Università degli Studi di Modena e Reggio Emilia

DIPARTIMENTO DI SCIENZE FISICHE, INFORMATICHE E MATEMATICHE

Corso di Laurea in Informatica

Analisi e implementazione di software per Customer Relationship Management

Candidato:
Simone Bisi

Relatore:
Ing. Riccardo Martoglia

Anno Accademico 2017–2018

Ringraziamenti

Ringrazio l'Ing. Martoglia per avermi concesso la possibilità di svolgere questa tesi sotto la sua supervisione come relatore, per la sua disponibilità ed i consigli forniti per la sua redazione.

Grazie a Stefano Zaniboni e tutta Rinnai Italia per avermi ospitato in questi mesi ed avermi fatto sentire parte attiva dell'azienda, con un ringraziamento particolare a Daniele Ruffini, per la compagnia e l'aiuto offerto in qualità di IT Manager.

Un enorme ringraziamento va ai miei genitori ed a tutta la mia famiglia: non credo di poter trovare parole adeguate per rendervi grazie abbastanza, ma, per quanto possibile, dedico questo traguardo a tutti voi che mi avete sempre supportato in questo percorso di studi e che, tra alti e bassi, non avete mai smesso di credere in me; un ringraziamento speciale va a mio fratello Andrea, certo che riuscirà prendere le decisioni giuste in questi ultimi anni dell'adolescenza.

Amici, mi sarebbe impossibile ringraziavi uno ad uno, perciò vi ringrazio tutti; grazie a tutti coloro che hanno allietato le mie serate, a quelli con cui ho condiviso una pinta e una storia in questi anni di Università, a chi mi ha accompagnato nelle lunghe giornate di studio, a quelli che vivono più lontano ma è come se fossero stati sempre qui ed a tutti i Peoni che non hanno mai smesso di fare festa.

Infine, grazie a Carlotta.
Per avermi preso per mano ed aver illuminato il cammino di questo nostro viaggio, giorno dopo giorno, verso un lungo futuro che partendo dalla scale del Nettuno di Bologna deve ancora essere scritto.

Modena, 10 Aprile 2019

Simone Bini

Parole chiave

Customer Relationship Management
Reparto commerciale
Web application
SuiteCRM
PHP

Abstract (English)

Businesses approaching in the contemporary commerce world are in constant competition, trying hard to find new customers and to keep the current ones interested to their brand.

Since 1980s the concept of "database marketing" begun to spread as the first approach to the digitalization of offices in order to reduce the hoarding of paper documents. With the propagation of the "computers on every desk and in every home" idea, the power to store huge amounts of data and analyze them with statistical methods has become reality.

Together with this new capabilities, the first innovative companies started to make their way through the developement of dedicated softwares, aimed on keeping a direct link with existing customers and to generate new "leads", perceived as new business opportunities, up to a real Customer-Relationship Management approach. Today, the adoption of company schemes and guidelines for placing the customer at the center of business decitions is a priority: the estimate for the CRM software market has reached 39.5 billion USD in 2017 and in 2018 it's expected to continue being the leader for the entire IT industry, with a 16% growth rate.

The purpose of this thesis is to study the implementation of a CRM based on open source software at the Italian branch of a renowned Japanese multinational corporation, leader in the industry of gas applicances, showing how this type of solutions could be used in the business world and customized according to requests of the commercial department, increasing the overall productivity and the easiness in performing pre-sales operations.

Abstract (Italiano)

Le aziende che si approcciano al mondo del business contemporaneo si trovano coinvolte in una costante sfida alla ricerca di nuovi possibili acquirenti e per mantenere alto l'interesse della propria clientela corrente.
Dagli anni 80 del XX secolo si è iniziato a diffondere il concetto di "database marketing", primi approcci ad uno schedario digitale che sostituisse il predominante dato cartaceo presente negli uffici. Con la diffusione del "computer su ogni scrivania ed in ogni casa" nel decennio successivo, la capacità di immagazzinare dati in grande dimensione e successivamente utilizzare metodi statistici per analizzarli è divenuta realtà.
Di pari passo con il progresso di queste nuove possibilità, le prime compagnie innovatrici hanno iniziato a farsi strada nello sviluppo di software dedicati a mantenere un collegamento diretto con i clienti già esistenti e nella generazione di "lead", intese come opportunità di vendita, fino ad arrivare ad un vero e proprio approccio di Customer-Relationship Management.
Ad oggi, l'adozione di politiche aziendali volte a porre il cliente al centro è una priorità: si è stimato che il valore complessivo del mercato dei programmi CRM abbia raggiunto 39.5 miliardi di dollari nel 2017 e che nel 2018 continuerà ad essere il leader dell'intero settore dei software, con un tasso di crescita del 16%.
In questa tesi verrà analizzata l'implementazione di un applicativo CRM basato su software open source presso la succursale italiana di una prestigiosa azienda giapponese, leader nel settore degli apparecchi a gas, mostrando come soluzioni di questo tipo possano essere utilizzate in ambiti aziendali e personalizzati in base alle richieste del reparto commerciale, aumentando la produttività complessiva e la facilità con cui vengono svolte le operazioni di pre-vendita.

Indice

Introduzione

Obiettivo del progetto

Il presente elaborato illustra il lavoro svolto dal laureando durante il periodo di tirocinio universitario presso l'azienda *Rinnai Italia S.r.l.*. L'obiettivo generale del progetto consisteva nella selezione e l'implementazione di un applicativo per la gestione dei rapporti con la clientela, detto *Customer Relationship Management* al fine di migliorare la situazione già esistente in azienda. Si è successivamente provveduto allo sviluppo delle richieste del reparto commerciale su base di un noto software open-source, sia tramite gli strumenti di personalizzazione offerti da esso che tramite modifiche al codice sorgente.

Problematiche affrontate

Il lavoro svolto ha richiesto una fase iniziale di analisi della situazione aziendale, nonché degli applicativi CRM disponibili sul mercato. Successivamente tramite colloqui con il reparto commerciale è stato possibile delineare un'insieme di richieste e di realizzarle in base alla loro urgenza. Le soluzioni realizzate sono state implementate tramite il linguaggio per template Smarty, MySQL, PHP e Python, oltre che i tre capisaldi dello sviluppo front-end dei siti web (HTML, CSS, JavaScript).

Struttura della tesi

Il lavoro di tesi è strutturato in due parti principali, composte ciascuna da diversi capitoli.

- **Introduzione**: il presente capitolo introduttivo.
- **Caso di Studio**, comprendente:
  1. **Analisi aziendale** rappresenta un capitolo iniziale dedicato all'analisi della compagnia Rinnai Italia e della casa madre Rinnai Corporation al fine di delinearne il profilo aziendale
  2. **Customer Relationship Management** studio della letteratura disponibile sulle pratiche di gestione delle relazioni con il cliente, sulla loro storia e degli applicativi disponibili
- **Progetto e Sviluppo**, comprendente:
  3. **Progetto** mostra le fasi necessarie ai fini della progettazione del software, precedenti alla fase vera e propria di implementazione, comprendente un'analisi del software *SuiteCRM* nella sua struttura interna
  4. **Implementazione** rappresenta le fasi di realizzazione dell'applicativo, mostrando i componenti che facilitano la creazione di nuove funzionalità e facendo riferimento alle porzioni di codice personalizzato realizzate per raggiungere i risultati desiderati
  5. **Distribuzione** si procede con la conclusione del progetto, descrivendo le tecniche impiegate per distribuire l'applicativo agli utenti finali all'interno dell'ambiente aziendale
- **Conclusione**: note e considerazioni personali sul progetto e sull'attività svolta

Infine, il lettore potrà trovare in appendice parti di codice al fine di illustrare al meglio l'effettiva implementazione delle soluzioni sperimentate.

Definizioni, acronimi e abbreviazioni

Entità	**Definizione**
CRM	Customer Relationship Management
B2B	business-to-business, modello di vendita diretto ad altre aziende
B2C	business-to-consumer, equivalente di vendita al dettaglio
HTML	HyperText Markup Language, linguaggio di markup per la formattazione di pagine web
Javascript	linguaggio di scripting utilizzato nello sviluppo web per fornire contenuti dinamici e/o interattivi
PHP	acronimo ricorsivo di *PHP: Hypertext Preprocessor*, linguaggio di scripting utilizzato lato server per la creazione di pagine web dinamiche
Smarty	template engine utilizzato per separare la logica PHP dall'HTML
Python	linguaggio di scripting tra i più diffusi al mondo, considerato anche vero e proprio linguaggio di programmazione
CSS	Cascading Style Sheets, linguaggio ausiliario all'HTML per fornire la formattazione delle pagine
MySQL	sistema per la gestione di basi di dati relazionali, conforme ad SQL (Structured Query Language)
GitHub	sito web utilizzato dagli sviluppatori per la condivisione di codice di software collaborativo

Parte I

Caso di Studio

Capitolo 1

Analisi aziendale

1.1 L'azienda

Questo capitolo rappresenta una ricerca iniziale effettuata sul background aziendale, al fine di identificarla all'interno del proprio settore e per comprenderne valori ed obiettivi.

1.1.1 Rinnai Corporation

Rinnai Corporation è una società con sede a Nagoya, Giappone, leader nel settore della progettazione, costruzione e commercializzazione all'ingrosso di materiale idrotermosanitario. Fondata come "Rinnai & Co" nel Settembre del 1920 da Hidejiro Naito e Kanekichi Hayashi, il nome dell'azienda è coniato da alcuni caratteri dei cognomi dei due fondatori, essendo "Rin" un'altra chiave di lettura per il carattere "Hayashi", e "Nai" da "Naito" [1].

Rinnai

Fig. 1.1: Il logo di Rinnai Corporation

L'azienda sviluppa da sempre tecnologie per la combustione del gas e a partire dal

1925 si diffonde in Asia (Cina, Taiwan e Corea del Sud). Nel 1937 inizia la produzione commerciale di apparecchi per acqua calda sanitaria, ma già nel 1938 viene riconvertita sotto supervisione dell'esercito per produrre parti di aeromobili. Nel 1946, la sede centrale, gravemente danneggiata dalla guerra, viene ricostruita e l'apparato manifatturiero viene riconvertito al suo stato originario. Nei decenni seguenti, Rinnai apre nuovi stabilimenti produttivi in Giappone ed inizia una collaborazione con la tedesca Schwank GmbH.
Durante gli anni Settanta l'espansione continua in Nord America, Australia, Brasile, Malesia, Nuova Zelanda e Taiwan ed avviene una ristrutturazione dell'ambiente aziendale, diventando a tutti gli effetti "Rinnai Corporation". È datata 1983 la quotazione alla borsa di Nagoya e Tokyo, mentre negli anni successivi avvengono la nascita di Rinnai Singapore Pte., Shangai Rinnai Co. Nel 2007 nasce Rinnai Italia e soli due anni dopo viene aperta un'ulteriore holding in Canada. Ad oggi, Rinnai opera in 29 filiali, di cui 6 stabilimenti produttivi per gli apparecchi e tutta la componentistica interna aventi ottenuto le certificazioni di qualità ISO9001 e ISO14000, impiegando 3.628 persone in Giappone e 8.824 nel resto del mondo [2].

1.1.2 Rinnai Italia SRL

Rinnai Italia nasce nel 2007 in seguito all'interessamento della casa madre nei confronti dell'azienda Aqua, società commerciale che rivendeva con ottimo riscontro gli scaldacqua Infinity in tutta Europa. La società era stata fondata dall'ing. Stefano Zaniboni, attuale Executive President della azienda italiana.
Dal sito web dell'azienda si ha il seguente breve estratto, che sintetizza la *mission* dell'azienda:

> *" [3] Rinnai Italia veicola sul mercato numerosi prodotti, dagli scaldacqua alle caldaie a condensazione, dai ventilconvettori alle asciugabiancheria. La distribuzione avviene attraverso una rete esclusiva di rivenditori e grossisti, presenti su tutto il territorio nazionale. Affiancamento tecnico, commerciale e di progettazione, assicurano il massimo servizio a installatori, studi di progettazione, rivenditori e utenti finali. Il core business di Rinnai Italia è rappresentato dai produttori di scaldacqua ai quali si*

affiancano numerose altre tipologie di prodotti: caldaie a condensazione, ventilconvettori e asciugabiancheria. "

1.2 Realtà operativa

1.2.1 Rivenditori

La distribuzione dei prodotti a marchio Rinnai in Italia avviene attraverso una rete esclusiva di rivenditori presenti su tutto il territorio. È su questi che si concentrano gli sforzi degli agenti del reparto commerciale.

1.2.2 Installatori

Gli installatori sono professionisti che effettuano il montaggio e l'allacciamento alla rete degli apparecchi. Dovranno inoltre fornire all'utente finale le istruzioni di funzionamento dell'apparecchio e consegnargli il libretto di istruzioni dello stesso. Essi potranno in seguito effettuare manutenzione e collaudi sugli apparecchi installati da loro stessi. Un'ulteriore sotto-categoria è composta dagli *Installatori con Assistenza* (I.C.A.), i quali saranno autorizzati anche ad effettuare operazioni di manutenzione su tutte le macchine Rinnai.
Gli installatori possono inoltre effettuare una richiesta per essere inclusi nell'elenco degli I.QU. (*Installatore Qualificato*) tramite l'invio di un form presente sul sito web di Rinnai Italia, ottenendo così un trattamento esclusivo e la possibilità di seguire i corsi di formazione dedicati.

1.2.3 CAT

I CAT sono i centri di assistenza tecnica, autorizzati e formati da Rinnai Italia tramite corsi professionali tenuti presso la sede centrale di Carpi, i quali vanno a comporre una vasta rete di tecnici sparsi per tutta Italia. Il loro compito è quello di risolvere le richieste di intervento per conto di Rinnai ai clienti finali. In questo modo è possibile ottenere assistenza qualificata su tutto il territorio senza però risultare in

elevati costi di mantenimento da parte dell'azienda.
I CAT sono anche incaricati di effettuare i collaudi e interventi in garanzia per conto di Rinnai e sono solitamente indirizzati dall'azienda stessa verso il cliente finale che richiede l'assistenza.
La procedura standard d'intervento consiste nei seguenti passaggi.

- Viene effettuata una richiesta di assistenza da parte dell'utente finale, il quale contatta direttamente Rinnai.
- Rinnai fornisce i dati all'utente finale del CAT di competenza della propria zona e, nel caso di collaudi, comunica al CAT la richiesta di intervento
- Il CAT interviene e compila i dati relativi all'intervento, inviandoli nuovamente a Rinnai che provvederà al loro salvataggio e ad eventuali valutazioni della garanzia o rimborso dell'intervento.

1.2.4 Utilizzatori

All'interno della logica business-to-business applicata da Rinnai, l'utente finale non ha un focus dedicato alla creazione di vendite, ma è comunque necessario che questo riceva la migliore assistenza possibile durante le fasi di installazione, collaudo e gestione della garanzia dell'apparecchio, in modo da ottenerne una buona fidelizzazione ai prodotti Rinnai. In seguito alla richiesta di un utilizzatore quindi è utile registrare le attività svolte da quest'ultimo al fine di tracciarne uno storico completo ed ottenere un'analisi ai fini statistici della diffusione del marchio in Italia.

1.3 Hardware informatico

La società ha optato per un approccio di mantenimento interno all'azienda della maggior parte degli applicativi, con una pratica detta software *on premise*. La presenza di una sala server dedicata permette la fruizione dei servizi direttamente dagli stessi, richiedendo sì un alto investimento iniziale, ma ammortizzato nel lungo

periodo. Questo permette all'azienda di essere indipendente per quanto riguarda l'installazione e lo sviluppo di nuovi applicativi sui server aziendali da parte del reparto IT. La rete è inoltre protetta dagli attacchi esterni da un firewall hardware, impiegato anche per limitare l'accesso a file dannosi nella rete interna (es. download di file *.exe* da domini che non utilizzano SSL).

1.4 Applicativi software

È stata premura del reparto IT dell'azienda separare ogni applicativo in macchine virtuali dedicate, in modo tale da ridurre al minimo i disservizi nel caso di malfunzionamenti, pur garantendo la massima interoperabilità.
Tutti gli applicativi fanno riferimento ad un sistema di *single sign-on*, basato su JA-SIG CAS (*Central Authentication Service*), il quale permette di utilizzare le stesse credenziali per accedere ai vari servizi. Il sottosistema di riferimento è quello di Active Directory di Microsoft, un servizio di rete distribuito su tutti i sistemi operativi Windows a partire da Windows 2000 Server. Tramite questo meccanismo, gli utenti ottengono le credenziali di accesso ai software aziendali direttamente dalla combinazione username/password utilizzata per accedere al proprio computer. Al primo accesso verrà inoltre creato un account fittizio all'interno del database dell'applicativo, rimanendo tuttavia trasparente rispetto a cambi di password o informazioni personali. Per ulteriori informazioni sui database impiegati si veda la sezione dedicata nel capitolo 3 (sez. 3.1.2.4).

1.5 Situazione iniziale

L'applicativo CRM installato precedentemente in Rinnai Italia era rappresentato da *SuiteCRM* versione 7.2.2, una versione datata maggio 2015. Il software era legato all'utilizzo della versione PHP 5.4, avente raggiunto end-of-life a settembre 2015 e per questo esponente l'intero lato server ad una serie di vulnerabilità [4, 5].
Il database del CRM contiene i dati di una grande quantità di aziende e contatti,

raccolti durante fiere e campagne pubblicitarie, ma anche inseriti da dipendenti esterni al settore commerciale, come ad esempio gli operatori di centralino e numero verde in seguito alle telefonate da centri assistenza e utenti finali, oppure tramite importazione automatica dai vari strumenti utilizzati in azienda, come il software gestionale, i cui dati devono sempre avere validità fiscale, e dello strumento di registrazione dei collaudi.
È stato quindi necessario tenere in considerazione questi strumenti già avviati in azienda, al fine di non effettuare modifiche *code-breaking* ai campi ed alle tabelle già presenti.

Capitolo 2

Customer Relationship Management

2.1 Definizione

Il **CRM** (*Customer Relationship Management*) è una strategia aziendale, parte del processo manageriale chiamato "marketing relazionale", utilizzato al fine di gestire le relazioni con il pubblico di un'azienda.
In letteratura esistono numerose definizioni di ciò che una pratica di customer relationship management dovrebbe significare; in seguito all'attività di ricerca ne vengono riportati alcuni estratti.

> *" ... strategia aziendale progettata per ottimizzare la redditività, le vendite e la soddisfazione dei clienti (customer satisfaction), e basata sull'organizzazione aziendale focalizzata sul cliente, la promozione della soddisfazione dei clienti e la gestione dei processi che collegano clienti e fornitori. "* ***- Gartner Group*** *- [6]*

> *" [...] mettere in atto una serie di processi organizzativi aventi come obiettivo la relazione, quindi la conoscenza, del cliente e dei fatti che lo coinvolgono. "* - ***M.Duse*** *- [7]*

> *“ I sistemi CRM, sono sistemi informativi/informatici dedicati alla gestione delle relazioni con i clienti: questi sistemi gestiscono le informazioni, sia nell’area commerciale, sia in quella delle attività di marketing, che in quella relativa alle attività di pre e post vendita. ”* - ***G. Motta*** - *[8]*

Ciò che infine emerge da numerosi studi è che negli ultimi decenni i cambiamenti del mercato, passando da un’economia di offerta ad una di domanda, hanno modificato il focus delle imprese in uno scenario nel quale il cliente deve essere posto al centro e per cui si rende necessario adottare approcci di gestione del rapporto col cliente. In questo modo l’utente fidelizzato disporrà di altri metodi di valutazione del prodotto o servizio ricevuto che esclusivamente il prezzo finale.
In un’azienda non orientata alla filosofia CRM, in seguito alla fase di raccolta dei dati relativi ad un possibile acquirente, potrebbero verificarsi diverse situazioni:

- non siano registrati in nessuna forma;
- non siano ricondivisi nell’ambiente aziendale;
- siano registrati in formati di difficile ricerca;
- siano registrati su sistemi non orientati alla condivisione, come fogli elettronici o note su dispositivi portatili.

Il CRM risulta quindi uno strumento fondamentale per il team marketing e gli agenti commerciali, in quanto viene utilizzato per aumentare la fidelizzazione del singolo cliente, per aumentare il parco clienti esistente tramite la creazione di nuovi contatti e per mantenere lo storico di tutte le attività svolte da e per essi, in modo da migliorare la produttività aziendale ed ottenere un incremento significativo delle vendite.

2.2 Storia

Il concetto di marketing nasce negli anni ’50 e ’60 come strategia fortemente focalizzata sul prodotto e basata sui mezzi di comunicazione di massa. In questo momento storico la relazione con il cliente non ha forte valore, in quanto il messaggio che l’azienda vuole trasmettere è diretto indistintamente a tutto il pubblico.

Con l'aumento della concorrenza nel libero mercato, nascerà tuttavia la necessità di caratterizzare un cliente per raggiungerlo nel modo più efficace, tramite l'adozione di tecniche orientate alla trasformazione di un cliente occasionale in un cliente fedele al brand. Fino all'inizio degli anni Ottanta, la gestione clientelare delle piccole e medie aziende era principalmente organizzata su liste e raccoglitori di documenti cartacei, mentre le società con fatturati elevati iniziavano ad avere la possibilità di organizzare i loro dati tramite l'impiego di mainframe ed i primi applicativi per la creazione di fogli elettronici. È interessante da sottolineare la presenza costante nelle aziende di un dispositivo inventato verso la metà degli anni Cinquanta, chiamato *rolodex*, termine coniato da *rolling index*, nel quale spesso venivano registrati i dati dei clienti sotto forma di biglietti da visita (fig 2.1).

Fig. 2.1: Rappresentazione grafica di un dispositivo rolodex

Verso la metà degli anni '80 la diffusione capillare del personal computer e dell'informatica nei processi lavorativi permise la creazione del concetto di *database marketing*, tramite la memorizzazione delle risposte del cliente ottenute tramite diverse forme di *direct marketing* (sondaggi, chiamate, e-mail) e del suo grado di soddisfazione complessivo. Nel 1986, Mike Muhney e Pat Sullivan, fondatori dell'azienda *Conductor Software*, decisero di sperimentare un nuovo approccio creando un progetto di software gestionale, inizialmente chiamato "YES", acronimo di "Yes, Everybody Sells!". Dal primo prototipo nacque "ACT", prima acronimo di "Activity Control Technology", successivamente "Automated Contact Tracking", fino a venire rappresentato solamente come "ACT!". "ACT!", descritto inizialmente come un "rolodex digitale" dai suoi utenti, permise per la prima volta una conservazione efficiente dei informazioni dei contatti ed arrivò ad essere molto apprezzato dal pubblico, vincendo inoltre nu-

merosi premi.
Si ha negli anni '90 la nascita vera e propria dell'acronimo CRM, passando attraverso le denominazioni precedenti *enterprise customer management* (ECM) e *customer information system* (CIS). In questo periodo società come la Siebel Systems iniziarono ad offrire software per *sales force automation* (SFA), integrando la gestione dei contatti con un servizio che permettesse la memorizzazione di informazioni riguardo clienti e competitori sul mercato. Una tendenza facilmente identificabile è quella del passaggio a CRM web-based, come avvenuto per diversi applicativi commerciali, nei primi anni 2000. I vantaggi di questo tipo di soluzioni sono numerosi: accesso non vincolato ai singoli computer aziendali, utilizzo efficiente delle licenze utente e facile integrazione con altri software aziendali.
La creazione del primo applicativo CRM open source risale al 2004 con *SugarCRM*, il quale arriverà ad attrarre oltre 123 milioni di dollari di investimento negli anni successivi da compagnie come Goldman Sachs [9, 10]. Rilasciato successivamente con licenza pubblica GNU (GPLv3), questo software ha aperto la strada per la creazione di diversi CRM open-source basati su suoi fork.
Ad oggi, l'implementazione di un CRM in ambito aziendale, affiancato da politiche che mettano il cliente al centro del focus aziendale, è sempre più una necessità per tutte le aziende che si muovono sul mercato globale.

2.2.1 Breve linea temporale

- 1956 • Invenzione del rolodex
- 1982 • Concetto di *database marketing*
- 1986 • ACT!
- 1993 • Siebel Systems
- 1998 • Oracle Sale Online, successivamente Oracle CRM
- 1999 • Siebel Sales Handheld, primo CRM per palmari
- 1999 • Salesforce
- 2000 • SAP CRM 2.0
- 2001 • Microsoft Dynamics CRM
- 2004 • SugarCRM, primo CRM open source
- 2013 • SuiteCRM

2.3 Implementazione

All'interno dell'azienda, il CRM deve innanzitutto rendere possibile una gestione pratica del rapporto con i possibili acquirenti. Un applicativo di questo tipo deve quindi poter poter rendere disponibile una cronologia delle attività per ogni contatto o azienda in un unico database condiviso dai dipendenti, al fine di ottenere un insieme di dati di rilievo per il reparto commerciale, ma soprattutto deve anche essere fruibile un insieme di *lead*: si intende lead un'azienda o individuo che abbia mostrato interesse verso un prodotto o servizio commercializzato dall'azienda e per questo inteso come potenziale cliente. Questa entità potrebbe, ad esempio, aver richiesto maggiori informazioni tramite telefono o compilazione di un form di richiesta, oppure aver visitato dei contenuti del sito web o partecipato ad una fiera in cui ha avuto contatto diretto con un agente commerciale. In ogni caso, l'individuo in questione risulta a disposizione per ricevere comunicazioni da parte della propria società ed è quindi necessario che i suoi dati vengano registrati, in modo da considerarli per l'invio futuro di campagne di direct marketing.
Il CRM può inoltre rendere disponibile l'integrazione di software esterni già presenti

in azienda, come il gestionale, il client di posta elettronica o l'applicazione documentale, rendendo disponibili funzioni di registrazione delle scadenze e promemoria. Un applicativo moderno deve inoltre risultare accessibile in cloud, quindi tramite browser web o applicazione mobile dedicata.

2.4 Applicativi moderni

Alla data di produzione di questo elaborato (2019), i 4 principali operatori mondiali nell'ambito CRM risultano *SalesForce*, SAP, Oracle e Microsoft. Vengono di seguito analizzati due esempi opposti di software CRM, la soluzione enterprise, orientata maggiormente a realtà aziendali di grandi dimensioni, ed il pacchetto open source, attivamente utilizzato in diversi settori produttivi.

2.4.1 SalesForce

Fig. 2.2: Il logo di SalesForce

SalesForce risulta il leader mondiale per quanto risulta il software CRM. L'azienda insegue fin dalla sua nascita l'idea di *"software as a service"*, ovvero un modello di distribuzione del software nel quale un servizio di hosting di terze parti mantiene online l'applicativo su un server di sua proprietà e lo rende disponibile all'acquirente tramite un client Internet, rappresentato da un qualsiasi browser o da un'applicazione mobile dedicata. La società ad oggi vanta oltre 29'000 impiegati e la sua capitalizzazione del mercato risultava di 102.5 miliardi di dollari ad ottobre 2018 [11].
L'utilizzo di *SalesForce* vincola al pagamento di un canone mensile, calcolato per

utente. Il software è limitato dal punto di vista della personalizzazione, in quanto, non avendo accesso al codice, risulta impossibile modificare l'aspetto del proprio CRM se non tramite diverse opzioni preimpostate.
Nonostante esistano oltre 3'000 estensioni sotto forma di plugin installabili su *Sales-Force* in modo da estenderne le capacità, circa la metà di queste risultano essere a pagamento. L'adozione di questo tipo di software è sicuramente un grosso investimento per qualunque piccola-media impresa italiana, ma anche per la maggior parte delle grandi imprese, dato che maggiore sarà il numero di utenti, maggiore sarà il canone mensile complessivo.

2.4.2 SuiteCRM

Fig. 2.3: Il logo di SuiteCRM

SuiteCRM è un fork software del popolare applicativo open source *SugarCRM*. Nel 2013, *SugarCRM* ha cambiato la sua logica di business, non investendo più nella propria versione open source, al tempo chiamata "Community Edition" (SugarCRM CE), prediligendo piuttosto le proprie versioni a pagamento. *SugarCRM* definisce attualmente il proprio operato come "Commercial Open Source", una pratica già vista nell'ambito OSS come per la distribuzione GNU/Linux Red Hat Entrprise. *SuiteCRM* è basato sull'ultima versione 6.5.5, l'ultima liberamente disponibile di *SugarCRM* prima della sua rimozione dai canali ufficiali di distribuzione del codice sorgente.
Il progetto di sviluppo nasce dalla software house britannica *Sales Agility* ed ha come obiettivo il proseguimento dell'idea iniziale di *SugarCRM*, ovvero la realizzazione di un applicativo stabile e sicuro come i concorrenti, ma che sia anche open

source e quindi di facile modifica da parte della community. Gli unici costi richiesti per l'installazione di un applicativo *SuiteCRM* riguardano le personalizzazioni e gli sviluppi di cui si vuole dotare il proprio software, con la differenza che tali estensioni rimarranno per sempre di proprietà dell'azienda.
Un altro vantaggio nell'utilizzo di questo genere di software, inoltre, è la possibilità di non pagare il canone di web hosting e di installare il proprio CRM direttamente all'interno del server aziendale. *SalesForce*, ad esempio, non permette questa agevolazione e prevede l'utilizzo esclusivo dell'applicativo in cloud.
Il codice di *SuiteCRM* risulta disponibile sulla piattaforma di condivisione del codice GitHub, rilasciato sotto licenza open source Affero GPL, vanta oltre 150 release, 13'000 commit [12] ed un esteso insieme di pagine di documentazione sul sito web ufficiale [13].

Parte II

Progetto e Sviluppo

Capitolo 3

Progetto

3.1 Richieste progettuali

La sezione seguente contiene le richieste di progetto valutate dal reparto commerciale per l'implementazione di un CRM più funzionale di ciò che era presente precedentemente in azienda. Le richieste sono pervenute tramite colloquio diretto o corrispondenza e-mail diretta al reparto ICT dell'azienda.

3.1.1 Requisiti funzionali

RF 1	
Titolo	**Modulo per la gestione delle filiali**
Logica	Nonostante la maggioranza delle aziende memorizzate nel database disponga di un unica sede, è necessario sviluppare un modulo aggiuntivo che permetta di salvare aziende dotate di filiali
Verifica	Inserimento, modifica, visualizzazione di una o più sedi per ogni azienda

RF 2	
Titolo	**Gestione degli eventi**
Logica	È necessario sviluppare un modulo che permetta di memorizzare in modo chiaro gli eventi ai quali partecipa Rinnai Italia, con relativi collegamenti ad aziende e contatti presenti ad essi
Verifica	Inserimento, modifica, visualizzazione di eventi con l'inserimento di dati utili a fini commerciali, con sottopannelli relativi ai partecipanti

RF 3	
Titolo	**Gestione dei corsi**
Logica	È necessario rivedere il modulo dei corsi già implementato, in modo che sia possibile memorizzare i corsi di vario tipo tenuti da Rinnai Italia
Verifica	Inserimento, modifica, visualizzazione dei corsi, con sottopannelli relativi ai partecipanti

RF 4	
Titolo	**Gestione delle visite**
Logica	È necessario sviluppare un modulo che permetta di memorizzare le visite effettuate dagli agenti commerciali presso le varie aziende, con possibilità di dare una valutazione complessiva alla visita ed un campo per eventuali note aggiuntive
Verifica	Inserimento, modifica, visualizzazione delle visite effettuate presso le aziende

RF 5	
Titolo	**Gestione dei premi**
Logica	È necessario sviluppare un modulo che permetta di memorizzare i premi assegnati per vari motivi dal reparto commerciale di Rinnai Italia alle proprie aziende più fidelizzate
Verifica	Inserimento, modifica, visualizzazione dei premi assegnati

RF 6	
Titolo	**Indicazione autorizzazioni privacy**
Logica	Sono necessari campi per registrare le preferenze di privacy specificate dall'azienda nel form di contatto verso Rinnai
Verifica	Modifica e visualizzazione dei flag delle preferenze di privacy

RF 7	
Titolo	**Inserimento valutazioni**
Logica	È necessario un campo per dare una valutazione generica all'azienda, in modo da identificare immediatamente le aziende più fidelizzate ed ordinarle nelle viste di elenco
Verifica	Modifica e visualizzazione di un campo esprimente valutazione

RF 8	
Titolo	**Linee guida relative alla modifica dei campi**
Logica	È necessario che durante la fase di modifica dei dati venga visualizzato un testo di aiuto che permetta all'utente di compilare al meglio l'inserimento o la modifica dei dati di un'azienda o contatto, limitando l'immissione di dati errati
Verifica	Visualizzazione di un campo che aiuto nelle fasi di inserimento e modifica dei dati

RF 9	
Titolo	**Campi chiave per la ricerca**
Logica	È necessario lo svilippo di un tipo di dato che permetta di catalogare e ricercare velocemente le aziende in base a campi chiave di varia natura
Verifica	Modifica e visualizzazione di un insieme di campi chiave per le aziende

3.1.2 Requisiti non funzionali

3.1.2.1 Requisiti di interfaccia

RNF 1	
Titolo	**Responsiveness**
Logica	L'applicativo web deve risultare *responsive*, ovvero adattarsi automaticamente alla dimensione del dispositivo su cui verrà visualizzato
Verifica	Verifica sui dispositivi finali degli utenti e tramite Developer Tools del browser Google Chrome

3.1.2.2 Requisiti prestazionali

RNF 2	
Titolo	**Prestazione generale**
Logica	L'applicativo web deve generare tutte le pagine nel tempo massimo di 1 secondo, escluse parti riguardanti configurazioni, personalizzazioni dell'interfaccia utente e più genericamente salvataggio di dati, per i quali il tempo può aumentare
Verifica	Verifica dei tempi di esecuzione lato client

3.1.2.3 Requisiti di comportamento

RNF 3	
Titolo	**Frequenza delle attività**
Logica	Il sistema non avrà un carico ingente di lavoro, dato il numero ridotto di utenti finali, limitato ai dipendenti dell'azienda
Verifica	Verifica del carico di lavoro lato server

3.1.2.4 Database

I database presenti in azienda sono interamente realizzati su MariaDB, un popolare fork del DBMS MySQL, realizzato dallo stesso programmatore che ha avviato e guidato per molti anni il progetto madre. Questa scelta progettuale semplifica le operazioni di backup e ripristino dei dati, permette l'accesso ai dati da un qualsiasi client compatibile e rende tutti gli applicativi di facile interconnessione. La sicurezza per l'accesso ai database è garantita innanzitutto dall'invisibilità del sistema da parte di un qualsiasi utente malintenzionato non connesso direttamente alla rete aziendale, sia che dal firewall precedentemente citato.

3.2 Tabella decisionale

La seguente tabella, chiamata *CRM Requirements Worksheet* e tratta dal libro *Implementing SugarCRM* [14] è un valido aiuto nell'individuare le necessità della propria impresa durante l'implementazione di un CRM.

Caratteristica	Scelte	Note
Tipo di clientela	☑ B2B ☐ B2C	Rinnai Italia segue una logica business-to-business, ma senza escludere l'attenzione verso l'utente finale. È quindi necessario mantenere sia aziende che contatti.
Origine degli introiti	☑ Prodotti ☐ Servizi	I prodotti devono essere visti come opportunità una tantum e non come pagamenti ricorrenti come nel caso dei servizi
Servizi di supporto	☐ Sì ☑ No	Le richieste di supporto di Rinnai sono già gestite dal software OTRS
Valore singola transazione	☐ Ridotti ☑ Elevate	Se l'azienda effettua singole transazioni con valori tendenzialmente elevati (superiori a €1000) e cicli di vendita
Ciclo di vendita	☐ Corto ☑ Lungo	di due settimane o più, risulta utile utilizzare i moduli Lead e Opportunità per misurare il valore complessivo degli introiti
Locazione delle vendite	☐ Singola ☑ Multipla	Anche se Rinnai Italia dispone di un'unica sede principale, i suoi agenti si spostano tra i clienti regolarmente. È quindi necessario che il CRM sia fruibile su tutti i dispositivi portatili in dotazione
Gestione del personale	☑ Sì ☐ No	Si sceglie di utilizzare questa funzionalità per salvare i dati essenziali del personale, come numeri di telefono e indirizzi

Caratteristica	**Scelte**	**Note**
Lead tracking	☑ Sì ☐ No	Si ritiene utile mantenere il modulo dei lead tenere traccia dei potenziali acquirenti futuri
Gestione delle attività	☑ Sì ☐ No	Si intende fare uso dei moduli del calendario e di pianificazione delle riunioni
Interfaccia estesa	☐ News feed ☐ Feed finanziario ☐ Email	Non si ritiene necessario sfruttare queste funzionalità nella dashboard iniziale, tanto meno il client email integrato, in quanto si dispone già di quest'ultimo ad un indirizzo esterno al CRM
Gestione documentale	☐ Sì ☑ No	Rinnai Italia dispone già di software per la gestione documentale, per cui è possibile disattivare i moduli integrati per questa funzionalità
Utenti internazionali	☐ Sì ☑ No	Nonostante sia possibile che individui di madre lingua non italiana visitino l'azienda, non si ritiene necessario curare la parte in inglese durante lo sviluppo del CRM. Se necessario sarà possibile rivedere questa funzionalità con modifiche minime alle etichette grazie alla facilità di configurazione di *SuiteCRM*

3.3 Architettura di SuiteCRM

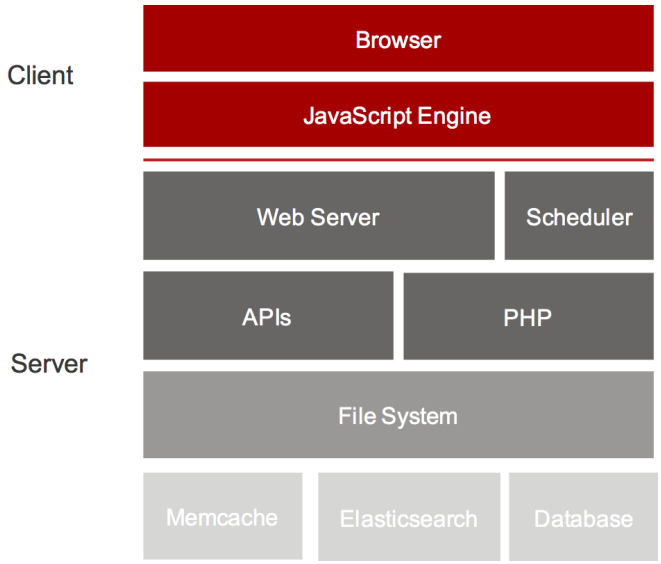


Fig. 3.1: Rappresentazione dell'architettura client-server dietro SuiteCRM

SuiteCRM è un'applicazione open source, interamente gestibile via browser, completa nelle funzionalità implementate quali la gestione dei dati di aziende e contatti, dei contratti, delle opportunità di vendita, la pianificazione di riunioni, attività ed appuntamenti. Il prodotto offre la possibilità di scelta del tema grafico all'utente finale, oltre che un supporto multilingua, e comprende funzioni per l'importazione e l'esportazione di dati.

Essendo web-based (fig. 3.1), *SuiteCRM* non richiede particolari vincoli di compatibilità: può essere eseguito su qualsiasi sistema operativo che disponga di un web server Apache ($\geq$ 2.2) o IIS ($\geq$ 8.0) [15] ed è costruito tramite linguaggio PHP ed al DMBS MySQL per la parte back-end, mentre fa uso di HTML, CSS e JavaScript per la parte front-end. Sono inoltre presenti il template engine *Smarty* e la nota libreria *jQuery*, che semplifica le operazioni via client dove possibile.

Il software garantisce il controllo, attraverso un insieme di pagine che permettono di avere velocemente a disposizione tutte le informazioni più rilevanti in relazione al singolo utente, come ad esempio, le opportunità commerciali in corso, i clienti, il

calendario, le campagne marketing.
Dopo l'analisi della situazione iniziale, si è scelto di non virare verso altre soluzioni CRM disponibili sul mercato, ma di aggiornare il software già installato e di svilupparne una successiva estensione per ragioni di mantenimento dei dati già presenti senza passare attraverso procedure di conversione, pur mantenendo la volontà di sviluppare un applicativo interno all'azienda, senza l'appoggio di fornitori esterni per la gestione, garantendone comunque una facile e veloce personalizzazione a carico del reparto IT.

3.3.1 Albero dei file

Viene di seguito descritta brevemente la struttura di un'installazione standard di *SuiteCRM*.

cache contiene i file precompilati della cache di *SuiteCRM*, utilizzati al fine di ottenere un miglioramento della performance generale, riducendo il numero di file che necessitano una interpretazione a runtime; questi includono template Smarty, Javascript e CSS in forma *minified*

custom contiene le personalizzazioni realizzate dal developer e dall'utilizzatore; può inoltre contenere codice realizzato da *SuiteCRM* al fine di mantenere la compatibilità con *SugarCRM*

data contiene le classi ed i file utilizzati per gestire i SugarBeans e le loro relazioni

examples contiene alcuni esempi base per interfacciarsi con l'API offerta dal software

include contiene una grande quantità di moduli fondamentali per l'uso del CRM, tra cui i tipi di dato ed i widget mostrati nella bacheca iniziale

install contiene file utilizzati durante l'installazione del CRM e dal pannello di amministrazione in seguito

jssource contiene i file sorgente in versione non minified di alcuni del codice JavaScript utilizzati

metadata contiene i metadati delle relazioni molti-a-molti incluse di default nei moduli di *SuiteCRM*

ModuleInstall codice per l'installatore dei moduli

modules contiene il codice per i moduli di *SuiteCRM*, sia quelli stock che quelli creati successivamente con il costruttore di moduli

themes contiene il codice e le immagini utilizzate per i temi del CRM

upload la cartella upload contiene i file che vengono caricati su SuiteCRM tramite form. La cartella upgrades al suo interno conterrà inoltre i file caricati durante la procedura di aggiornamento o l'installazione dei moduli

service codice per le API SOAP and REST v2

XTemplate template engine utilizzato inizialmente da *SugarCRM* e successivamente rimpiazzato da Smarty, ma mantenuto per retrocompatibilità

Zend framework object-oriented PHP di componenti riutilizzabili

3.3.2 Schema Model-View-Controller (MVC)

SuiteCRM utilizza il diffuso design pattern modello-vista-controllo (MVC), permettendo allo sviluppatore di estendere facilmente le funzionalità del software. Questo schema permette di ottenere una divisione tra i componenti del sistema in tre aree funzionali, separando le logiche dei dati (model), l'aspetto dell'interfaccia utente (View) e le operazioni sul database (Controller). Il sistema è costruito sulla base del framework di *SugarCRM CE 6.5*.

3.3.2.1 Model

Il componente Model è rappresentato dai SugarBean ed ogni relativa estensione della classe. Il modello SugarObject estende ulteriormente il metodo di sottoclasse e permette la creazione di sottoclassi nei vardefs. Questo include l'ereditarietà di campi, relazioni ed indici, ma non imitando tutto questo ad una singola istanza. Esistono 6 tipi di template Sugar Object:

Basic contiene i campi base richiesti da tutti i moduli

Person basato sui moduli Contatti e Leads

Issue basato sul modulo Bugs e problemi

Company basato sul modulo Aziende

File basato sul modulo Documenti

Sale basato sul modulo Opportunità

3.3.2.2 View

Le View vengono normalmente utilizzate per mostrare dati all'utente o permettergli di eseguire azioni. Esistono 3 viste principali disponibili per un modulo:

Detail View La Detail View (fig. 3.2) mostra i dettagli di un oggetto e mostra i relativi sottopannelli collegati. I sottopannelli sono a tutti gli effetti List View che mostrano un set personalizzato di oggetti e che sono collegate all'oggetto padre. `./<modulo>/metadata/detailviewdefs.php` gestisce il layout della Detail View del un modulo e `./<modulo>/metadata/subpaneldefs.php` definisce i sottopannelli che vengono mostrati nella Detail View di un modulo.

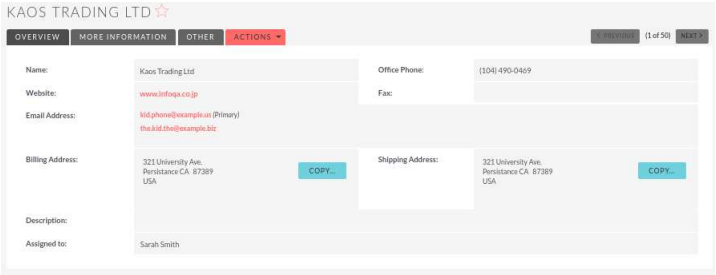


Fig. 3.2: Detail View di un elemento

List View questa view (fig. 3.3) gestisce la visualizzazione ad elenco di un modulo ed verrà utilizzata anche per mostrare i risultati di un'operazione di ricerca. L'utente vedrà questa schermata di default quando aprirà un modulo del CRM. Da qui l'utente può selezionare uno specifico record dell'elenco per visualizzarne i dettagli, oppure effettuare azioni di eliminazione, modifica ed esportazione per record multipli, selezionando la relativa checkbox di selezione.

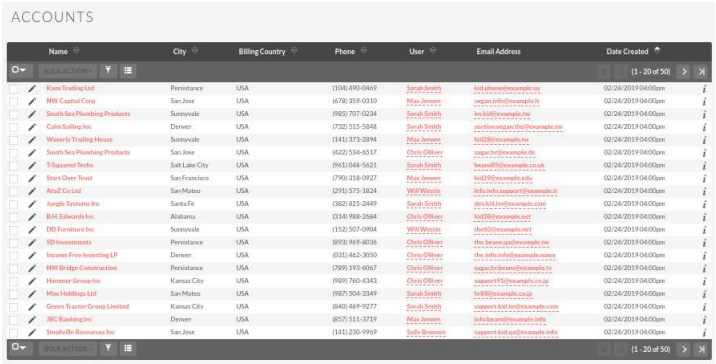


Fig. 3.3: List View di un modulo

Edit View l'Edit View (fig. 3.4) viene usata quando l'utente crea un nuovo record oppure quando vengono modificati i dati di uno già esistente. L'Edit View può anche essere raggiunta direttamente dalla List View. Il file `./<modulo>/metadata/editviewd` gestisce il layout dell'Edit View del modulo e la sua visualizzazione può essere sincronizzata dal pannello di amministrazione con la DetailView, in modo che entrambe mostrino gli stessi campi.

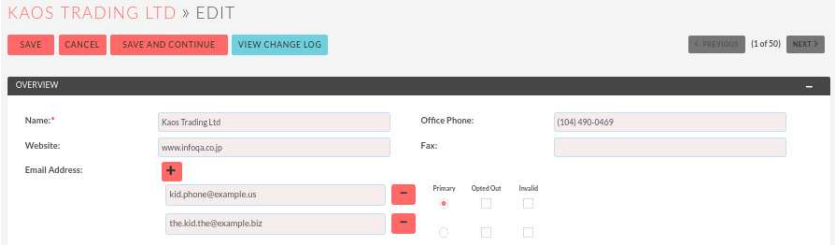


Fig. 3.4: Edit View di un elemento

3.3.2.3 Controller

Il controller principale (*SugarController*) gestisce le azioni ricevute dall'Edit View per il salvataggio di un record. Ogni modulo può effettuare l'override e quindi l'estensione del controller tramite l'inserimento del file `controller.php` all'interno della directory del modulo.
Esistono 3 azioni principali su cui lo sviluppatore può effettuare modifiche.

pre_save lo sviluppatore può effettuare l'override dei parametri di popolamento del form

action_save questa azione fornisce allo sviluppatore pieno controllo sul salvataggio del record

post_save questa funzione gestisce il successivo setup della view

Quando la modalità sviluppatore viene disattivata, il VardefManager costruisce la cache dei vardef in un unico file che verrà caricato a run time. Se il file della cache non viene trovato, allora verrà selezionato il file posizionato nella directory del modulo. Un processo analogo viene svolto per tutti i custom fields e ogni altro tipo di estensione all'interno di `custom/ext`.

3.4 Scelte progettuali

Questa sezione descrive le scelte affrontate durante la progettazione dei moduli, prima di arrivare alla fase di implementazione vera e propria nel capitolo seguente.

3.4.1 Progettazione dei moduli

Il focus principale dell'attività di progettazione si è concentrato sul modulo *Aziende* e su ciò che esso avrebbe dovuto mostrare. È infatti tramite esso che i reparti commerciale e vendita potranno avere maggior riscontro dell'implementazione di un CRM funzionante nella loro attività.

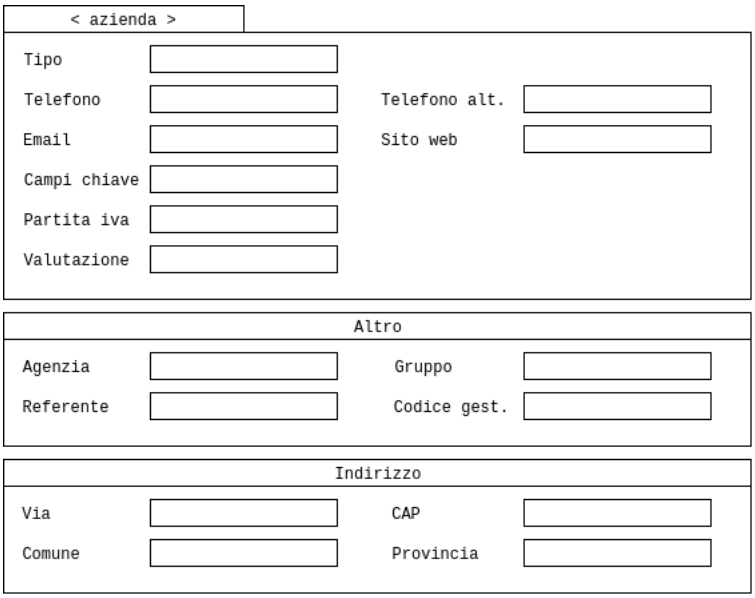


Fig. 3.5: layout atteso per visualizzazione e modifica di un'azienda

I campi identificati, come mostrato in fig. 3.5, consistono nell'insieme delle informazioni chiave di una società (nome, tipo, telefono, etc.), nell'indirizzo, più un insieme di campi destinati a registrare collegamenti con altre aziende: l'agenzia rappresenta un intermediario molto utilizzato nell'ambito termoidraulico come collegamento tra rivenditori e clienti, mentre il gruppo rappresenta un conglomerato di più compagnie di cui essa fa parte. Il referente aziendale è un collegamento con un dipendente di Rinnai, come ad esempio il manager dell'area geografica dove l'azienda ha sede. Infine, lo spazio destinato ai campi chiave servirà per soddisfare il requisito di ricerca rapida, mentre la valutazione permetterà di registrare con una valutazione

da 1 a 10 il rapporto che l'azienda ha con Rinnai.

La pagina dell'azienda dovrà inoltre interfacciarsi con diversi moduli aggiuntivi.

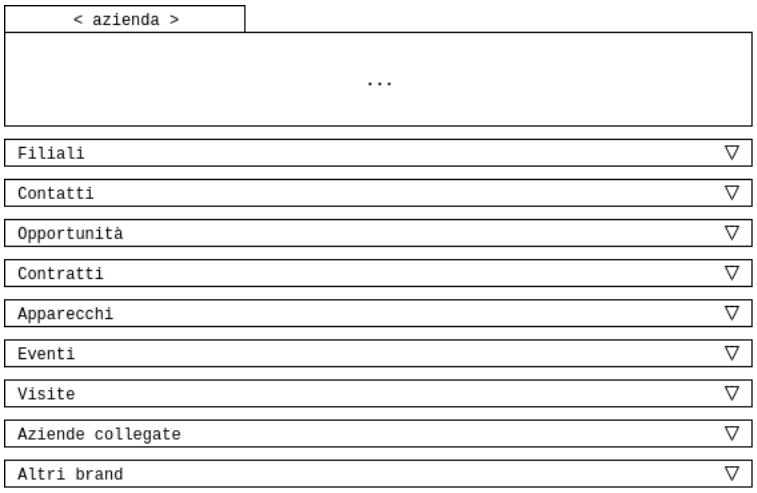


Fig. 3.6: layout di un'azienda con sottopannelli

Nella figura 3.6 vengono mostrati i vari sottopanelli che potranno fornire informazioni rilevanti sui collegamenti della società.

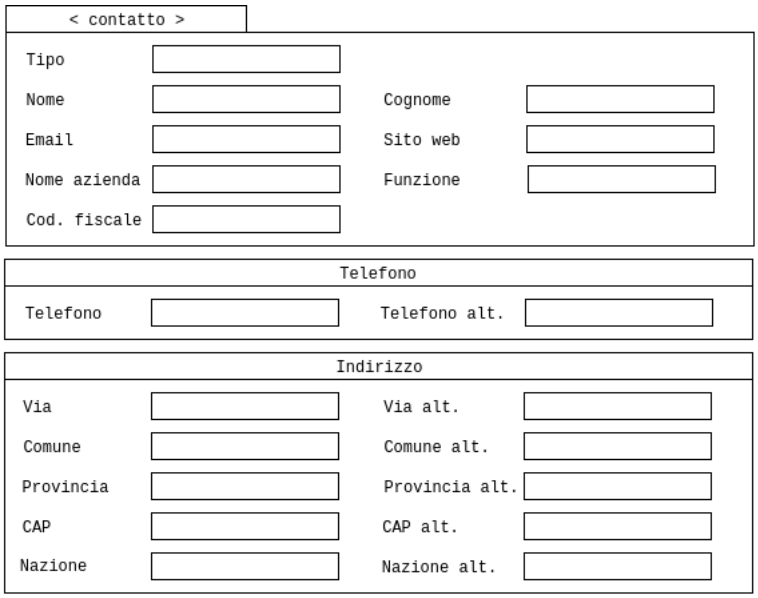


Fig. 3.7: layout di un contatto

Un approccio simile è stato utilizzato per delineare i campi utili nella registrazione di un contatto (fig. 3.7).

3.4.2 Campi richiesti

Ogni agente dovrà poter applicare un voto sommario all'azienda; tale valutazione verrà successivamente visualizzata tramite un'immagine facilmente comprensibile dall'utente.
Il campo di ricerca richiede che ogni azienda venga catalogata da diverse chiavi tramite selezione multipla. È stato quindi utilizzato lo stesso processo seguito dagli *hashtag*, un campo impiegato nei social network moderni per il raggruppamento dei post (fig. 3.8).

Fig. 3.8: progetto per la realizzazione del campo di ricerca

È stato inoltre identificato come requisito utile l'inserimento di un pannello dedicato alla registrazione rapida di note sul cliente (fig. 3.9), eventualmente riutilizzabile anche per altri moduli dell'applicativo.

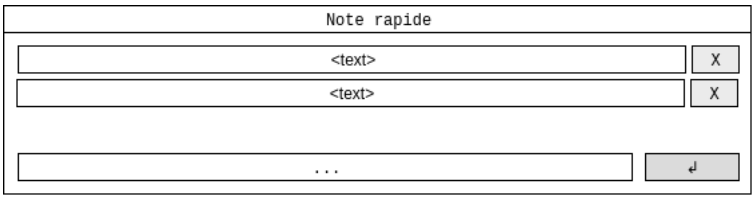
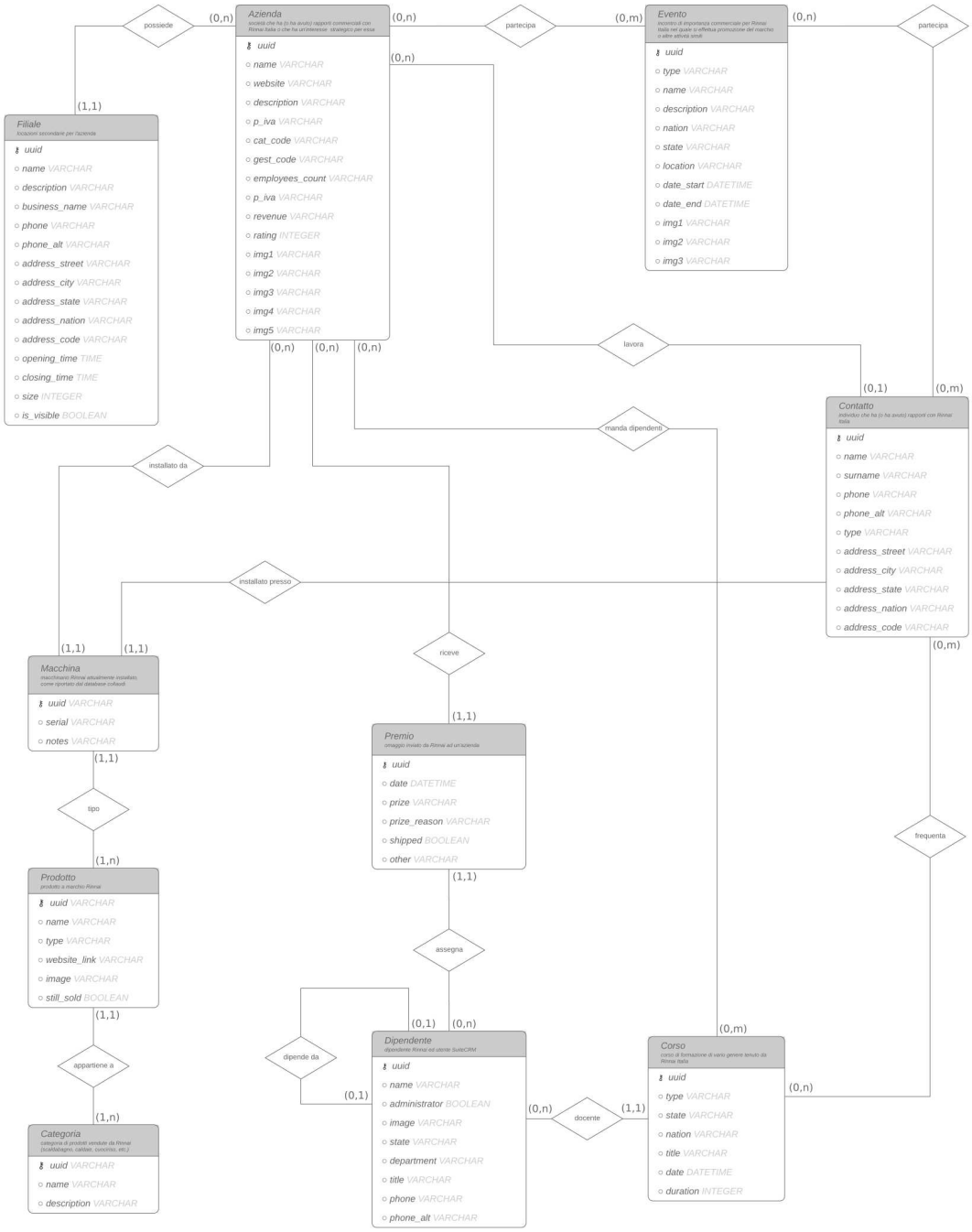


Fig. 3.9: progetto per il pannello dedicato alle note

Come illustrato nella sezione dedicata all'implementazione, la via più pratica per effettuare modifiche sul CRM risulta la creazione di nuovi tipi di dato, tramite i quali verranno gestite la visualizzazione e la modifica dei valori attribuiti a tali campi.

3.5 Diagramma ER



Capitolo 4

Implementazione

4.1 Aggiornamento di un'installazione precedente

SuiteCRM offre la possibilità di effettuare un aggiornamento di una versione già installata del software tramite pacchetti di patch. Come precedentemente specificato, essendo già presente in azienda una versione datata del software, è stato necessario eseguire un adeguato procedimento, i cui punti vengono elencati di seguito:

- effettuare un backup dell'installazione corrente di *SuiteCRM* prima di iniziare l'aggiornamento
- disabilitare ogni tipo di op-cache di PHP
- aumentare il tempo massimo di esecuzione di PHP tramite *max_execution_time* e *max_input_time*
- aumentare la dimensione massima di upload dei file, sull'installazione corrente di *SuiteCRM* tramite il menù di amministrazione e su PHP tramite *post_max_size* e *upload_max_filesize*
- aumentare la memoria massima concessa a PHP tramite *memory_limit*
- effettuare un riavvio del servizio Apache

L'aggiornamento è iniziato dalla versione installata (7.2.2) ed è proseguito senza difficoltà con le successive 7.3.3, 7.4.4, 7.6.10. A partire dalla 7.8.26 è stato necessario effettuare l'upgrade della versione PHP installata sul server; la versione minima richiesta sarebbe *PHP 5.6*, ma avendo questa raggiunto EOL a dicembre 2018, quindi non ricevente più patch di sicurezza, si è deciso di effettuare l'upgrade (codice 4.1) direttamente alla versione sviluppata attivamente, *PHP 7.3*.

```
yum install yum-utils
yum-config-manager --enable remi-php73
yum update
```

Codice 4.1: codice per l'attivazione della repository PHP 7.3 su CentOS

I pacchetti successivi permettono quindi di continuare l'aggiornamento alla versione 7.9.17 e alla 7.10.13.

4.2 Strumenti integrati

SuiteCRM dispone di tre componenti di sviluppo, il cui scopo è ridurre il lavoro effettivo sul codice e permettere alcune modifiche anche ad un utente dotato dei permessi di amministrazione. Questi strumenti sono *Module Builder*, *Studio* e *Dropdown Editor*.

4.2.1 Module Builder

Module Builder è uno strumento grafico utilizzato per la costruzione di un modulo personalizzato al fine di utilizzarlo all'interno della propria installazione di *SuiteCRM* oppure di esportarlo per aggiungerlo su un'altra istanza del software. Module Builder organizza i moduli in "pacchetti", ognuno dei quali può contenere uno o più moduli (fig. 4.1). Scegliendo di creare un nuovo modulo, si noterà che il sistema prevede l'utilizzo di alcuni template (Base, Azienda, File, Persona, Vendita) al fine di ridurre il tempo utilizzato nell'introduzione di campi basi.

Fig. 4.1: Module Builder in *SuiteCRM* 7.11

La creazione di un modulo condivide buona parte degli strumenti con il modulo Studio, il quale verrà analizzato di seguito.

4.2.2 Studio

Il modulo Studio di *SuiteCRM* (fig. 4.2) permette la creazione di numerose personalizzazioni. In seguito alla selezione del modulo da modificare, verranno visualizzate i seguenti strumenti di modifica.

Fig. 4.2: Studio in *SuiteCRM* 7.11

Etichette consente la modifica delle etichette del modulo corrispondente nella lingua scelta.

Campi strumento per aggiungere, rimuovere o modificare campi personalizzati,

oltre a permettere la modifica delle etichette associate a ciascun campo. I campi presenti di default non potranno essere modificati, oltre che per modifiche all'etichetta o al tipo di scelte possibili nel caso di un menù dropdown.

Relazioni Mostra le relazioni tra il modulo corrente ed altri moduli del sistema e consente la creazione di nuovi collegamenti. I collegamenti presenti di default normalmente non potranno essere modificati tramite questo strumento.

Maschere permette la modifica alle varie view che costituiscono un modulo. Possono essere personalizzate la maschera di dettaglio, la maschera di modifica e le maschere di ricerca, in modalità base ed avanzata.

Sottopannelli viene utilizzato per modificare i sottopannelli mostrati sulla maschera di dettaglio del modulo corrente. La rimozione dei sottopannelli non è disponibile tramite questo modulo e dovrà essere effettuata tramite modifiche al codice.

4.2.3 Dropdown Editor

Questo strumento permette la modifica dei menù dropdown del sistema. Tutti i valori in questo tipo di menù potranno essere aggiunti, rimossi o modificati senza andare ad agire direttamente sui file di configurazione della lingua.

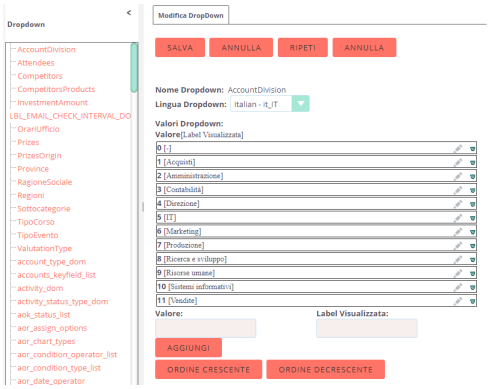


Fig. 4.3: Editor dei menù dropdown in *SuiteCRM* 7.11

4.3 Ambiente di sviluppo

Oltre che con gli strumenti offerti da *SuiteCRM* sopra citati, le modifiche al codice sono state apportate direttamente lato server, tramite collegamento `ssh` ed editor `vim`, o tramite commit ad una repository `git` creata appositamente sul server di sviluppo (codice 4.2) e clonata sul computer locale (codice 4.3).

```
1 git init
2 git config --global user.email "bisi.simone@gmail.com"
3 git config --global user.name "Simone"
4 git add --all .
5 git commit -m "Primo commit"
6 git push
```

Codice 4.2: comandi per l'inizializzazione della repository lato server

```
1 git init
2 git remote add origin ssh://root@192.168.0.37:22/var/www/html/suitecrm
3 git pull origin master
```

Codice 4.3: comandi per la copia della repository lato client

4.4 Creazione dei tipi di dato

Per soddisfare completamente le richieste del reparto commerciale si è resa necessaria la creazione di tipi di dato personalizzati.
I SugarField sono oggetti che effettuano l'interpretazione del campo specificato nel file dei metadati. I tipi di dato presenti di default possono essere trovati in `include-/SugarFields/Fields`. Al suo interno vi si trova un insieme di cartelle che effettuano il rendering dei diversi tipi di dato (Date, Text, Enum, etc.). Il template che si trova nella cartella `Base` contiene i metodi fondamentali che verranno poi estesi tramite il meccanismo di sottoclassi per effettuare la visualizzazione del campo nella DetailView.
I SugarField sono interpretati dal framework Sugar quando nelle varie componenti del MVC (EditView, DetailView, ListView) sono invocate per un determinato mod-

ulo. Le cartelle del tipo di dato contengono normalmente un insieme di file scritti nel linguaggio template Smarty con conseguente supporto all'HTML per la visualizzazione del campo. Il SugarField può anche contenere un file PHP che abbia il nome convenzionale `SugarField<tipo di dato>.php` per effettuare l'override dei metodi del tipo `Base` e quindi aggiungere ulteriori controlli all'interpretazione dei dati.

4.4.1 Hashtag

Per la creazione di questo tipo di dato si è utilizzato *Selectize.js* [16] . Costruita su jQuery e dalla dimensione contenuta in 7kb, questa libreria è distribuita pubblicamente su GitHub con licenza Apache 2.0 e permette la personalizzazione del campo HTML `<select>` in diverse forme. Per ottenere il risultato desiderato si è fatto uso della funzione "Tagging" combinata con il plugin "remove_button".

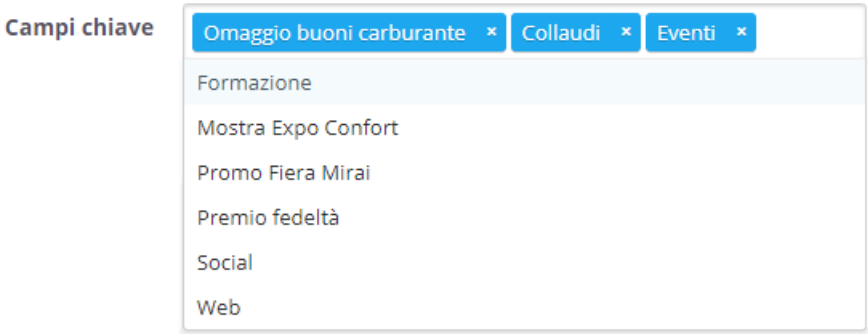


Fig. 4.4: Dimostrazione del tipo di dato Hashtag

Tramite codice PHP personalizzato, il tipo di dato va ad interfacciarsi con i menù dropdown tramite l'array globale `$GLOBALS['app_list_strings']` ed ottiene la lista dei possibili valori tramite un indice composto dal nome del modulo seguito dal nome del campo.

4.4.2 Semaphore

Questo tipo di dato si è reso utile per mostrare all'utente finale una valutazione di immediato riconoscimento tramite i colori convenzionalmente accettati per i semafori: rosso per criticità elevata, giallo per media urgenza, verde per uno stato meno importante. La DetailView fa uso di tre immagini caricate sul server (fig. 4.5), mentre l'EditView è composta da un menù dropdown.

Fig. 4.5: Aspetto del tipo Semaphore

4.4.3 Rating

Ai fini di creare un campo che permettesse una visualizzazione rapida da parte dell'utilizzatore del gradimento verso una certa società è stato creato un campo contenente una valutazione generica da 0 a 10. Il dato viene memorizzato nel database come intero (`INTEGER`) e successivamente la sua visualizzazione viene gestita dal CRM tramite i caratteri `ASCII` "☆" (black star, U+2605) e "★" (white star, U+2606) che, essendo presenti in Unicode 1.1 (giugno 1993), garantiscono la compatibilità con qualsiasi tipo di browser presente e futuro (fig. 4.6).

4.4.4 Partite IVA

Per facilitare l'inserimento dei dati relativi ad un'azienda è stato realizzato un tipo di dato che prendesse in input la partita IVA di un'azienda e, oltre a verificarne la

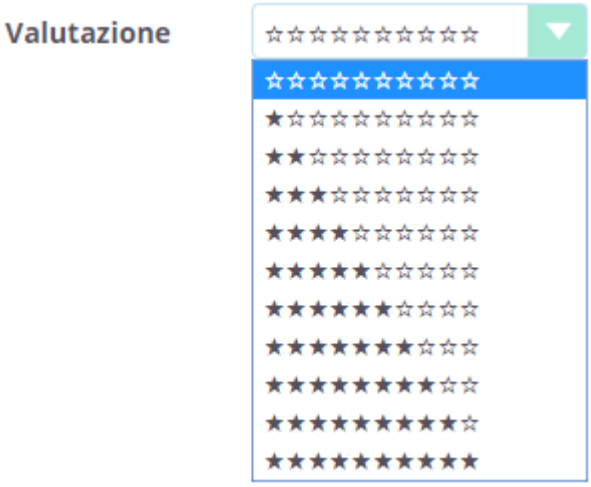


Fig. 4.6: Dropdown del tipo di dato Rating

possibile invalidità, fornisse informazioni utili per il completamento della scheda di un'azienda.

Inizialmente si è tentato di effettuare la richiesta tramite il sito dell'Agenzia delle Entrate (`telematici.agenziaentrate.gov.it/VerificaPIVA/Scegli.do`), ma questo prevede l'inserimento di un codice CAPTCHA, impedendo l'invio di richieste automatizzate. In seguito ad ulteriori ricerche si è scelto di utilizzare lo strumento dell'Unione Europea VIES (*VAT Information Exchange System*), inviando una richiesta a `ec.europa.eu/taxation_customs/vies/vatResponse.html` ed effettuando l'elaborazione del risultato ottenuto. La richiesta viene effettuata tramite le API `XMLHttpRequest`, elemento fondamentale della tecnica AJAX (*Asynchronous JavaScript and XML*) [17]. All'interno del template Smarty viene effettuata una richiesta dal client, il quale si interfaccerà con il file `check.php`. In seguito, elaborato l'output ottenuto con uno script JavaScript, suddividendolo in 4 campi: ragione sociale, indirizzo, CAP, città (fig. 4.7).

Essendo già presente un campo per la registrazione della partita IVA è stato modificato il tipo di dato interno visualizzato da *SuiteCRM* tramite il file `vardefs.php` per il modulo `Accounts`, evitando di doverne creare uno nuovo ed effettuare una successiva copia. Avendo mantenuto lo stesso tipo di dato memorizzato dal database

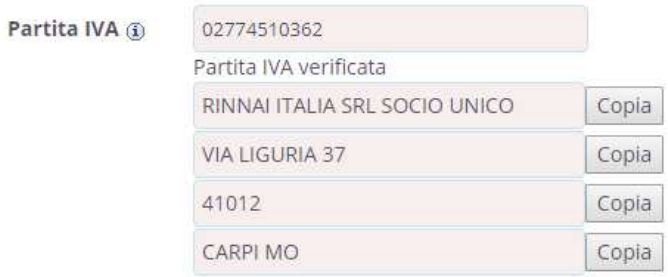


Fig. 4.7: Inserimento di una partita IVA valida

precedentemente, ossia il tipo di dato `VARCHAR`, non è stato necessario effettuare altre operazioni sul database e la conversione è risultata immediata. Il codice responsabile per questo tipo di dato è mostrato nell'appendice A.

4.4.5 Note rapide

Il seguente tipo di dato è stato realizzato allo scopo di ottenere un campo per l'inserimento di note rapide. La richiesta, inviata tramite il form, viene elaborata da PHP e permette la comunicazione con il database tramite query personalizzate; ciò consente di agire direttamente sulla *DetailView* senza passare per l'*EditView*, velocizzando l'operazione di inserimento.

Fig. 4.8: Pannello per la visualizzazione e l'inserimento delle note

Questo campo può dunque risultare un valido aiuto agli agenti commerciali, al fine di memorizzare in modo rapido alcune informazioni non classificabili nei vari moduli collegati.

4.4.6 Map

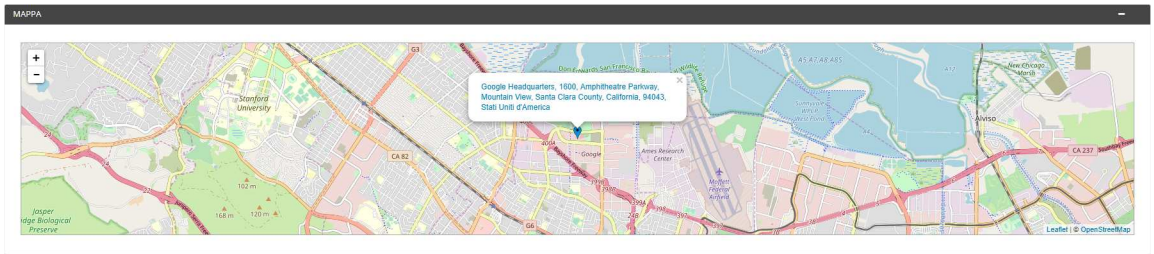


Fig. 4.9: Aspetto della mappa di OpenStreetMap generata tramite Leaflet

Per arricchire la visualizzazione della pagina di dettaglio di un'azienda si è scelto di utilizzare le informazioni relative all'indirizzo per mostrare una mappa interattiva che ne fornisse la posizione.
La realizzazione di questo tipo di dato è basata su *Leaflet*, una libreria JavaScript open-source dalla dimensione contenuta che consente una facile integrazione delle mappe offerte dal servizio *OpenStreetMap* all'interno di qualsiasi progetto. Il dato `Openstreetmap` è stato quindi registrato come `non-db` tramite modifica al template in `modules/DynamicFields/templates/Fields/Template/Openstreetmap.php`, in modo che non sia registrato nel database, ma venga solo utilizzato per visualizzare la mappa nella DetailView (codice 4.4).

```
class TemplateOpenstreetmap extends TemplateField {
    var $data_type = 'openstreetmap';
    var $type = 'openstreetmap';

    function get_field_def() {
        $def = parent::get_field_def();
        $def['source'] = 'non-db';
        return array_merge($def, $this->get_additional_defs());
    }
}
```

Codice 4.4: esempio di template per tipo di dato non-db

Il codice utilizzato per la DetailView del tipo Map è disponibile nell'appendice B.

4.5 Creazione dei moduli

Come illustrato precedentemente, la presenza del costruttore di moduli rende possibile la creazione semplificata dei nuovi moduli che andranno a costituire il CRM. Di seguito vengono elencati i moduli creati al fine di soddisfare le condizioni richieste.

Filiale utilizzato per registrare le filiali di una determinata azienda

Visite registra le visite effettuate da un agente presso un'azienda

Eventi utilizzato per registrare gli eventi di rilevanza commerciale per Rinnai Italia, con possibilità di collegare aziende e contatti partecipanti, oltre che eventuali lead

Corsi utilizzato per registrare i corsi tenuti da Rinnai, divisi per tipo di corso (CAT, ICA, IQU, Online, Commerciale)

Premi registra i premi inviati ad aziende che si sono distinte per vendite o con cui è già avviato un rapporto di fidelizzazione

Categorie Prodotti macro-categoria che raggruppa i tipi di prodotti venduti da Rinnai

Prodotti utilizzato per segnare i modelli, in vendita o fuori produzione, commercializzati da Rinnai

Apparecchi modulo per collegamento successivo del database collaudi, al fine di collegare l'utente finale possessore dell'apparecchio e, se possibile, installatore e rivenditore

Competitori modulo utilizzato per registrare eventuali altri marchi venduti dalle aziende

4.6 Modifiche al modulo Aziende

Il focus principale durante la realizzazione dell'applicativo è stato posto sul modulo destinato a registrare i dati delle aziende, tramite l'inserimento dei campi realizzati con lo strumento Studio ed effettuando modifiche minori al template Smarty. Con le migliorie effettuate è quindi possibile ottenere immediatamente le informazioni fondamentali di un'azienda (fig. 4.10), quali la partita IVA e l'indirizzo principale, oltre che i possibili collegamenti con un'altra azienda se questa fa parte di un gruppo di società o l'appartenenza ad un'agenzia di commercio.

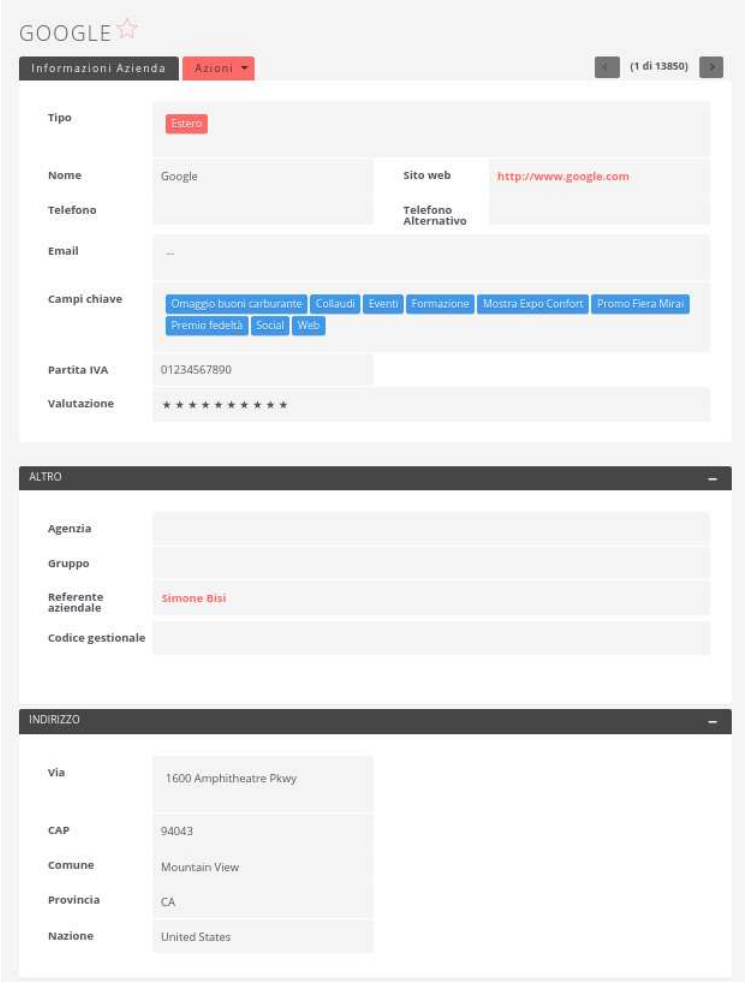


Fig. 4.10: Esempio di azienda visualizzata sul CRM

4.7 Conversione di fogli di calcolo

Per ragioni di praticità, il reparto commerciale si è spesso trovato nella situazione di memorizzare dati su fogli di calcolo, utilizzando la suite Microsoft Office. In questo caso, le richieste degli installatori di essere inseriti nella categoria "Installatori IQU", compilate tramite form online e pervenute tramite e-mail, venivano successivamente trascritte in un file singolo. La richiesta prevedeva la conversione dei dati dal formato XLS (eXceL Spreadsheet) a veri e propri record sul CRM.
Il linguaggio scelto per la conversione è stato Python, data la facilità di prototipazione di uno script e la vasta gamma di librerie disponibili. È stato innanzitutto convertito il file XLS in CSV (Comma Separated Values), il quale rappresenta un formato di file basato su testo, in cui i dati vengono delimitati da un separatore di colonna, spesso rappresentato da una virgola, da cui il nome del formato, e da uno di riga, normalmente rappresentato dal carattere "a capo", ovvero CRLF nei sistemi Windows e solo LF nei sistemi Unix-like. Python dispone nativamente dalla versione 2.3 di una libreria *csv* dalla quale si ottengono procedure per lettura e scrittura su questi file.
Lo script Python così realizzato viene parzialmente mostrato nell'appendice C.

4.8 Importazione dallo strumento dei collaudi

Rinnai dispone di un applicativo utilizzato internamente e dai CAT per la registrazione dei collaudi degli apparecchi. Per praticità, il database di questo applicativo è disponibile sullo stesso server del CRM, in modo da condividerne i dati delle aziende. È stato quindi possibile realizzare una serie di procedure che permettono di interfacciare il modulo *Apparecchi* con lo strumento dei collaudi, ottenendo in automatico una lista di macchine installate nel sotto-pannello dedicato nella scheda di ogni installatore.
Le procedure utilizzate, programmate per essere eseguite quotidianamente sul server tramite *event scheduler*, sono mostrate nell'appendice D.

4.9 Modifiche al modulo di ricerca

La ricerca globale (*Unified Search*) è un campo di ricerca alternativo, rappresentato da un'area normalmente posizionata nell'angolo superiore destro del tema (fig. 4.11) che permette agli utenti di cercare un dato in tutto il database, piuttosto che su un unico modulo. La maggior parte delle richieste è facilmente realizzabile andando

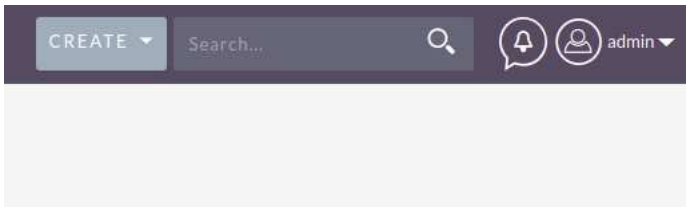


Fig. 4.11: Area di ricerca globale

ad attivare il flag `unified_search` sul campo del modulo desiderato (codice 4.5). Il CRM provvederà poi a selezionare il determinato campo quando si effettua una ricerca.
Si è utilizzato questo metodo per rendere possibile la ricerca via partita IVA, numero di telefono e numero di telefono alternativo.

```
1 $dictionary['<modulo>']['fields']['<campo>']['unified_search'] = true;
```

Codice 4.5: stringa per l'attivazione della ricerca globale sul campo di un modulo

Un richiesta del reparto commerciale prevedeva la semplificazione della ricerca del tipo di società. Ad esempio, se un'azienda venisse memorizzata come "Westin S.P.A." la ricerca globale non riuscirebbe a trovarla utilizzando la stringa "Westin SPA". Si è quindi reso necessario andare a modificare il comportamento della ricerca in base alla ricezione dei diversi input: si analizza il testo ricevuto in input dalla barra di ricerca e si verifica la presenza dell'abbreviazione, sia in stile puntato che non, e se viene rilevata una delle due forme verrà inserita l'altra (codice 4.6). La ricerca globale viene gestita dai file presenti nella cartella `modules/Home`. Le modifiche riportate sono state effettuate sul file `modules/Home/UnifiedSearchAdvanced.php`.

```
$name_where = strtolower($where_clauses[0]);

if(substr_count($name_where, "accounts.name"))
{
        $ragsoc = array(
                0 => 's.a.s.',
                1 => 's.n.c.',
                2 => 's.r.l.',
                3 => 's.r.l.s.',
                4 => 's.p.a.',
                5 => 's.a.p.a.',
        );
        $undotted_string = strtolower(str_replace('.', '', $name_where));
        foreach ($ragsoc as $rs)
        {
                $undotted_rs = strtolower(str_replace('.', '', $rs));
                // found 'business name with dots', add a clause for 'business name without dots'
                if(substr_count($name_where, $rs))
                {
                        $where_clauses[] = str_replace($rs, $undotted_rs, $name_where);
                        break;
                }
                // found 'business name without dots', add a clause for the 'business name with dots'
                else if(substr_count($undotted_string, $undotted_rs))
                {
                        $where_clauses[] = str_replace($undotted_rs, $rs, $name_where);
                        break;
                }
        }
}
```

Codice 4.6: codice per la ricerca semplificata delle ragioni sociali

Capitolo 5

Distribuzione

Il termine inglese *deployment* rappresenta la distribuzione del software al cliente. In questa fase del lavoro, il prodotto verrà utilizzato quotidianamente dal pubblico e sarà premura dello sviluppatore andare a correggere eventuali problematiche non individuate nella fase di test e fornire supporto per istruire l'utilizzatore su nuove funzionalità del software.

5.1 Revisione

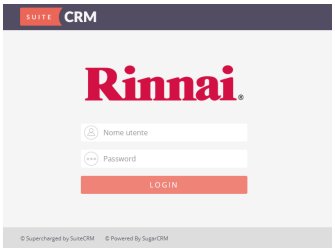


Fig. 5.1: Schermata di login di *SuiteCRM* 7.11 con logo Rinnai

Al termine della fase di sviluppo è stata effettuata una revisione generale del codice, sotto supervisione dell'IT manager, nella quale si è verificata l'effettiva ottimizzazione delle soluzioni apportate, sono state rimosse parti dedicate al debugging dell'applicativo e sono stati aggiunti commenti esplicativi delle varie funzionalità,

al fine di semplificare eventuali operazioni di manutenzione future. È stato inoltre messo alla prova il design *responsive* del nuovo tema *SuiteP* (fig. 5.1) e verificato che le modifiche apportate non avessero alterato il suo comportamento. Come ultimo passaggio prima del passaggio dal server testing a quello in ambiente lavorativo sono state disattivate la modalità sviluppatore, allo scopo di ottenere la generazione automatica della cache ed il relativo aumento delle prestazioni, e ridotto il grado di logging da *debug* a *fatal*, in modo da non ottenere file di log di grandi dimensioni.

```
1 $sugar_config['logger']['level'] = 'fatal';
```

5.2 Installazione parallela

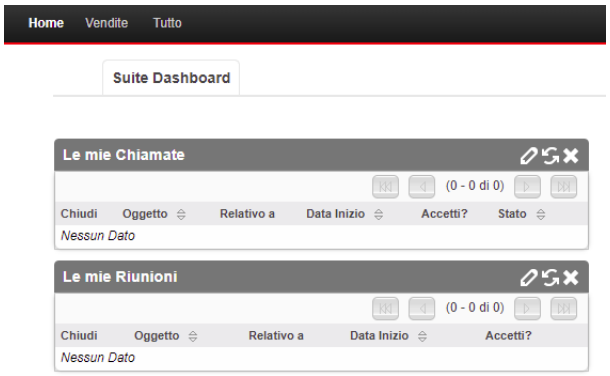


Fig. 5.2: Aspetto della vecchia installazione di *SuiteCRM*

Durante le prime fasi dell'utilizzo, è possibile visualizzare l'applicativo precedentemente installato (fig. 5.2) ad un indirizzo alternativo, in modo da ridurre il downtime in caso di malfunzionamento del sistema. La presenza di un errore non identificato precedentemente nel codice PHP causerebbe infatti il mancato caricamento della totalità della pagina, impedendo l'accesso ad un eventuale dato richiesto. Innanzitutto è stata spostata la cartella di installazione del CRM tramite un normale comando Unix:

```
mv suitecrm suitecrm-old
```

È stato poi importato il backup del database con un nome differente in modo di permettere alla vecchia installazione di puntare ad esso:

```
mysqldump -u root -p qsuitecrm --single-transaction --quick --lock-tables=false > qsuitecrm.sql
mysql -u root -p qsuitecrm-old < qsuitecrm.sql
```

Data la natura datata della versione di partenza di SuiteCRM, è stato necessario configurare il server Apache in modo che le richieste effettuate sulla cartella scelta venissero redirette a PHP 5.4.

```
ProxyPassMatch ^/suitecrm-old/(.*.php)$ fcgi://127.0.0.1:9000/var/www/html/
```

Tramite una modifica al file di configurazione ed una pulizia della cache sarà quindi disponibile in caso di necessità al nuovo indirizzo.

5.3 Utilizzo da parte degli utenti finali

Come precedentemente specificato, non è possibile intendere il CRM come una pratica limitata all'installazione di un nuovo software, bensì consiste nell'implementazione di una *strategia aziendale*, nella quale ogni reparto dovrà collaborare per massimizzare il risultato finale.

- La divisione marketing avrà premura di registrare le iniziative promozionali sostenute e gli eventi di notevole interesse.
- Gli agenti del reparto commerciale dovranno inserire ed aggiornare le anagrafiche presenti in seguito ai colloqui con la clientela e registrare le visite effettuate, in modo da ottenere uno storico dettagliato dell'interazione dell'azienda con Rinnai Italia.
- Il reparto post-vendita potrà trarne vantaggio, utilizzandolo per avere uno storico della relazione dell'azienda di un installatore, centro assistenza o utente finale.

Sarà inoltre necessario ottenere un'ottima comunicazione tra i vari reparti coinvolti, al fine di coordinare al meglio la gestione delle informazioni raccolte. È infine possibile redarre un insieme di linee guida da distribuire al personale per garantire la massima comprensione di ciò che dovrebbe risultare da un regolare utilizzo dell'applicativo, come una corretta registrazione dei valori nei determinati campi e delle relazioni tra i moduli.

Conclusione

Il progetto di ammodernamento del software CRM in funzione presso Rinnai Italia permette all'azienda di poter utilizzare uno strumento moderno, più sicuro ed affidabile per la fidelizzazione del cliente e la generazione di uno storico dei contatti, fondamentale per il reparto vendite interno all'azienda ed i vari agenti commerciali di tutta la penisola.
Lo sviluppo del software è stato eseguito in un ambiente giovane e stimolante, sotto costante supervisione dell'IT manager, ed è risultato in un'esperienza molto positiva e dal forte impatto per l'accrescimento delle proprie conoscenze personali.
SuiteCRM risulta attualmente un perfetto esempio di come l'Open Source, anche in ambito aziendale, possa risultare la soluzione più efficace ad una vasta gamma di problemi, senza sfigurare davanti ad opzioni più costose e chiuse, che porterebbero inevitabilmente alla creazione di vincoli verso le aziende sviluppatrici e installatrici del software, oltre che alla sottoscrizione di esosi canoni annuali, vincolati al numero di utenti.
Per la sua stessa natura, il caso analizzato risulta ovviamente un problema che potrà evolvere nel tempo e mutare in base alle esigenze del reparto aziendale. Possibili sviluppi futuri individuati durante l'attività di tirocinio sono elencati di seguito.

Integrazione PBX Il centralino telefonico PBX (*Private Branch eXchange*) attualmente in uso in azienda, permette l'integrazione con il software CRM, in modo da collegare automaticamente le telefonate al relativo storico del cliente. Questa soluzione era prevista nel progetto iniziale, ma non è stata realizzata data la necessità di aggiornare il server ed il software Elastix presente in azienda

ad una versione più recente.

Collegamento OTRS Il sistema di ticketing adottato dall'azienda (OTRS, *Open-source Ticket Request System*) viene utilizzato internamente per l'instradamento delle richieste di centralino e numero verde verso il reparto tecnico, oltre che essere stato adeguato anche per la gestione interna di ticket rivolti al reparto informatico, di facile consultazione da parte dell'IT Manager. Essendo anch'esso un applicativo open source realizzato in Perl, la creazione di script per la lettura del suo database risulta di facile realizzazione. Una possibile integrazione con OTRS permetterebbe di visualizzare nella scheda di un determinato installatore o utente finale tutti i ticket a lui correlati.

Bibliografia e Sitografia

[1] Rinnai Malaysia — How It All Started. *https://www.rinnai.com.my/history/*.

[2] Rinnai — Company Profile. *https://www.mcexpocomfort.it/__novadocuments/399188?v=636428143051630000*.

[3] Rinnai - Wikipedia. *https://it.wikipedia.org/wiki/Rinnai*.

[4] MITRE Corporation, Serkan Ozkan. PHP 5.4.0: Security vulnerabilities. *https://www.cvedetails.com/vulnerability-list.php-?vendor_id=74&product_id=128&version_id=38980*, 2012-2019.

[5] Karma(In)Security, Egidio Romano. *Tales of SugarCRM Security Horrors*. *http://karmainsecurity.com/tales-of-sugarcrm-security-horrors*, Aprile 2017.

[6] Gartner, Inc. CRM - Customer Relationship Management - Gartner IT Glossary. *https://www.gartner.com/it-glossary/customer-relationship-management-crm*.

[7] M. Duse. *Il CRM strategico. Come migliorare la competitività aziendale fidelizzando e centralizzando il cliente*. F. Angeli, second edition, 2011.

[8] G. Motta. I sistemi informativi integrati ERP. *Mondo Digitale*, 1(1), 2002.

[9] SugarCRM Funding Rounds — Crunchbase. *https://www.crunchbase.com/organization/sugarcrm#section-funding-rounds*.

[10] SugarCRM Secures $40 Million in Funding From Goldman Sachs & Co. *https://www.businesswire.com/news/home/20130821005207/en/SugarCRM-Secures-40-Million-Funding-Goldman-Sachs*, 2013.

[11] Salesforce, Inc Revenue 2006-2018 — CRM — MacroTrends. *https://www.macrotrends.net/stocks/charts/CRM/salesforce,-inc/revenue*.

[12] SuiteCRM - Open source CRM for the world. *https://github.com/salesagility/SuiteCRM*.

[13] SuiteCRM Documentation Site. *https://docs.suitecrm.com*.

[14] Angel Magaña, Michael Whitehead. *Implementing SugarCRM 5.x*. Packt Publishing, first edition, 2010.

[15] SuiteCRM Compatibility Matrix. *https://docs.suitecrm.com/admin/compatibility-matrix/*.

[16] Selectize.js. *https://selectize.github.io/selectize.js/*.

[17] AJAX - The XMLHttpRequest Object. *https://www.w3schools.com/xml/ajax_xmlhttprequest_create.asp*.

Lista delle figure

Appendice A

Codice relativo alla verifica di una partita IVA

```
<?php

    $piva = $_REQUEST["piva"];

    $url = "http://ec.europa.eu/taxation_customs/vies/vatResponse.html?";

    $complete = $url.
                    "memberStateCode=IT".
                    "&number=".$piva.
                    "&requesterMemberStateCode=IT".
                    "&requesterNumber=02774510362".
                    "&action=check";

    $html = file_get_contents($complete);

    $dom = new DomDocument();
    $dom->loadHTML($html);

    foreach($dom->getElementsByTagName('fieldset') as $element)
    {
        $fieldset = $element->nodeValue.'<br>';
        break;
    }

    $exploded = array_filter( explode("\n", $fieldset) );
    $exploded = array_map('trim', $exploded);

    $validity = $exploded[1][0];
    if($validity == 'Y' || $validity == 'T')
    {
        //Nome
        $name = $exploded[12];
        //Via numciv citta
        $place = $exploded[16];

        //First number occurrence = 'numciv cap citta'
```

```
37          preg_match('/^\D*(?=\d)/', $place, $m);
38              $pos = isset($m[0]) ? strlen($m[0]) : false;
39
40          //Via senza numciv
41          $street = substr($place, 0, $pos-1);
42
43          //Numciv cap citta
44          $tmp = substr($place, $pos);
45          $array = explode(' ', $tmp, 3);
46
47          //Via + numciv
48          $street .= " ".$array[0];
49
50          //Cap
51          $cap = $array[1];
52
53          //Citta + prov
54          $size = strlen($array[2]);
55          $idx = strpos($array[2], ' ');
56
57          $idx = strpos(strrev($array[2]), " ");
58          $idx = strlen($array[2]) - 1 - $idx;
59
60          $place = ucwords(strtolower(substr($array[2], 0, $idx+1)));
61          $prov = substr($array[2], $idx+1, $size);
62
63          if($name === '---')
64              echo "0||Raggiunto limite massimo richieste, riprovare piu tardi.";
65          else
66              echo "1||".$name."||".$street."||".$cap."||".$place." ".$prov;
67      }
68      else
69      {
70          echo '0||Partita IVA non trovata';
71      }
72  ?>
```

Codice A.1: file check.php

```
1   {if strlen({{sugarvar key='value' string=true}}) <= 0}
2       {assign var="value" value={{sugarvar key='default_value' string=true}} }
3   {else}
4       {assign var="value" value={{sugarvar key='value' string=true}} }
5   {/if}
6
7   <input type='text' name='{{sugarvar key='name'}}'
8       id='{{sugarvar key='name'}}' size='{{$displayParams.size|default:30}}'
9       value='{$value}' title='{{$vardef.help}}'
10      {{if !empty($displayParams.accesskey)}} accesskey='{{$displayParams.accesskey}}' {{/if}} {{$displayParams.field}}
11      onkeyup="showHint(this.value)">
12
13  <p>
14      <span id="txtHint"></span>
15
16      <p><input type="hidden" value="" id="_piva_name" readonly><button style="display: none;"
17          type=button id="nameButton" onclick="copyToClipboard('name')">Copia</button></p>
18      <p><input type="hidden" value="" id="_piva_street" readonly><button style="display: none;"
19          type=button id="streetButton" onclick="copyToClipboard('street')">Copia</button></p>
20      <p><input type="hidden" value="" id="_piva_cap" readonly><button style="display: none;"
```

```
21          type=button id="capButton" onclick="copyToClipboard('cap')">Copia</button></p>
22      <p><input type="hidden" value="" id="_piva_place" readonly><button style="display: none;"
23          type=button id="placeButton" onclick="copyToClipboard('place')">Copia</button></p>
24  </p>
25
26  {literal}
27  <script>
28      function showHint(str)
29      {
30          if (str.length < 11)
31          {
32              //Hidden buttons, hidden inputs
33              setButtonStyle("display: none;");
34              setInputType("hidden");
35              document.getElementById("txtHint").innerHTML = "";
36              return;
37          }
38          else
39          {
40              var xmlhttp = new XMLHttpRequest();
41              xmlhttp.onreadystatechange = function()
42              {
43                  if (this.readyState == 4 && this.status == 200)
44                  {
45                      //Split the response with the delimiter
46                      var responseArray = this.responseText.split("||");
47                      if(responseArray[0] == '1')
48                      {
49                          //Visible buttons, visible inputs
50                          setButtonStyle("");
51                          setInputType("text");
52
53                          document.getElementById("txtHint").innerHTML = "Partita IVA verificata";
54
55                          document.getElementById("_piva_name").value = toTitleCase(responseArray[1]);
56                          document.getElementById("_piva_street").value = toTitleCase(responseArray[2]);
57                          document.getElementById("_piva_cap").value = responseArray[3];
58                          document.getElementById("_piva_place").value = responseArray[4];
59                      }
60                      else
61                      {
62                          //Hidden buttons, hidden inputs
63                          setButtonStyle("display: none;");
64                                      setInputType("hidden");
65
66                          document.getElementById("txtHint").innerHTML = responseArray[1];
67                      }
68                  }
69              };
70              xmlhttp.open("GET", "include/SugarFields/Fields/Partitaiva/check.php?piva=" + str, true);
71              xmlhttp.send();
72          }
73      }
74
75      function setInputType(type)
76      {
77          document.getElementById("_piva_name").type = type;
78          document.getElementById("_piva_street").type = type;
79          document.getElementById("_piva_cap").type = type;
80          document.getElementById("_piva_place").type = type;
81      }
82
```

```
83      function setButtonStyle(style)
84      {
85          document.getElementById("nameButton").style = style;
86          document.getElementById("streetButton").style = style;
87          document.getElementById("capButton").style = style;
88          document.getElementById("placeButton").style = style;
89      }
90
91      function copyToClipboard(id)
92      {
93          var copyText = document.getElementById("_piva_"+id);
94          copyText.select();
95          document.execCommand("copy");
96
97          document.getElementById("nameButton").innerHTML = "Copia";
98          document.getElementById("streetButton").innerHTML = "Copia";
99          document.getElementById("capButton").innerHTML = "Copia";
100         document.getElementById("placeButton").innerHTML = "Copia";
101
102         document.getElementById(id+"Button").innerHTML = "Copiato";
103     }
104
105     function toTitleCase(str)
106     {
107         return str.replace(
108         /\w\S*/g,
109             function(txt)
110             {
111                 return txt.charAt(0).toUpperCase() + txt.substr(1).toLowerCase();
112             }
113         );
114     }
115 </script>
116 {/literal}
```

Codice A.2: file EditView.tpl del tipo di dato Partitaiva

Appendice B

Codice per la visualizzazione di OpenStreetMap tramite Leaflet

```
<div id="mapcontainer">
    <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable=no" />
    <link rel="stylesheet" type="text/css" href="http://cdn.leafletjs.com/leaflet/v0.7.7/leaflet.css" />
    <script type="text/javascript" src="http://cdn.leafletjs.com/leaflet/v0.7.7/leaflet.js"></script>

    <div id="map" style="width: auto; height: 300px; border: 1px solid #AAA;"></div>

    <script>
        {literal}
        var map = L.map( "map", {
                center: [44.78, 10.88],
                minZoom: 2, zoom: 2
            });
        {/literal}

        string = "{$fields.billing_address_street.value}";

        var
            cap_len = "{$fields.billing_address_postalcode.value}".length,
            city_len = "{$fields.billing_address_city.value}".length;

        if(cap_len > 1)
            string = string + " {$fields.billing_address_postalcode.value}";
        else if(city_len > 1)
            string = string + " {$fields.billing_address_city.value}";
        else
            string = string + " {$fields.billing_address_country.value}";
        map_fetcher(string);

        {literal}
            function map_fetcher(string)
            {
                var markers;
                string = string.replace(new RegExp(" ", "g"), "+");
                url = "https://nominatim.openstreetmap.org/search.php?q="+string+"&format=json";
                fetch(url)
```

```
37                  .then((resp) => resp.json())
38                  .then(function(data) {
39                      if(data[0])
40                      {
41                          markers = data[0];
42                          L.tileLayer( "http://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png", {
43                          attribution: "&copy; <a href='https://www.openstreetmap.org/copyright'>OpenStreetMap</a>",
44                          subdomains: ["a", "b", "c"]
45                          }).addTo( map );
46                          var myIcon = L.icon({
47                              iconUrl: "maps/images/pin24.png",
48                              iconRetinaUrl: "maps/images/pin48.png",
49                              iconSize: [29, 24],
50                              iconAnchor: [9, 21],
51                              popupAnchor: [0, -14]
52                          });
53                          L.marker( [markers.lat, markers.lon], {icon: myIcon} )
54                              .bindPopup( "<a href='#' target='_blank'>" + markers.display_name + "</a>" )
55                              .addTo( map ).
56                              openPopup();
57                          map.setZoom( 15 );
58                          map.setView( [markers.lat, markers.lon] );
59                      }
60                      else
61                      {
62                          var node  = document.getElementById("map");
63                          map.off();
64                          map.remove();
65                          node.parentNode.removeChild(node);
66                          document.getElementById("mapcontainer").innerHTML = "Impossibile visualizzare l'indirizzo.";
67                      }
68                  })
69                  .catch(function() { });
70              }
71          {/literal}
72      </script>
73  </div>
```

Codice B.1: DetailView per il tipo di dato Openstreetmap

Appendice C

Codice parziale per la conversione del file CSV

```
import csv
import mysql.connector as mariadb
import uuid
import MySQLdb

with open('iqu.csv', 'r') as csvfile:

    # Reader CSV
    csv_reader = csv.DictReader (csvfile, delimiter=';', quotechar='|')

    # Connection
    mariadb_connection = mariadb.connect(host, user, password, database='suitecrm', buffered = True)
    cursor = mariadb_connection.cursor(dictionary=True)

    # Elenco partite iva
    cursor.execute('SELECT account_piva_c FROM accounts INNER JOIN accounts_cstm ON id = id_c')
    pive = cursor.fetchall()
    pive_array = {}
    for pi in pive:
        pive_array[pi['account_piva_c']] = '1'

    # Parsing
    for row in csv_reader:

        name = row['Ragione Sociale'].title()

        cod_gest = row['Codice Cliente']

        address_street = row['Indirizzo'].title()
        address_number = row['Nr. Civico'].title()
        address_cap = row['CAP'].title()
        address_prov = row['Provincia'].title()
        address_city = row['Citta'].title()

        address_street_alt = row['Indirizzo 2'].title()
        address_number_alt = row['Nr. Civico 2'].title()
```

```
37          address_cap_alt = row['CAP 2'].title()
38          address_prov_alt = row['Provincia 2'].title()
39          address_city_alt = row['Citta 2'].title()
40
41          piva = row['P. IVA'].rjust(11, '0')
42          cf = row['C.F.']
43
44          phone = row['Telefono 1']
45          phone_alt = row['Telefono 2']
46
47          email = row['Email']
48          email_pec = row['PEC']
49          website = row['Sito']
50
51          account_uuid = uuid.uuid4()
52          office_uuid_alt = uuid.uuid4()
53
54          ##########################################################
55
56          # Presenza nel database
57          if piva not in pive_array:
58
59              # Non presente: insert
60              insert_account(account_uuid, name, \
61                             address_street, address_city, address_prov, address_cap, \
62                             phone, phone_alt, \
63                             website, agency_uuid, cf, piva, cod_gest, \
64                             address_street, address_number, address_city, address_prov, address_cap, \
65                             phone, phone_alt, website, account_uuid)
66
67              # Se con doppia sede
68              if(address_street_alt):
69                  insert_office(office_uuid_alt, name+" (alternativo)", \
70                                address_street_alt, address_number_alt, address_city_alt, address_prov_alt,
    address_cap_alt, \
71                                phone, phone_alt, website, account_uuid)
72          else:
73              # Presente: update su quella gia esistente
74
75              # Seleziona id dell'azienda (accounts)
76              update_query = f"SELECT id_c FROM accounts_cstm WHERE account_piva_c = '{piva}' LIMIT 1"
77              cursor.execute(update_query)
78
79              account_uuid = cursor.fetchall()
80              for a in account_uuid:
81                  account_uuid = f"{a['id_c']}"
82
83              # Aggiorna azienda (accounts)
84              update_account(account_uuid, name, \
85                             address_street, address_city, address_prov, address_cap, \
86                             phone, phone_alt, \
87                             website, agency_uuid, cf, piva, cod_gest, \
88                             address_street, address_number, address_city, address_prov, address_cap,
89                             phone, phone_alt, website, account_uuid)
90
91              if(address_street_alt):
92                  insert_office(office_uuid_alt, name+" (alternativo)", \
93                                address_street_alt, address_number_alt, address_city_alt, \
94                                address_prov_alt, address_cap_alt, \
95                                phone, phone_alt, website, account_uuid)
96
97          ##########################################################
```

```
98
99          # Email
100         if(len(email_pec)):
101             email_uuid = uuid.uuid4()
102             insert_email(email_uuid, email.lower(), email.upper(), 0, account_uuid)
103         # Email PEC
104         if(len(email_pec)):
105             email_pec_uuid = uuid.uuid4()
106             insert_email(email_pec_uuid, email_pec.lower(), email_pec.upper(), 1, account_uuid)
107
108         ##########################################################
109
110     mariadb_connection.commit()
111     mariadb_connection.close()
```

Codice C.1: esempio di script Python utilizzato per la conversione degli installatori IQU

Appendice D

Procedure MySQL utilizzate per la conversione dei collaudi

```
DELIMITER $$
DROP PROCEDURE IF EXISTS sb_modello_to_aos_products$$
CREATE PROCEDURE sb_modello_to_aos_products()
BEGIN
    # Creazione prodotti
    # sb_modello to aos_products
    REPLACE INTO qsuitecrm.aos_products (id, name, part_number, aos_product_category_id)
    SELECT id_modello, descrizione, codice,
    (CASE
        WHEN rinnai_sb.sb_modello.prodotto LIKE 'Caldaia'
            THEN (SELECT id FROM qsuitecrm.aos_product_categories WHERE name LIKE 'Caldai%')
        WHEN rinnai_sb.sb_modello.prodotto LIKE 'Scaldabagno'
            THEN (SELECT id FROM qsuitecrm.aos_product_categories WHERE name LIKE 'Scalda%')
        WHEN rinnai_sb.sb_modello.prodotto LIKE 'Altro'
            THEN (SELECT id FROM qsuitecrm.aos_product_categories WHERE name LIKE 'Asciuga%')
        ELSE '-1'
    END) as category
    FROM rinnai_sb.sb_modello
    WHERE deleteUser IS NULL;

    # Parte custom della tabella prodotti
    #sb_modello to aos_products_cstm
    REPLACE INTO qsuitecrm.aos_products_cstm (id_c, subcategory_c)
    SELECT
    id_modello,
    (CASE
        WHEN rinnai_sb.sb_modello.gamma LIKE 'Domestico'
            THEN '1'
        WHEN rinnai_sb.sb_modello.gamma LIKE 'Professionale'
            THEN '2'
        ELSE '-1'
    END) AS gamma
    FROM rinnai_sb.sb_modello
    WHERE deleteUser IS NULL;
END $$

```

```
37 DROP PROCEDURE IF EXISTS sb_collaudo_to_rinad_machinery$$
38 CREATE PROCEDURE sb_collaudo_to_rinad_machinery()
39 BEGIN
40     # Creazione apparecchi
41     # sb_modello to rinad_machinery
42     TRUNCATE TABLE rinad_machinery;
43     REPLACE INTO qsuitecrm.rinad_machinery (id, name, description)
44     SELECT id_collaudo, UPPER(numero_serie), note
45     FROM rinnai_sb.sb_collaudo
46     GROUP BY numero_serie
47     WHERE deleteUser IS NULL
48     AND id_installatore IS NOT NULL;
49
50     # Collegamento apparecchio - prodotto
51     # sb_collaudo to rinad_machinery_aos_products_c
52     TRUNCATE TABLE rinad_machinery_aos_products_c;
53     REPLACE INTO qsuitecrm.rinad_machinery_aos_products_c (id, rinad_machinery_aos_productsaos_products_ida,
         rinad_machinery_aos_productsrinad_machinery_idb)
54     SELECT UUID(), id_modello, id_collaudo
55     FROM rinnai_sb.sb_collaudo
56     GROUP BY numero_serie
57     WHERE deleteUser IS NULL
58     AND id_installatore IS NOT NULL;
59
60     # Collegamento apparecchio - installatore
61     # sb_collaudo to rinad_machinery_accounts_c
62     TRUNCATE TABLE rinad_machinery_accounts_c;
63     REPLACE INTO qsuitecrm.rinad_machinery_accounts_c (id, rinad_machinery_accountsaccounts_ida,
         rinad_machinery_accountsrinad_machinery_idb)
64     SELECT UUID(), id_installatore, id_collaudo
65     FROM rinnai_sb.sb_collaudo
66     GROUP BY numero_serie
67     WHERE deleteUser IS NULL
68     AND id_installatore IS NOT NULL;
69
70     # Creazione utilizzatori
71     TRUNCATE TABLE qsuitecrm.rinut_utilizer;
72     INSERT INTO qsuitecrm.rinut_utilizer(id, name, surname, genid, address, city, state, phone, supername, code, mail,
         cap)
73     SELECT
74         UUID(),
75         TCASE(nome),
76         TCASE(cognome),
77
78         UPPER(
79             CASE
80                 WHEN codice_fiscale IS NULL OR codice_fiscale = '' THEN
81                     CONCAT(SUBSTRING(cognome, 1, 3), SUBSTRING(nome, 1, 3), cap, telefono)
82                 ELSE
83                     codice_fiscale
84                 END
85         ) AS genid,
86         TCASE(CONCAT(via, ' ', nr_civico)) AS indirizzo,
87         TCASE(comune),
88         UPPER(provincia),
89         telefono,
90         TCASE(CONCAT(
91             (CASE
92                 WHEN nome LIKE 'ANONIMO' THEN ''
93                 WHEN nome LIKE '-' THEN ''
94                 ELSE nome END),
95             ' ',
```

```
96              (CASE
97                  WHEN cognome LIKE 'ANONIMO' THEN ''
98                  WHEN cognome LIKE '-' THEN ''
99                  WHEN cognome LIKE nome THEN ''
100                 ELSE cognome END)))
101         AS nominativo,
102         codice_fiscale,
103         mail,
104         cap
105     FROM rinnai_sb.sb_collaudo
106     WHERE id_installatore IS NOT NULL
107     GROUP BY genid
108     HAVING nominativo <> '';
109
110     # Collegamento apparecchio - utilizzatore
111     # sb_collaudo to rinut_utilizer_rinad_machinery_c
112     TRUNCATE TABLE qsuitecrm.rinut_utilizer_rinad_machinery_c;
113     INSERT INTO qsuitecrm.rinut_utilizer_rinad_machinery_c (id, rinut_utilizer_rinad_machineryrinut_utilizer_ida,
         rinut_utilizer_rinad_machineryrinad_machinery_idb)
114         (SELECT
115             UUID(),
116             id,
117             id_collaudo
118         FROM
119             ( SELECT
120                 id_collaudo,
121                 UPPER(CASE WHEN codice_fiscale IS NULL OR codice_fiscale = '' THEN CONCAT(SUBSTRING(cognome, 1, 3),
         SUBSTRING(nome, 1, 3), cap, telefono)
122                       ELSE codice_fiscale END) AS genid,
123                 numero_serie
124               FROM
125                 rinnai_sb.sb_collaudo
126               GROUP BY genid
127               HAVING genid <> '') tmp
128             JOIN
129                 qsuitecrm.rinut_utilizer ut
130             ON tmp.genid = ut.genid );
131 END $$
132 DELIMITER ;
```

Codice D.1: procedure MySQL create per l'importazione dal database collaudi

```
1 CREATE EVENT sync_machinery
2     ON SCHEDULE
3         EVERY 1 DAY
4     STARTS (TIMESTAMP(CURRENT_DATE) + INTERVAL 1 DAY + INTERVAL 7 HOUR)
5     DO
6         CALL sb_modello_to_aos_products;
7         CALL sb_collaudo_to_rinad_machinery();
```

Codice D.2: evento MySQL creato per l'esecuzione periodica delle procedure