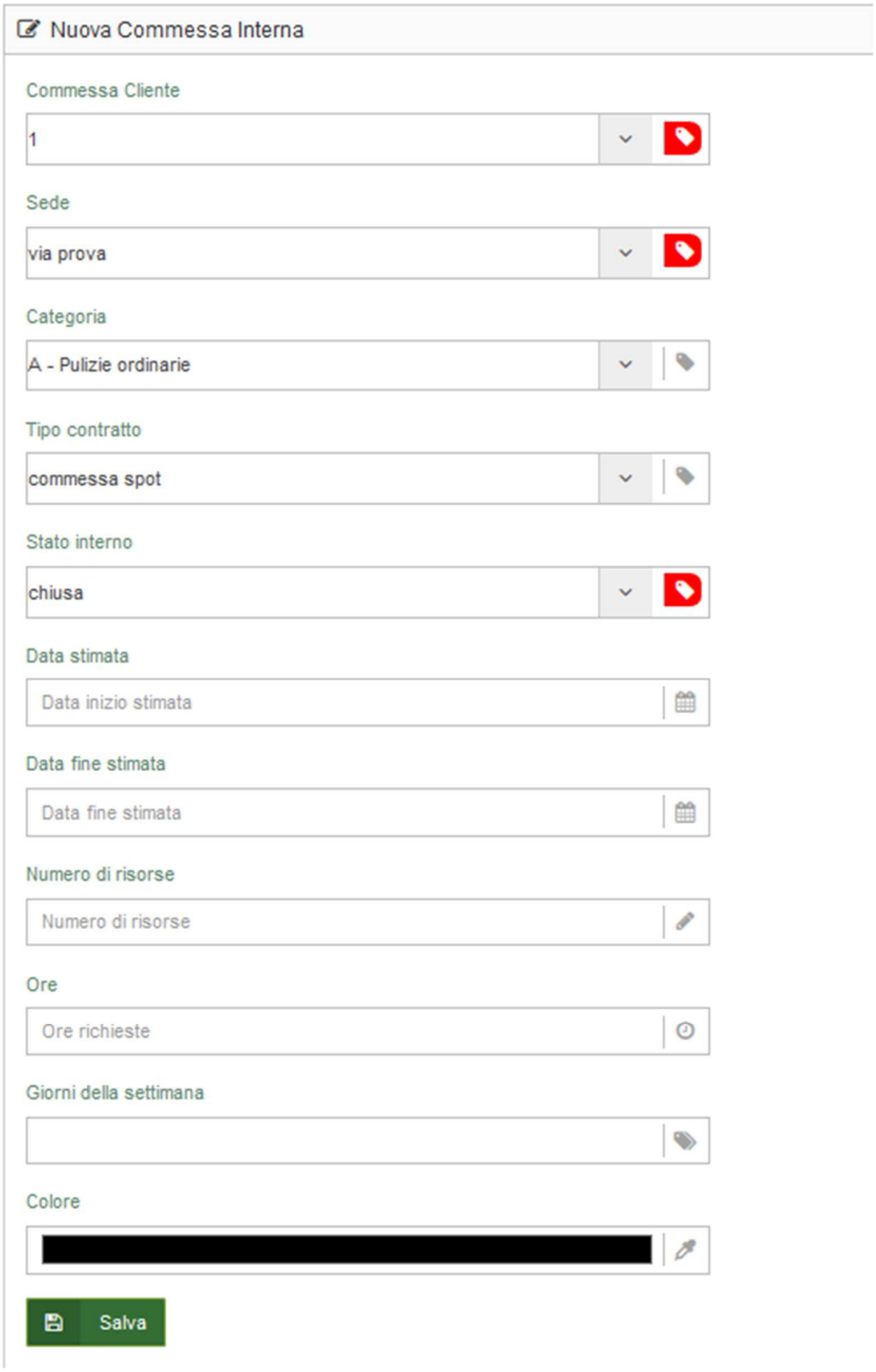

Figura 25: form commessa interna

La *commessa interna* è il documento che rappresenta il lavoro, contiene le informazioni della *tipologia di lavoro* che si svolgerà presso il cliente.
Viene creata una *commessa interna* per ogni informazione inserita nel preventivo, perché una

commessa interna può essere associata ad una sola categoria di lavoro.
La *commessa interna* è il passaggio che precede la creazione degli *appuntamenti*. I dati che vengono inseriti in essa saranno riportati nella creazione degli appuntamenti (ovviamente con la possibilità di essere confermati o modificati).
Al salvataggio del documento, in modalità di visualizzazione, comparirà il tasto per poter creare gli appuntamenti.
Da una commessa interna possono essere creati più appuntamenti e non vi è un limite massimo.
Se sono già stati creati degli appuntamenti da una determinata commessa interna, sarà presente anche un tasto *Rinnovo* per poter proseguire la creazione degli appuntamenti.

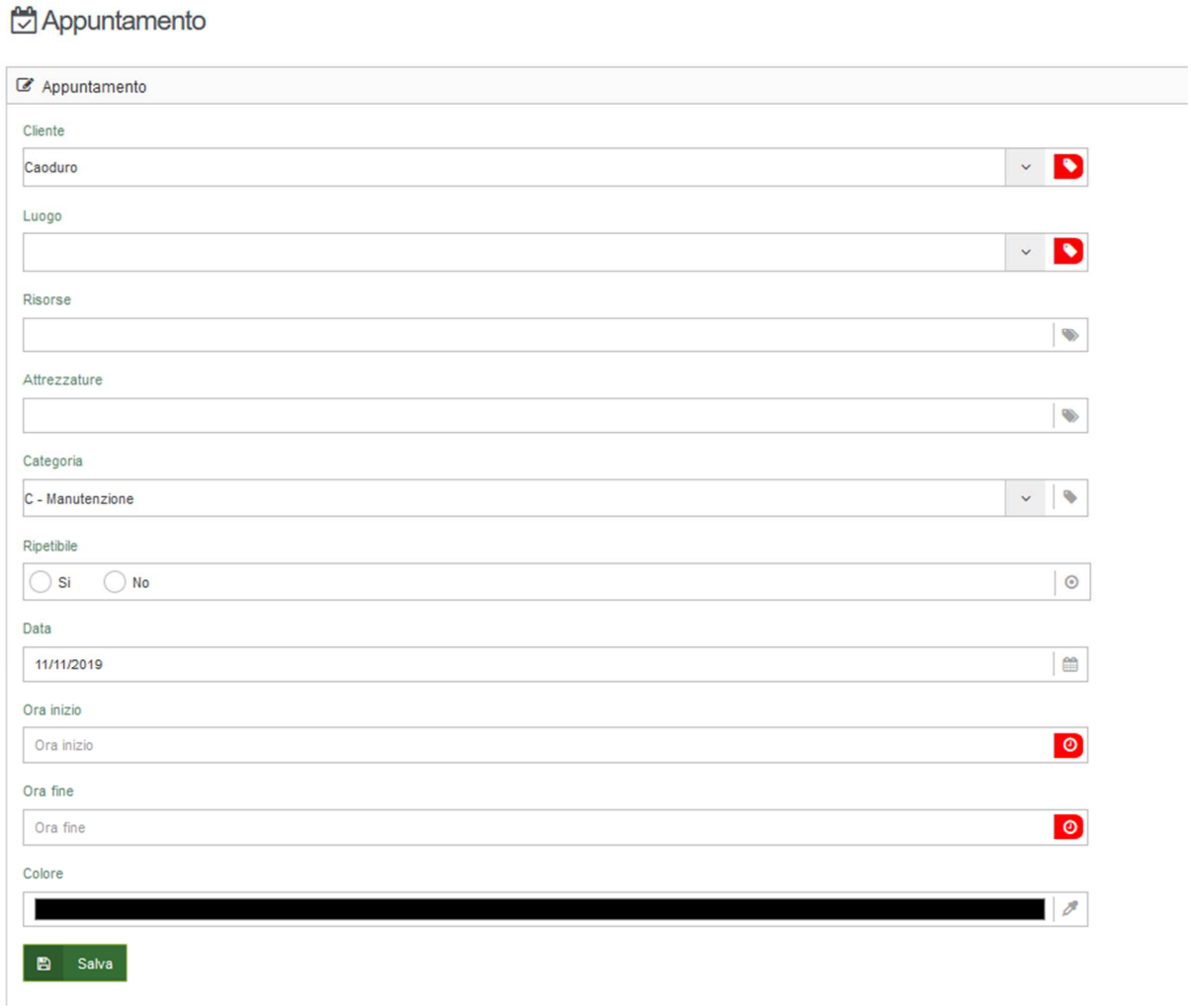

Figura 26: form appuntamento

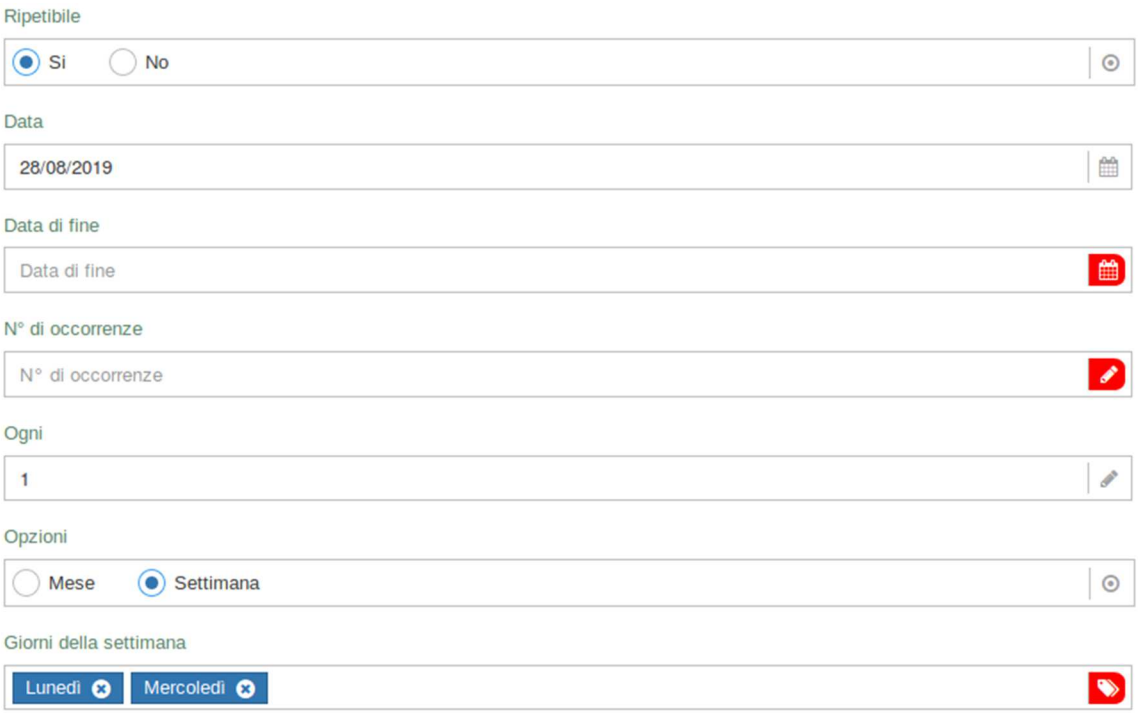

Figura 27: dettaglio ripetibilità

L'appuntamento, cioè l'assegnazione di una o più risorse ad un'attività fissata sul calendario, insieme al calendario stesso, sono la parte centrale dell'applicativo. È possibile crearlo in due modi: tramite la *commessa interna* (come specificato sopra) oppure direttamente dal calendario, dove è presente un tasto apposito.
Se *l'appuntamento* viene creato dalla *commessa interna*, i campi in comune, tra i due form, vengono già compilati per agevolare il lavoro (con la possibilità di essere cambiati).
Il campo della *ripetibilità* serve per indicare se quell'appuntamento è ripetibile oppure no.
Nel caso in cui esso fosse ripetibile, compariranno i campi in più che devono essere compilati.
Questi campi sono: la *data di inizio,* la *data di fine* o in alternativa il *numero di occorrenze,* la *cadenza settimanale* o la *cadenza mensile*.

Se l'*appuntamento* è ripetibile e i campi sono stati compilati correttamente, sul calendario saranno visualizzati tanti appuntamenti quanti sono quelli previsti.
Questi appuntamenti vengono calcolati e creati da una specifica funzione che si trova nel controller degli appuntamenti, questa funzione viene eseguita solo ed ogni volta che viene salvato un appuntamento per la prima volta.
L'algoritmo, che esegue il calcolo dei giorni in cui devono essere fissati gli appuntamenti, è stato pensato in questo modo:

- l'utente inserisce la data di inizio e la data di fine, sceglie i giorni della settimana in cui devono essere fissati gli appuntamenti;
- viene calcolato il primo lunedì che precede la data di inizio e viene presa la prima domenica che segue la data di fine;
- si iniziano a calcolare i giorni della settimana partendo dalla settimana del lunedì

calcolato e si procede così fino ad arrivare alla settimana della domenica calcolata, controllando che le date trovate siano comprese tra la data di inizio e la data di fine selezionate dall'utente;
- per ogni data che viene calcolata viene creato l'appuntamento copiando i restanti dati da quelli inseriti dall'utente;
- in ogni appuntamento viene tenuta traccia dell'appuntamento che lo precede in modo da collegarli uno con l'altro.
- Se, invece di compilare il campo della data di fine, è stato compilato il campo del numero delle occorrenze, l'algoritmo, invece di calcolare la prima domenica che segue la data di fine, conterà gli appuntamenti che vengono creati e si fermerà quando raggiungerà il numero delle occorrenze.

Oltre alla cadenza settimanale della ripetibilità dell'appuntamento, sono state inserite altre due opzioni: la cadenza ogni x settimane e la cadenza ogni x mesi (x rappresenta il numero che può essere inserito nel campo apposito).
Nel caso di scelta della cadenza ogni x settimane l'algoritmo funziona allo stesso modo, come spiegato poc'anzi, con la differenza che non si calcolano i giorni di ogni settimana ma vengono effettuati salti di x settimane.
Nel caso di scelta della cadenza ogni x mesi, l'algoritmo ha un funzionamento leggermente diverso:
- si calcola il primo *giorno della settimana* (che è stato selezionato) del mese;
- si considera la settimana del mese della *data di inizio* scelta;
- si calcola il *giorno della settimana* della settimana calcolata (prima, seconda, terza, etc.) partendo dal giorno calcolato nel primo punto;
- si esegue questa operazione finché non si raggiunge la *data di fine* o il *numero di occorrenze* indicato.

In aggiunta, se è stata scelta più di una risorsa, sul calendario l'appuntamento sarà frazionato: si vedranno tanti appuntamenti quante sono le risorse inserite.

Di default, l'appuntamento, sul calendario, avrà il colore che è stato scelto nel campo omonimo sul form della risorsa. Se, invece, durante la creazione del documento viene scelto un colore differente, quindi diverso dal nero (come in foto), allora l'appuntamento sarà visibile sul calendario con questo colore.

Nella creazione delle tabelle, nei file di migrations, è possibile indicare quali campi hanno l'obbligo di essere compilati. L'indicazione di obbligatorietà, nei form, viene evidenziata con il simbolo della tipologia di campo (testo, numero, etc.) di colore rosso (come si può notare in tutte le foto dei form). Ovviamente se questi campi non vengono compilati il sistema non fa procedere con il salvataggio del documento.
Un altro modo per evidenziare l'obbligatorietà della compilazione di un campo consiste

nell’inserire il controllo nel *model* dell’entità. La variante è usata soprattutto nel caso in cui la compilazione di un campo diventa obbligatorio solo in dipendenza dalle scelte fatte negli altri campi (per esempio: se l’appuntamento è ripetibile, è obbligatorio compilare la data di fine periodo).

5.2 Calcolo occupazione delle risorse

L'algoritmo sviluppato è la parte più importante dell'applicazione, è la parte che differenzia questo prodotto da quelli presenti sul mercato.
Questo algoritmo si occupa di fornire, in tempo reale, l'occupazione delle risorse.

Una delle esigenze/problematiche emerse nel colloquio con le aziende riguarda proprio questo argomento: nel momento in cui viene acquisito un nuovo lavoro ed è necessario programmare gli appuntamenti, è difficile, considerata la mole di lavoro, comprendere nell'immediato quali risorse saranno disponibili a svolgere il lavoro.
Focalizzata questa problematica, la si è analizzata per trovare una soluzione.
La prima fase è consistita nel comprendere quali fattori vengono considerati per decidere quale risorsa è idonea a svolgere un determinato lavoro.
È emerso che sono due i fattori principali che determinano l'assegnazione di un lavoro:

- compatibilità: la risorsa è in grado di svolgere quel particolare lavoro (per varie ragioni, perché serve un brevetto o una patente particolare, etc.);
- disponibilità: la risorsa sarà disponibile (quindi non occupata a svolgere altri lavori ed è nell'orario di lavoro) a svolgere il lavoro per tutta la durata.

Quindi è stato concordato che l'output dell'algoritmo avrebbe suddiviso le risorse in quattro sottogruppi: *compatibile e disponibile*, *compatibile e non disponibile*, *non compatibile e disponibile*, *non compatibile e non disponibile*.
È stato concordato in fase di analisi la possibilità di visualizzare tutte le combinazioni e quindi tutte le risorse; questo per diverse ragioni:

- una determinata risorsa risulta compatibile ma non disponibile perché impegnata in un altro lavoro, l'utente decide, però, che è più idonea a svolgere il nuovo lavoro e di conseguenza spostare o riassegnare quello già presente;
- una determinata risorsa non è compatibile con questo specifico lavoro, ma conosce già il cliente (ha eseguito altri tipi di lavoro presso di lui) per cui l'utente decide di assegnare il lavoro a questa risorsa;
- etc.

È stata una scelta presa per eliminare qualsiasi tipo di vincolo all'utente, in modo da renderlo libero di organizzarsi il lavoro al meglio.
L'utente è reso consapevole della possibilità di conflitti o non disponibilità delle risorse, ma anche conscio di potersi riorganizzare il lavoro in qualsiasi momento.

Nella fase successiva si è verificato quali fattori fossero da controllare per affermare in quale sottogruppo una risorsa fosse da inserire.
Il controllo della *compatibilità* si basa su un unico fattore: la categoria. Più semplicemente, se alla risorsa in questione è stata assegnata la categoria di quel determinato lavoro, la risorsa risulta *compatibile*, altrimenti è *non compatibile*.

Al contrario, il controllo della *disponibilità* ha più fattori da esaminare:

- orario di lavoro;
- appuntamenti.

Il primo fattore, l'*orario di lavoro*, ha più informazioni che devono essere controllate, ossia: la prima in assoluto è relativa all'orario *lavorativo*, quindi se nei giorni della settimana indicati l'orario della risorsa è indicato come *lavorativo*; ma è anche da controllare se in quelle date precise, la risorsa risulta in *permesso*, *ferie*, *malattia*, etc.
Il secondo fattore, gli *appuntamenti*, riguarda il controllo degli appuntamenti già assegnati alla risorsa, quindi se la risorsa risulta già assegnata a qualche lavoro nel periodo e orario indicato.
Se anche solo uno di questi controlli non restituisce esito positivo la risorsa risulta *non disponibile*, in caso contrario risulta *disponibile*.

La terza fase ha riguardato la trasformazione dell'analisi fatta in codice, quindi trovare un modo di eseguire i controlli in modo efficace, non troppo complesso, elegante e ricavare l'output nel minor tempo possibile.
Per cui la soluzione che è stata adottata è la seguente:

1. viene controllato se è un evento ripetibile;
2. se l'evento non è ripetibile, per ogni risorsa:
   -si controlla se la categoria del lavoro è stata assegnata a tale risorsa, quindi se è compatibile o non compatibile;
   -si controlla che l'orario e il giorno indicato siano per la risorsa orario di lavoro;
   -si controlla se non ci sono già appuntamenti in quel giorno per quell'orario;
   -in base ai risultati delle ricerche, verrà inserita nel sottogruppo giusto.
3. se l'evento è ripetibile, vengono salvati i giorni della settimana selezionati e vengono calcolati il lunedì precedente alla data di inizio e la domenica seguente alla data di fine; per ogni risorsa:
   -si controlla se la categoria del lavoro è stata assegnata a tale risorsa, quindi se è compatibile o non compatibile;
   -per ogni settimana presente nel periodo selezionato e per ogni giorno della settimana selezionato si controlla se per la risorsa è orario di lavoro e non ha appuntamenti già fissati. Alla prima corrispondenza, ossia se non è orario di lavoro o ha appuntamenti, si interrompe il controllo per quella risorsa e sarà inserita in uno dei due sottogruppi in cui compare *non disponibile*.

Per agevolare il lavoro ed offrire all'utente una maniera semplice per ottenere il risultato, l'elenco completo, contenente i quattro sottogruppi, può essere ottenuto ed aggiornato al click di un bottone. È stato scelto di agire in questo modo per via dei tanti fattori che vengono considerati nello svolgimento dell'algoritmo.
Per cui l'utente, previo corretto inserimento dei dati, dovrà cliccare il tasto di fianco al campo delle risorse per popolare correttamente, con i dati aggiornati, il campo.

6. TEST E RILASCIO DELL'APPLICAZIONE

Ancora prima di iniziare la fase di sviluppo, con i clienti è stato affrontato il tema del rilascio dell'applicazione, più nello specifico quale sarebbe stata la soluzione hardware migliore per essi.
All'interno del team, alcune persone si sono occupate di elaborare le varie soluzioni possibili relative a questo tema, con pro e contro e costi/benefici, da poter presentare ai clienti.
La principale differenza tra le soluzioni proposte risiede nell' utilizzo di un server proprietario all'interno dell'azienda (se ne possiede uno) oppure affidarsi ad un servizio erogato da terzi che mettere al servizio i propri server.
Nel secondo caso, le soluzioni proposte si sono basate sui servizi offerti da Amazon Web Services (AWS) e sui servizi offerti da aruba.it.
Un cliente ha scelto di installare l'applicazione su un proprio server all'interno dell'azienda, in quanto in possesso di esso senza costi per un servizio aggiuntivo.
Gli altri clienti hanno scelto di affidarsi al servizio offerto da aruba.it siccome già tutt'ora usano servizi offerti da esso. La scelta è ricaduta su un Cloud VPS, un server virtuale con caratteristiche specifiche, questa opzione prevede la possibilità di espandere le caratteristiche scelte qualora la situazione ne richieda la necessità.
In entrambi i casi, il sistema operativo adottato è Linux dato che è totalmente portabile su un altro server qualora fosse necessario.

I test effettuati da parte nostra sono stati eseguiti ogni qualvolta è stata applicata una modifica in modo da verificare che tutto funzionasse correttamente e non ci fossero errori.
I test sull'applicazione, invece, da parte dei clienti, sono iniziati dopo il primo rilascio.
Nel primo rilascio è stata sviluppata tutta la parte dei documenti, quindi creazione, visualizzazione, modifica ed eliminazione, con le relative relazioni e la possibilità di inserire in modo spot (non ripetitivi) gli appuntamenti.
È stata presa questa decisione per rendere i clienti totalmente consapevoli e aggiornati sull'andamento dello sviluppo e attivi nel testare l'applicativo per riscontrare eventuali errori o anomalie.
Dopo questa prima fase, con i clienti sono stati concordati i successivi rilasci e quali aggiornamenti avrebbero dovuto contenere compatibilmente con quello già presente e in ordine di importanza.
Nel dettaglio, i successivi rilasci sono stati:

- possibilità di creare appuntamenti ripetitivi;
- possibilità da parte dell'operatore di registrare le proprie ore lavorate presso un cliente e in più l'estensione del preventivo con la dinamicità delle righe associate;
- ampliamento della ripetibilità degli appuntamenti, nello specifico la possibilità di creare un appuntamento ripetitivo ogni x mesi/settimane;
- calcolo dell'occupazione delle risorse a runtime sulla creazione degli appuntamenti.

In ogni rilascio sono state inserite modifiche relative a problemi segnalati o suggerimenti da parte dei clienti. Prima di procedere con le modifiche, sono state discusse e valutate dal team per comprendere meglio se fossero necessarie e, soprattutto, quale soluzione sarebbe stata la migliore.

Alla data di produzione di questo documento, siamo in attesa di ricevere i feedback relativi sull'intera applicazione, ossia con la presenza di tutti gli elementi richiesti.
Siamo interessati a comprendere se il lavoro svolto ha migliorato l'organizzazione del lavoro e se i dati sono reperibili in modo rapido e semplice così da poter migliorare sempre di più l'applicativo.

7. CONCLUSIONI

Nel momento in cui avremo ricevuto i feedback relativi all'intera applicazione, verranno decisi i prossimi step per poter rendere l'applicazione un prodotto e metterlo sul mercato a disposizione di possibili nuovi clienti.

Poter far parte, fin dall'inizio, del team che si è occupato dell'analisi e dello sviluppo dell'applicativo, mi ha dato modo di osservare e apprendere quali sono le fasi che compongono il raggiungimento della creazione di un nuovo prodotto.
Ho preso più confidenza nell'uso dello strumento e sulle mie capacità, ho avuto modo di confrontarmi e partecipare alle decisioni che sono state prese e che hanno portato alla creazione dell'applicativo. Inoltre, ho acquisito nuove conoscenze, sia in termini di realizzazione di un prodotto che in termini di linguaggio di programmazione.
Ho avuto la possibilità di relazionarmi con i clienti, di ascoltare le proprie perplessità/dubbi e domande, di ricevere e commentare con essi le loro necessità e le loro osservazioni.
Come molto spesso capita, il cliente non sempre è in grado di comprendere il lavoro che viene fatto per raggiungere un determinato risultato e, quindi, il perché di determinate scelte piuttosto che altre. Il cliente ha padronanza nel suo campo, per cui ho compreso meglio come devono essere gestite le relazioni con essi.

7. BIBLIOGRAFIA

http://www.san.it/
http://www.railstutorial.it/book/da_zero_a_deploy
https://www.html.it/guide/guida-ruby-on-rails/
https://www.ionos.it/digitalguide/server/know-how/postgresql/
http://www.claudiodesio.com/ooa&d/mvc.htm
www.ifma.it
https://it.wikipedia.org/wiki/
https://stackoverflow.com/
https://api.rubyonrails.org/
https://gorails.com/forum
https://wrapbootstrap.com/theme/smartadmin-responsive-webapp-WB0573SK0