POLITECNICO DI TORINO

Corso di Laurea Magistrale in
INGEGNERIA INFORMATICA
Anno Accademico 2019/2020

TESI DI LAUREA MAGISTRALE

PROGETTAZIONE E SVILUPPO ERP/CMS AZIENDALE HARDWARE CONNECTED

RELATORE:
Prof. Cabodi Giampiero

CANDIDATO:
Lacerenza Danilo

ANNO ACCADEMICO 2019/2020

RINGRAZIAMENTI

Prima di procedere con la trattazione, vorrei dedicare qualche riga a tutti coloro che mi sono stati vicini in questo percorso di crescita personale e professionale.

Un sentito grazie al mio relatore Prof. Cabodi Giampiero per la sua infinita disponibilità e per aver atteso i miei tempi tecnici lavorativi.

Ricordo chiaramente l'ostinazione con cui ho deciso di iniziare questo percorso di studi, reso ancora più impegnativo dai miei ritmi lavorativi sempre intensi. C'è una persona che mi ha sempre convinto a non mollare, che mi ha dato in ogni istante supporto morale e affettivo e che ha reso questo sogno realtà, mia moglie Claudia, senza di lei non sarei mai potuto arrivare fin qui. Grazie per avermi tenuto per mano in ogni momento, soprattutto in quelli sconforto, per aver creduto in me con pazienza e dedizione.

Ringrazio Riccardo Barbotti e Giulia Fiore, amici veri che mi hanno aiutato a superare l'ultimo ostacolo insormontabile di questo percorso di studi, Tommaso Salvetti, mentore e amico con il quale ho condiviso successi e fallimenti, e i miei colleghi di Bitboss SRL Davide Brienza e Paola Biondo fantastici compagni che con eccelsa competenza hanno contribuito attivamente alla realizzazione e al successo di PrimoUP.

Un infinito grazie va dedicato all'Ing Daniele Cavallero, responsabile del reparto IT di Primo Group SpA, che, insieme all'amministratore delegato Dott. Mirko Puccio, ha fortemente voluto e creduto nel progetto PrimoUP, contribuendo enormemente a livello progettuale e organizzativo, dandomi oltretutto un'indimenticabile opportunità di crescita professionale.

Infine, vorrei dedicare un piccolo grazie alla mia perseveranza.

per il piccolo DAVIDE, da mamma e papà
"Questa parte della mia vita,
questa piccola parte della mia vita
si può chiamare felicità!"

SOMMARIO

PREMESSA

La creazione di un software non si limita alla semplice implementazione di uno specifico algoritmo. L'attività più importante è affidata alla progettazione. L'implementazione algoritmica è solo l'ultimo step di una più complessa attività che parte dall'analisi dei requisiti, dallo studio di fattibilità e si conclude con la traduzione in uno specifico linguaggio di programmazione.

Il progetto trattato in questo elaborato è PrimoUP, un software web, interamente custom, per la gestione di tutti i processi aziendali di un'importante catena di cliniche odontoiatriche. Ho curato personalmente ogni aspetto di questo software, dall'analisi delle problematiche aziendali che mi hanno permesso di progettare il software ottimizzando il lavoro del committente, alla scelta dei tools necessari, fino all'implementazione e ai test.

Il software è stato ideato con la finalità di rendere il lavoro in azienda paperless. L'analisi, la progettazione e l'implementazione sono state affidate alla società Bitboss srl.
Bitboss srl nasce nel 2017 come impresa innovativa ed è ad oggi incubata presso l'I3P del Politecnico di Torino. Oltre ad esserne co-fondatore e socio, sono stato parte integrante del team di sviluppo, dedicando circa un anno di lavoro full-time, ricoprendo i ruoli di project manager e lead developer.

1. INTRODUZIONE

A seconda delle funzionalità offerte da un software, possiamo distinguere quest'ultimo in due macro categorie: ERP o CRM.

Con ERP si intendono quei software che offrono un pieno controllo su tutti i processi interni. L'acronimo "Enterprise resource planning" identifica la pianificazione delle risorse di impresa. Grazie a queste soluzioni è possibile individuare sin da subito l'eventuale presenza di errori o di rallentamenti sia durante la produzione sia in fase di distribuzione.
Potremmo definire ERP un software composto da tanti programmi volti a gestire i più rilevanti processi di business di un'azienda quali, ad esempio, vendite, acquisti, gestione magazzino, finanza, contabilità, ecc...

CRM è invece l'acronimo di "Customer Relationship Management", un software utilizzato per la gestione dei contatti, la gestione delle vendite, la produttività. L'obiettivo di un sistema CRM è la gestione generale dei contatti. Obiettivo è quello di raccogliere e memorizzare i dettagli di contatto e a seguire le attività di vendita.

Un'altra importante tipologia di software è rappresentata dai CMS, ovvero Content Management System. Si identificano tutti quei programmi web il cui compito è facilitare la gestione dei contenuti. La generazione della pagina avverrà mediante una configurazione da parte dell'amministratore. La modifica, l'integrazione di nuove funzionalità e la cancellazione di programmi obsoleti non dovranno più essere affidate alla società che ha sviluppato il software, senza dover modificare il codice sorgente ma potrà essere eseguita utilizzando particolari pagine web usate per generare e modificare i contenuti, chiamate builder.

1.1 Situazione iniziale dell'azienda

Al momento dell'inizio della progettazione di PrimoUP, l'azienda Primo Group SPA, commissionaria del software oggetto di questo elaborato, era in attività già da una decina di anni. Il software in uso era un software

stand-alone, disponibile solo nella versione compatibile con sistemi operativi Windows. E' emersa fin da subito l'indiscutibile necessità di mantenere lo storico delle attività svolte dall'apertura dell'azienda fino al passaggio al nuovo software, delineando quindi l'esigenza non solo di creare un nuovo software ad-hoc in grado di migliorare i tempi delle attività in clinica, ma anche l'organizzazione dei dati in modo tale da renderli compatibili con il sistema in essere.
Questa operazione è stata svolta in due passaggi distinti:

- import, inserimento in un ambiente di test dei dati scaricati dal vecchio software (salvataggio file csv, estrazione mediante accesso al database..) in modo tale da essere facilmente interpretati e analizzati;
- conversione, normalizzazione dei dati così ottenuti nel database utilizzato dal software finale.

Tali interventi ricoprono un lavoro molto delicato a cui prestare molta attenzione, ma è stato chiaro fin da subito che, trattandosi di un'attività già in essere, le tempistiche per la migrazione al nuovo sistema erano molto strette.

1.2 Obiettivi e necessità dell'azienda

L'obiettivo principe che ha indotto il committente al passaggio ad un nuovo software era rendere "paperless" l'attività aziendale - priva, quindi, di qualunque foglio cartaceo - e accompagnare ogni processo lavorativo a supporti informatici: trasformazione di contratti, preventivi, consensi e quant'altro in documenti PDF, eliminazione post-it e note varie ed ottenimento di un unico fulcro in cui inserire le informazioni di pazienti e medici, ottimizzando così i tempi di lavoro e riducendo la duplicazione di dati.
Un'altra importante richiesta era poter integrare diversi hardware con il sistema che si sarebbe creato, come i ortopantomografi (macchinari per eseguire radiografie dentali), i dispositivi di pagamento POS e altro.

Al fine di ottenere un software progettato seguendo tutte le specifiche imposte dal committente, ma comunque compatibile con i vecchi sistemi, graficamente accattivante e moderno ma comunque user-friendly, abbiamo

concordato di non basarci sul vecchio software e progettare ex-novo tutte le funzionalità necessarie al corretto svolgimento dell'attività aziendale.

2. PROGETTAZIONE SOFTWARE

Progettare un software, piccolo o grande che sia, ricopre numerose attività preliminari alla sua concreta traduzione in un particolare linguaggio di programmazione. La scelta algoritmica e la sua implementazione sono solo la parte conclusiva di un più ampio lavoro.
Le fasi cardine necessarie ad una corretta progettazione ed implementazione di un software possono essere racchiuse in 4 macro aree:

1. studio della fattibilità e raccolta delle specifiche;
2. analisi dei requisiti;
3. progettazione del software;
4. realizzazione del software.

Analizzando le varie aree possiamo definire quanto una serie di operazioni da compiere:

1. Per prima cosa, come appena descritto, è stato necessario avviare lo studio di fattibilità.
   La richiesta del committente non era di un semplice porting del suo vecchio software su tecnologie più moderne, ma prevedeva, invece, la progettazione di nuove funzionalità da integrare. Da sottolineare inoltre, in tale fase iniziale, che, in funzione della tecnologia utilizzata, si sarebbe potuta figurare l'impossibilità di sviluppo di un particolare programma utilizzato fino ad allora.
   Possiamo dividere la fattibilità in 2 gruppi: fattibilità tecnica ed economica. Nello studio della prima, abbiamo analizzato gli aspetti tecnici e le procedure da mettere in atto per rendere possibile l'implementazione desiderata del cliente. Per quanto concerne la seconda, non meno importante, essa ha rappresentato la valutazione dei costi e dei benefici che il software finale avrebbe portato al committente.
2. Aspetto importante nella fase di analisi dei requisiti è trascurare i dettagli tecnici, garantendo al cliente una piena comprensione del

problema e della soluzione offerta. Non è infatti necessario valutare immediatamente la quantità e la tipologia dei dati da salvare, o l'interazione che l'utente dovrà compiere per eseguire particolari funzionalità: questi saranno aspetti che dovranno essere trattati in fase successive. D'altro canto, al fine di ottenere un risultato condiviso, è stata indispensabile la creazione di un documento che dettagliasse i tempi e i costi. Partendo dai desiderata del committente (prima fase indispensabile per progettare correttamente un software) di fondamentale importanza è la raccolta delle specifiche fornite dal committente, che serviranno a delineare un modello di applicazione da realizzare.

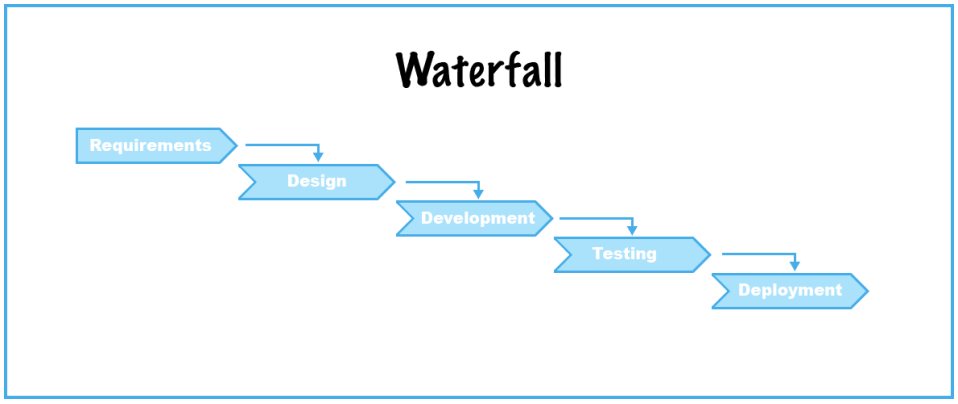
Al fine di una puntuale valutazione che indicasse tempi di sviluppo e conseguente offerta economica, ha ricoperto un ruolo cruciale la raccolta delle specifiche fornite dal committente. In questa fase, spesso, il cliente non ha le idee chiare su ciò che vorrebbe. E' compito di un buon analista riuscire a decodificare i requisiti necessari alla realizzazione delle sue richieste. In questo frangente il mio lavoro è stato tanto lungo quanto meticoloso; nulla, infatti, deve essere lasciato al caso. E' stato infatti necessario incontrarci diverse volte organizzando meeting con più figure aziendali, tra le quali hanno contribuito in modo attivo e concreto i responsabili di settore, CTO e CEO.

3. La progettazione del software, come lo sviluppo, non è avvenuta in un unico spazio temporale. Il progetto PrimoUP non è infatti stato realizzato come software creato e consegnato nella sua completezza al cliente (nel gergo "progetto chiavi in mano"), con un classico approccio di sviluppo a cascata (waterfall), bensì analizzando ciclicamente i desiderata e progettando il software. Questa particolare metodologia, chiamata Agile, ha fatto al caso nostro. La scelta del modello da seguire, la progettazione e lo sviluppo sono avvenuti, come detto, avvalendoci di incontri bisettimanali. La progettazione del software rappresenta il core del lavoro nel suo complesso, merita di essere dettagliata nello specifico nel corso dei seguenti paragrafi.
4. Lo sviluppo si è quindi tradotto nello step immediatamente successivo alla sua progettazione. In prima fase mediante la creazione di

wireframe. Un wireframe è una guida visiva che rappresenta la struttura scheletrica di un sito Web, creato con lo scopo di disporre gli elementi per realizzare al meglio uno scopo particolare. Sono schizzi che danno l'idea di come gli elementi grafici possano essere implementati. La fase successiva è stata tradurre i wireframe in mockup, ovvero la trasformazione di un disegno stilizzato in una foto di come gli elementi grafici saranno effettivamente implementati, con le delle corrette colorazioni, forme ed elementi grafici che saranno utilizzati. Infine, i mockup così definiti sono stati tradotti in codice di programmazione e produzione delle pagine web finali.

2.1 Agile e Scrum

Il classico processo di sviluppo di un software può essere definito come processo Waterfall (a cascata), attraverso il quale si suddivide il lavoro in fasi consecutive: analisi dei requisiti con raccolta delle specifiche, progettazione del software analizzando ogni aspetto tecnico, sviluppo attraverso la programmazione e la fase di test che si conclude con l'approvazione del cliente, il lancio del prodotto e la chiusura del progetto.



Quando le specifiche non sono note a priori, quando le funzionalità richieste variano in funzione dei risultati prodotti da altre, questo sistema non è utilizzabile. Come nel caso di specie, questo tipo di approccio non era percorribile a causa della mancanza di consapevolezza di ciò che il cliente avrebbe veramente desiderato e di quale fosse la UX (user experience) migliore per gli utilizzatori in funzione dei processi aziendali richiesti.

Abbiamo pertanto optato per la gestione di una progettazione dinamica, avvalendoci di un particolare framework di sviluppo chiamato Scrum.
Scrum è parte di una più ampia collezione di metodologie di lavoro denominata Agile. Tutti gli aspetti del lavoro devono essere visibili ai responsabili, garantendone piena trasparenza. Per far ciò si procede frequentemente all'ispezione del prodotto durante il suo sviluppo; così facendo il percorso di sviluppo potrà essere adattato alle nuove esigenze, dovute a cambi di condizione del mercato, nuovi obiettivi aziendali etc.
I suoi creatori, Ken Schwaber e Jeff Sutherland, si sono ispirati al termine usato nel gioco del rugby, in cui, effettivamente, il termine Scrum indica letteralmente mischia, situazione di gioco in cui i giocatori della squadra compongono un insieme compatto che spinge nella stessa direzione: tutto il team lavora insieme, in maniera sinergica con il cliente.

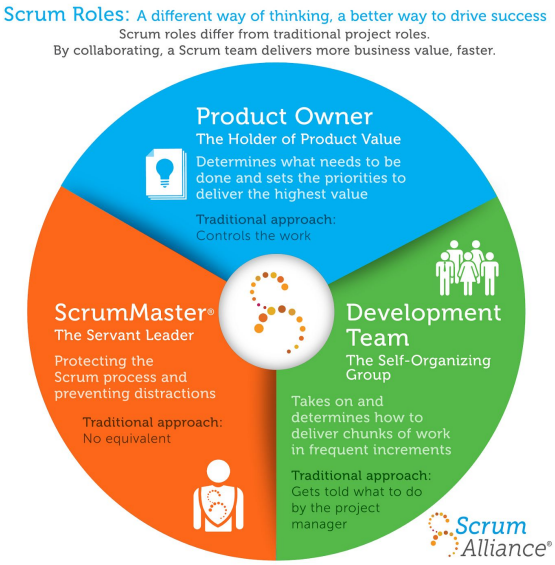
2.1.1 Sprint

Come tutte le metodologie Agile, Scrum si basa sulla divisione del progetto in più fasi, chiamate Sprint.
Lo Sprint è il cuore del framework Scrum. Della durata compresa tra 2 e 4 settimane, ogni Sprint è nella pratica un mini progetto, il cui scopo è presentare una particolare parte del software che risponda a determinati requisiti, utilizzabile e potenzialmente rilasciabile.
Lo sprint inizia con un meeting che coinvolge tutto il team: sviluppo, project manager, cliente etc. Lo sprint meeting ha come obiettivo la presentazione di nuove funzionalità, operative ed implementabili, che vengono fatte valutare al cliente, scegliendo come dovranno essere implementate, appunto, durante il corso dello sprint. Si configura così un sistema interattivo che consente di incrementare poco alla volta, ma molto di frequente, le funzionalità del progetto.
Contemporaneamente, fatta eccezione per il primo sprint, viene verificato l'andamento complessivo del progetto, valutando le funzionalità implementate durante lo sprint precedente, raccogliendo quindi i feedback del cliente ed eventualmente correggendo ciò che non rispecchia i suoi desiderata, rendendo pertanto il prodotto più coerente ed aumentando la soddisfazione del committente.

2.1.2 Scrum Team

Con Scrum Team si definisce l'insieme di tutti gli attori che interverranno durante il processo di sviluppo. I ruoli delle risorse che ne fanno parte sono 3:

1. product owner
2. scrum master
3. development team



Analizzando le peculiarità nel dettaglio possiamo definirli come segue:

1. Il Product Owner definisce il lavoro da svolgere e l'ordine con cui deve essere completato. Raccoglie la voce degli stakeholders (clienti, management e chiunque vanti un interesse nella buona riuscita del prodotto), le necessità dell'utente finale, i requisiti del mercato e, sulla base di questi elementi, stabilisce le priorità di sviluppo per il Team Scrum. Per queste ragioni è solitamente una figura interna all'azienda cliente, con autonomia decisionale e competenza tecnica, in grado di sopperire ad ogni problematica relativa all'utilizzo pratico del software in sviluppo. Raramente il cliente delega questo ruolo ad una figura esterna; l'aspetto rilevante è che egli sia reale conoscitore dei processi aziendali.

2. Il secondo ruolo è lo Scrum Master, il responsabile del processo e leader a servizio (servant-leader) dello Scrum Team. Chi ricopre questo ruolo dovrà essere esperto della metodologia Scrum, in modo tale da garantirne la corretta applicazione e assicurandosi che il Team comprenda e segua le regole che la caratterizzano perché il progetto abbia successo. Ciò che risulta essenziale a garantire tale riuscita è la capacità dello Scrum Master nel favorire il lavoro del Team di Sviluppo, rimuovendo ostacoli, organizzando meeting di confronto e, soprattutto, proteggendolo da ogni possibile distrazione: ogni membro del gruppo deve poter lavorare al 100% sullo sviluppo e lo Scrum Master si assicura che questo avvenga.
3. Il Team di Sviluppo compone invece la squadra di lavoro tecnico, costituito da programmatori, designer, project manager (che solitamente coincide con lo Scrum Master, anch'esso può far parte, infatti, del Team di Sviluppo). Chi concretamente porta a termine gli Sprint e fornisce le funzionalità da implementare è questo insieme coordinato di soggetti, autogestito e cross-funzionale.

2.1.3 Artefatti

Per gestire efficacemente un progetto, il metodo Scrum presenta diverse caratteristiche che lo rendono particolarmente performante. Un team che passa dal tradizionale metodo di gestione dei progetti Waterfall alla gestione dei progetti con Scrum, nota in effetti una differenza e un miglioramento su diversi aspetti, quali la collaborazione e la consegna del valore.
Sono 4 gli artefatti:

1. product backlog
2. items
3. sprint backlog
4. increment

Nello specifico:

1. Il Product Backlog è la lista ordinata di tutti gli elementi necessari al prodotto, nonché delle attività atte a realizzarlo.
2. Queste ultime vengono denominate Items e, a differenza di una tipica lista di requisiti del prodotto, contengono le User Story. La lista viene

prioritizzata, vale a dire viene deciso con quale priorità svolgere i compiti, ed è una delle principali caratteristiche di Scrum che lo differenziano dal classico metodo Waterfall di gestione dei progetti; il team si concentra sulle attività più importanti e definisce le attività da svolgere per prime con una appropriata stima di tempo. Responsabile del Product Backlog è il Product Owner: soltanto lui, infatti, può modificare il posizionamento degli items.

Aspetto da sottolineare è che il Product Backlog rimane visibile e modificabile anche quando il progetto è in corso d'opera. La cosa fondamentale è che le modifiche siano comunicate a tutti i membri dello Scrum Team e che siano comprese.

All'inizio di ogni sprint le priorità vengono poi ridefinite a seconda dell'andamento dei lavori.

Gli items sono dunque, riassumendo, una lista di compiti da portare a termine per la realizzazione del prodotto. Non solo, essi si compongono di User Story che sono una descrizione informale delle caratteristiche del prodotto che il team deve sviluppare.

A differenza di una semplice lista "to do", le User Story hanno la caratteristica di essere descritte con un criterio di impersonificazione in modo tale da raccontare la funzionalità come se la si stesse vivendo o utilizzando. Possiamo fornire una semplice frase - la più utilizzata nella cultura informatica - con la quale sia possibile definire la User Story e racchiudere tutte le caratteristiche e funzionalità che si necessitano per realizzare il prodotto:

"Come un (attore) (vorrei, dovrei o potrei svolgere) (un'azione) affinché possa raggiungere (un obiettivo)".

Es: Come un utente potrebbe effettuare la registrazione per usufruire dei corsi online.

3. Gli Sprint Backlog sono quella parte di Product Backlog che contiene i compiti e le attività da compiere solo in un determinato sprint. Durante lo Sprint Planning viene deciso quali sono gli items da inserire nello Sprint Backlog, in quanto saranno oggetto di lavoro in un determinato sprint; qui sono i membri del team stesso a decidere quali items includere e su cui lavorare: il Product Owner non ha potere decisionale in tale frangente. Il lavoro viene distribuito all'interno del

team su base volontaria e viene aggiornata quotidianamente la stima del lavoro da fare tramite i Daily Stand-up (stand-up meeting).

Può essere modificato da ogni membro del team perché appeso e visibile su un tabellone per tutti i membri del team. In questi casi viene solitamente utilizzata la metodologia Kanban, ovvero suddivisa in 4 fasi: backlog, development, testing e completed ma secondo lo scrum sono definite più fasi per garantire l'interoperabilità tra project manager e sviluppatori, ma tali aspetti saranno chiarite successivamente analizzando lo sviluppo pratico di PrimoUp.

4. Si definisce invece incremento tutto ciò che è stato realizzato durante uno sprint secondo la definizione che, di fatto, ha stabilito il team. Rispondendo alla domanda "quando un lavoro viene considerato completato?" i membri del team decidono di comune accordo cosa vogliono considerare come un incremento.

   In alcuni casi gli incrementi alla fine di uno sprint risultano essere uguali allo Sprint Backlog; ciò accade nei casi di team altamente performanti. Ma più spesso durante uno sprint il team incontra impedimenti e situazioni da risolvere anche con l'aiuto dello Scrum Master ed è dunque in virtù di tale aspetto che gli incrementi risultano essere diversi dallo Sprint Backlog.

Perché i 4 artefatti rendono performante lo Scrum Team?
Questi quattro artefatti servono per due principali ordini di motivi:

- Il primo è che il team riesce ad allocare la priorità lavorativa in base al valore dei compiti da svolgere. Questo concetto ribadisce l'importanza di consegnare valore al cliente finale, caratteristica, questa, della metodologia Agile di cui Scrum è un framework. Dalle prime fasi di progettazione del prodotto fino al suo completamento.
- Il secondo è che senza collaborazione questi artefatti non avrebbero il significato che Scrum conferisce loro. Comunicare le modifiche, preoccuparsi che i compiti siano chiari e comprensibili, offrirsi volontariamente per svolgere un lavoro, sono strategie che fanno in modo che i membri dei team interagiscano reciprocamente e generino il massimo valore.

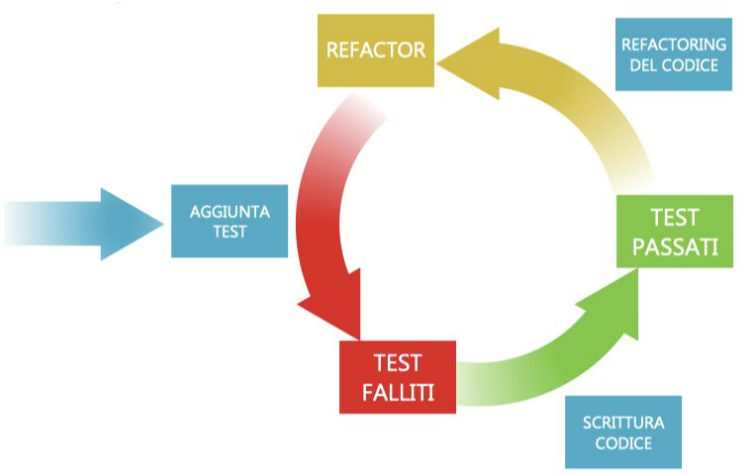
2.2 Test Driven Development e Continous Integration

Il Test Driven Development (abbreviato in TDD) è un processo di sviluppo del software, nel quale la stesura delle procedure funzionali è preceduta (driven) dalla stesura di test automatici. Il TDD si articola in cicli di sviluppo composti da tre fasi:

1. Red, cioè la prima fase, prevede che lo sviluppatore scriva un test automatico per la funzionalità da implementare. Tale test fallirà necessariamente in quanto creato preventivamente ad hoc per l'implementazione della funzionalità stessa.
2. Green: durante questa seconda fase lo sviluppatore scrive la quantità minima di codice necessaria al superamento del test.
3. Refactor: la terza fase prevede il refactoring del codice, cioè la sua ristrutturazione.

I vantaggi apportati dal TDD riguardano l'affidabilità, la solidità e la pulizia del codice.
La solidità è garantita dai test stessi che vengono generalmente ripetuti in seguito ad ogni nuova versione (rilascio software), garantendo così che il

nuovo codice introdotto non abbia ripercussioni su procedure scritte precedentemente.
La necessità di scrivere un test prima della stesura del codice aiuta lo sviluppatore nella comprensione delle specifiche, facendo inoltre emergere preventivamente eventuali errori, incomprensioni o inesattezze, nelle specifiche stesse, migliorando quindi l'affidabilità del codice. Infine i continui refactoring danno luogo ad un codice più pulito.



La Continous Integration (abbreviato CI) è una pratica, complementare al TDD, che consiste nel frequente allineamento degli ambienti di sviluppo dei singoli sviluppatori, con la mainline, cioè l'ambiente condiviso. Scopo del processo di Continuous Integration è aumentare la qualità del software eseguendo in modo continuo, durante tutto il ciclo di sviluppo, il controllo di qualità e correttezza del codice, che solitamente sono rimandati dopo la fase di rilascio. La realizzazione di un progetto software complesso richiede lo sviluppo di numerosi moduli software, legati tra loro e interdipendenti.

In questo contesto di continua evoluzione è fondamentale mantenere il controllo delle variazioni apportate al codice verificando che:

- siano rispettate le dipendenze con gli altri moduli;
- sia sempre possibile realizzare una build di tutto il sistema;
- i test automatici, associati ai singoli moduli o a livello di sistema, abbiano successo.

La Continuous Integration prevede innanzitutto la centralizzazione del repository dei sorgenti, tramite l'utilizzo di un sistema di versioning dei sorgenti, e l'utilizzo di un sistema, detto CI server, che effettui in automatico l'esecuzione dei test.

Al fine di limitare problemi derivanti dallo sviluppo simultaneo da parte di diversi programmatori, è richiesto che gli appartenenti al team di sviluppo aggiornino costantemente i dati presenti nel repository, riducendo al minimo il lasso di tempo in cui il codice sorgente rimane memorizzato esclusivamente sulla macchina di sviluppo locale del singolo programmatore, al fine di limitare i rischi di conflitti e disallineamenti rispetto la versione del repository centrale.

2.2 Progettazione in pratica

La corretta progettazione del software è stata l'attività cruciale. Una corretta progettazione deve riuscire a conciliare il più possibile le richieste del cliente con la possibilità tecnologica alla loro implementazione. Se da una parte i cliente ha accettato la progettazione con un approccio di tipo Agile, nel quale tempistiche e funzionalità non sono note a priori aiutandoci nel lavoro di analisi, è stato comunque necessario un lavoro di preventivazione per presentare un'offerta commerciale da sottoporre al CDA della società cliente per partecipare alla gara d'appalto. Definire a priori la quantità di sprint necessari per arrivare al software completo, il numero di risorse, tra sviluppatori, designer, project manager etc.. necessari è stata una valutazione molto rischiosa a livello imprenditoriale. Se da un lato sovrastimare il tempo necessario allo sviluppo porta ad un incrementare gli utili, dall'altro una progettazione sottostimata porta a perdere giornate di lavoro, causate esclusivamente dalla non precisa analisi delle problematiche e della loro risoluzione.
Grazie a diversi anni di esperienza e al lavoro di squadra messo in atto dalla mia società, posso ritenermi soddisfatto per la durata del progetto, perfettamente in linea alle aspettative di progetto. Ciò ci ha permesso di consegnare al cliente un software finito dopo circa un anno dall'inizio dei lavori, come da crono-programma. Devo ammettere però che, al fine di

garantire tempistiche esatte, per diversi sprint è stato necessario un impegno extra, lavorando di giorno e di notte, giorni feriali e festivi.

2.2.1 Meeting

Come previsto dalla metodologia Scrum, abbiamo concordato con il cliente di adottare sprint di lavoro di durata bisettimanale.
Un po' come una catena di montaggio, o più attinente all'ambito informatico ad una pipeline di un microprocessore, i primi due sprint sono stati necessari a rendere operative tutte le risorse attive sul progetto, dal designer al tester.
Il primo sprint meeting si è svolto con il lavoro di dettaglio dei requisiti voluti dal cliente a livello delle prime funzionalità da implementare. Le prime due settimane sono state perciò impiegate a redigere dei mock-up che aiutassero il nostro cliente a capire come sarebbe stato implementato il lavoro analizzando durante lo sprint meeting.

All'inizio del secondo sprint abbiamo quindi analizzato il lavoro svolto, approvato e programmato per la relativa implementazione che si sarebbe svolta durante le successive due settimane. Contemporaneamente abbiamo ripreso il lavoro di progettazione di altre funzionalità avviando così un percorso di sviluppo che ha reso attive tutte le risorse, nello specifico i programmatori in "ritardo" di uno sprint rispetto ai designer.
Il team era composto da 1 designer, 3 sviluppatori, 1 project manager e 1 tester. Negli sprint meeting sono intervenuti il mio team, il responsabile IT cliente, accompagnato da figure di riferimento per le singole aree aziendali trattate durante lo sprint e, alcune volte, l'amministratore delegato.
Avendo diversi anni di esperienza alle spalle come analista e programmatore, mio ruolo è stato da una parte progettare il software a livello tecnico, sviluppare le funzionalità e aiutare gli altri programmatori alla realizzazione dei task assegnati, possiamo pertanto racchiuderli un po' in analista, un po' sviluppatore, un po lead developer.

Un'attività importante è stata la condivisione con tutto il team dell'avanzamento dello sviluppo, allineando tutto il team sullo stato dell'arte un giorno a settimana, sprint meeting settimanali di aggiornamento per condividere cosa è stato fatto e cosa si ha in programma di fare.

Gli stand-up meeting, che sono il cuore di un progetto Agile, sono inoltre serviti a condividere le informazioni più importanti e risolvere eventuali problematiche sorte. Tipicamente in una delle pause caffè giornaliere, ogni team member condivideva cosa avesse fatto il giorno precedente e cosa era in programma per la giornata attuale, discutendo di potenziali ostacoli per i quali a volte, semplicemente grazie alla condivisione, sono bastati pochi secondi per arrivare alla soluzione.

3. SCELTA TECNOLOGIE UTILIZZATE

Capiti i problemi da risolvere e analizzati i processi aziendali il successivo step ha previsto la scelta delle tecnologie da utilizzare per implementare concretamente quanto progettato.
Primo step è stato valutare l'architettura del software. Relativamente ai software aziendali ed all'esperienza acquisita negli anni, ho avuto modo di lavorare con clienti con due filosofie di pensiero prevalente:

- tutto ciò che riguarda l'azienda rimane all'interno dell'azienda stessa, nessun dato dovrà essere fruito al di fuori della rete aziendale per ragioni di sicurezza, in questi casi è opportuno creare un software stand-alone;
- i software devono poter essere fruiti da qualunque dispositivo, da qualunque parte del mondo attraverso Internet, è invece il caso di software distribuito

Un software stand-alone è un software capace di funzionare da solo, in maniera indipendente da altri applicativi. Il software viene installato direttamente sul computer che dovrà utilizzarlo. Da un lato si avrà un sistema stabile, sviluppato ad-hoc per la macchina destinataria, evitando pertanto qualunque problema di compatibilità. D'altra parte è però poco manutenibile, risente di problemi dovuti ad aggiornamenti del sistema operativo ove risiede e i dati necessitano di continui backup in quanto memorizzati localmente.
Parliamo invece di sistemi distribuiti quando il software è costituito da diversi processi interconnessi tra loro. Le comunicazioni avvengono esclusivamente tramite lo scambio di opportuni messaggi, che rispettino una particolare struttura dati, (protocollo di comunicazione). Un sistema distribuito prevede che i dati non siano memorizzati sulla macchina che usufruirà del sistema stesso ma, a seconda della tecnologia utilizzata, risiedono in uno o più punti esterni. Il software deve far uso di un sistema di interconnessione tra nodi, nel nostro caso Internet. Ciò può essere considerato sicuramente uno svantaggio, al quale porre rimedio mediante un attento utilizzo degli strumenti di sicurezza, con continui aggiornamenti e monitoraggio. D'altro

canto sono innumerevoli le peculiarità che rendono un sistema distribuito moderno e funzionale, al passo con l'evoluzione dell'informatica, delle tecnologie hardware in essere.
Attraverso le applicazioni distribuite si garantisce la programmazione concorrente, ovvero la possibilità di suddividere il carico di lavoro su più macchine contemporaneamente. Altra peculiarità è l'essere un sistema integrato, ovvero costituito da diversi componenti, sia dal punto di vista hardware che software. Ciò garantisce la scalabilità del sistema, ovvero l'adattamento del carico di lavoro in funzione dell'aumentare dell'utilizzo del software.

3.1 Web Application

Con il termine Web Application definiamo tutte quelle applicazioni distribuite che fanno uso di protocolli web per comunicare. Il funzionamento è basato sull'utilizzo di un'applicazione residente su un server web, unico punto di accesso per recuperare le informazioni richieste. L'accesso avviene tramite un browser (o altro programma che implementi gli standard di comunicazione World Wide Web). La comunicazione avviene tra due endpoint client e server mediante specifici protocolli di comunicazione tipicamente HTTP.
Il funzionamento prevede l'utilizzo di un client per accedere all'applicazione remota, connettendosi a funzionalità di elaborazione che si troveranno sulla macchina server.
Il client (attraverso un browser) effettua la richiesta di una o più risorse (pagina web, grafica, immagini...) tramite specifiche URL. Le applicazioni web permettono la generazione di codice dinamico, lato server e lato client. Il server risponderà con il risultato generato dinamicamente, in funzione dei dati passati dal client, di ciò che è salvato nel database connesso all'applicazione o seguendo specifiche logiche di programmazione.
Le applicazioni web hanno molti vantaggi, tra i più importanti possiamo identificare:

- manutenzione semplice e veloce, senza replicazione dei software su tutti i dispositivi che dovranno usufruire del software;
- non esistono problemi di compatibilità, il software non è eseguito dal sistema operativo ma dal browser che interpreta le pagine web

fornendo un risultato visito a pagine HTML e permettendo l'interazione dell'utente attraverso Javascript. Si potrà pertanto far uso dell'applicazione da qualunque dispositivo, con l'unico vincolo che sia installato un browser (qualsiasi computer, smartphone, tablet, smart tv...);

- i dati non sono salvati sul dispositivo che fa uso dell'applicazione ma solo sul server remoto, evitando problemi dovuti ai danneggiamenti dei dispositivi e garantendo la consistenza dei dati essendo i server ridondanti e facendo uso di sistemi avanzati di memorizzazione e salvataggi.

Può essere pertanto considerata la scelta più valida oggigiorno, sfruttando al meglio la tecnologia, massimizzando l'utilizzo delle risorse e garantendo la massima sicurezza dei dati memorizzati in modo decentralizzato, mettendo in atto politiche di replicazione dati e salvataggi distribuiti.
Durante la scelta delle tecnologie utilizzate è necessario valutare scrupolosamente anche i difetti e le problematiche::

- ponendo i dati sulla Internet pubblica si creano problemi dovuti ad eventuali attacchi informatici, cryptolocker, ddos, exploit etc;
- per poter accedere ad un sistema distribuito, si dovrà disporre di una connessione stabile e veloce, in grado di evadere importanti quantità di dati in upload per consentire agli utenti il salvataggio di documenti, fotografie;
- l'esigenza di connettere diversi hardware in clinica (quindi collegati alle macchine client) e scambiare dati su un server remoto indroduce problemi di fattibilità tecnologica e comunque notevole complessità.

Nonostante i problemi che si sarebbero potuti verificare, la scelta migliore è stata far uso di un sistema client-server, prevenendo possibili attacchi, garantendo connessioni Internet nelle cliniche con velocità sufficiente all'utilizzo del software ma soprattutto progettando correttamente la comunicazione tra dispositivi hardware e sistema web.

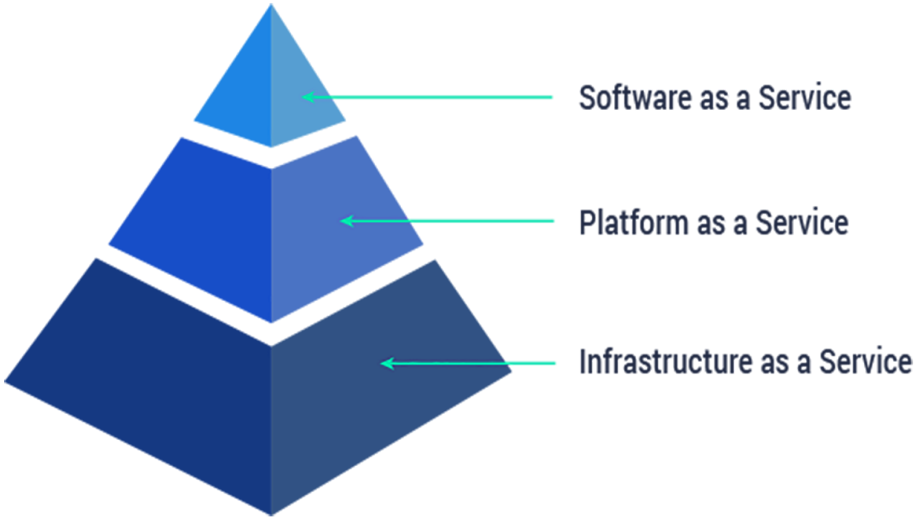
3.2 Tipologie di servizio Internet: SaaS, PaaS, IaaS

La decisione di scegliere un sistema Cloud era pertanto compiuta e rimaneva da identificare le risorse e l'architettura da utilizzare.

Con il termine cloud computing ci riferiamo ad un paradigma di erogazione di servizi offerti su richiesta da un fornitore a un cliente finale attraverso la rete internet (come l'archiviazione, l'elaborazione o la trasmissione dati), a partire da un insieme di risorse preesistenti, configurabili e disponibili in remoto sotto forma di architettura distribuita.

Le risorse non vengono pienamente configurate e messe in opera dal fornitore appositamente per l'utente, ma gli sono assegnate, rapidamente e convenientemente, grazie a procedure automatizzate, a partire da un insieme di risorse condivise con altri utenti lasciando all'utente parte dell'onere della configurazione. Quando l'utente rilascia la risorsa, essa viene similmente riconfigurata nello stato iniziale e rimessa a disposizione nell'insieme condiviso delle risorse, con altrettanta velocità ed economia per il fornitore.

La correttezza nell'uso del termine è contestata da molti esperti: se Rob van der Meulen e Christy Pettey vedono queste tecnologie come un'evoluzione tecnologica offerta dalla rete Internet, altri, come Richard Stallman, le considerano invece come una parola ingannevole ideata dalla commercializzazione per far cadere gli utenti nel tranello dei software offerti come servizio, che spesso li privano del controllo delle loro attività informatiche.
I servizi che possono essere erogati sfruttando infrastrutture cloud, sono di tre tipologie distinte: SaaS (Software as a Service), PaaS (Platform as a Service) e IaaS (Infrastructure as a Service).



L’acronimo SaaS si riferisce al termine Software as a Service, intendiamo la tipologia di servizio più completa. L’utente finale non ha bisogno di nessuna conoscenza informatica per utilizzare l’applicazione o i servizi erogati. In questo caso l’utente non ha bisogno di scaricare o installare nessun tipo di file, ma i servizi sono utilizzabili semplicemente con una connessione internet e un browser.
Il principale vantaggio è la possibilità di usare i servizi su qualsiasi dispositivo e in qualsiasi luogo. Possiamo trovare tra i classici esempi di SaaS le soluzioni di archiviazione (Dropbox, iCloud, Google Drive...). Questa tipologia di servizi viene spesso erogata con strategie freemium oppure può richiedere una sottoscrizione basata sul tempo di utilizzo o sul numero delle utenze. L’utilizzo del modello Saas, tendenzialmente riduce i costi derivanti dalle licenze, dalla gestione e installazione degli aggiornamenti.

Il software PrimoUP è stato pensato e progettato per essere utilizzato dalle cliniche del nostro cliente, ma non escludendo che in un futuro possa essere rilasciato come software SaaS, messo a disposizione pertanto per altre catene di cliniche per la cura, in prima battuta per cliniche odontoiatriche, ma valido anche per visite specialistiche e ambulatoriali.

Il PaaS (Platform as a Service) può essere visto come una piattaforma ponte tra le applicazioni (SaaS) e la parte infrastrutturale (IaaS). In questo caso, il fornitore del servizio si occupa dell’infrastruttura hardware, mentre l’utente dovrà installare il sistema operativo e occuparsi di sviluppare la sua applicazione. Questa tipologia di cloud è dedicata alle software house, alla ricerca di un ambiente di sviluppo senza avere gli oneri che derivano dalla gestione dell’hardware. Inoltre, con il cloud PaaS gli sviluppatori hanno la possibilità di sfruttare la scalabilità dinamica, l’automazione per i backup dei database e un set di linguaggi di programmazione specifici. Ci sono poi alcuni PaaS “containers” (come Red Hat OpenShift), che consentono agli sviluppatori e ai sistemisti di avere un ambiente dove l’infrastruttura e le piattaforme che supportano le applicazioni ed i database sono automatizzate e perfettamente integrate. L’utente in questo caso si trova di

fronte ad una soluzione flat e la fatturazione è, nella maggior parte dei casi, periodica.

Il software PrimoUP è rilasciato su sistemi PaaS, su macchine remote messe a disposizione da provider. Ciò ha reso il sistema facilmente manutenibile, dandomi la possibilità di intervenire in modo semplice e veloce in caso di interventi.

Infine con IaaS (Infrastructure as a Service) intendiamo l'infrastruttura hardware che si presenta alla base di ogni servizio cloud. Il provider offre un hardware virtuale (CPU, RAM, spazio e schede di rete) e quindi la flessibilità di un'infrastruttura fisica, senza l'onere per l'utente, della gestione fisica dell'hardware. Questa tipologia è dedicata agli amministratori di sistema o sistemisti, i quali non gestiscono fisicamente la struttura ma le istanze da attivare, le caratteristiche network ed infine le risorse da utilizzare.

Un altro vantaggio importante è che il fornitore di Cloud ha una fatturazione basata esclusivamente sul tempo di utilizzo, senza canoni fissi in modo da offrire la massima elasticità; va infatti sottolineato che tutto l'hardware virtuale è scalabile e misurabile automaticamente.

Anche in questo caso utilizzare un servizio IaaS ha reso possibile svincolare il software dalla capacità di calcolo necessaria. Ciò ha reso il sistema performante, scalabile in funzione delle richieste, del numero di utenti utilizzatori ma soprattutto adattabile al costante aumento di cliniche, verificatosi dall'inizio della realizzazione del software ad oggi.3.2.1 Vantaggi dell'infrastruttura As A Service.

3.2.1 Problematiche legate all'utilizzo in Cloud

Come trattato precedentemente esistono diversi problemi legati all'utilizzo di un sistema Cloud, su tematiche talmente delicate da aver creato un ambito lavorativo atto a garantire la maggior sicurezza possibile.

La sicurezza del cloud computing si riferisce ad un'ampia gamma di politiche, tecnologie e controlli atti alla protezione di dati, applicazioni e

infrastrutture associate di cloud computing. La sicurezza di questo ambito è un sotto-dominio della sicurezza informatica nel suo complesso.

I problemi di sicurezza associati al cloud computing si dividono in due ampie categorie: problemi di sicurezza affrontati dai cloud provider (organizzazioni che forniscono servizi cloud) e problemi di sicurezza affrontati dai loro clienti (aziende, organizzazioni o privati che utilizzano i servizi offerti dal cloud stesso). La responsabilità è condivisa: il provider deve assicurare che la loro infrastruttura è sicura e che i dati e le applicazioni dei clienti sono protette, mentre gli utenti devono adottare misure per rendere le loro applicazioni difficilmente violabili.

Quando un'organizzazione decide di effettuare lo storage dei propri dati o di ospitare un'applicazione sul cloud, perde la possibilità di avere l'accesso fisico ai server che contengono queste informazioni. Di conseguenza, i potenziali dati sensibili sono esposti al rischio di attacchi interni. Secondo un recente report della Cloud Security Alliance, gli attacchi che partono dall'interno del cloud provider sono la sesta più grande minaccia nel cloud computing. Pertanto, i cloud providers devono garantire che vengano eseguiti controlli approfonditi sui dipendenti che hanno accesso ai server nei loro data center.

Al fine di risparmiare risorse, ridurre i costi e mantenere alta l'efficienza, i cloud provider spesso memorizzano i dati di più clienti sulla stessa macchina. Di conseguenza, esiste la possibilità che i dati privati di un utente possano essere visualizzati da altri utenti (eventualmente anche loro concorrenti). Per gestire tali situazioni sensibili, i fornitori di servizi cloud dovrebbero garantire il corretto isolamento dei dati e la separazione logica dell'archiviazione.

L'ampio uso della virtualizzazione nell'implementazione dell'infrastruttura cloud crea problemi di sicurezza per i clienti di un servizio di cloud pubblico. La virtualizzazione altera il rapporto tra il sistema operativo e l'hardware sottostante: sia esso relativo al computing, all'archiviazione o persino al networking. Ciò introduce un ulteriore livello che deve essere configurato, gestito e protetto correttamente. Una delle preoccupazioni include la potenziale compromissione del software di virtualizzazione, o "hypervisor".

Ad esempio, una violazione della workstation dell'amministratore con il software di gestione della virtualizzazione può causare l'interruzione dei servizi erogati dall'intero data center o la riconfigurazione a piacere di un utente malintenzionato.

Ogni azienda ha il proprio sistema di gestione dell'identità per controllare l'accesso alle informazioni e alle risorse informatiche. I fornitori di servizi cloud integrano il sistema di gestione delle identità del cliente nella propria infrastruttura, utilizzando la tecnica del SSO o sfruttando un sistema di identificazione basato su dati biometrici oppure ancora fornendo un proprio sistema di gestione delle identità.
I fornitori di servizi cloud proteggono fisicamente l'hardware IT (server, router, cavi ecc.) da accessi non autorizzati, interferenze, sbalzi di corrente, furti, incendi e disastri naturali inoltre assicurano la continuità operativa. Normalmente questi servizi si possono avere usufruendo di data center di livello mondiale (ovvero, professionalmente specificati, progettati, costruiti, monitorati e gestiti). Bisogna sottolineare che la sicurezza informatica senza la sicurezza fisica dei server non vale nulla; poiché il malintenzionato che riesce a mettere mano fisicamente sulle macchine può violarle scavalcando tutta l'infrastruttura virtuale creata per proteggerle.

Varie preoccupazioni in materia di sicurezza delle informazioni relative all'IT e ad altri professionisti associati ai servizi cloud sono generalmente gestite attraverso attività pre- e post-impiego come programmi di sensibilizzazione e formazioni sulla sicurezza. Essi sono molto importanti per creare una conoscenza collettiva riguardante i rischi derivanti da azioni apparentemente innocue, come per esempio controllare che informazioni sono presenti su una chiavetta USB trovata per terra davanti all'azienda.

I cloud providers garantiscono che tutti i dati critici (ad esempio i numeri delle carte di credito o le password) siano mascherati o crittografati e che solo gli utenti autorizzati abbiano accesso ai dati nella loro interezza. Inoltre, le identità e le credenziali digitali devono essere protette come dovrebbero esserlo tutti i dati che il provider raccoglie o produce sull'attività dei clienti nel cloud.

3.3 Ambiente LAMP

L’architettura software necessaria per lo sviluppo di applicazioni web prevede l’utilizzo in primo luogo di un server web. Un server web (o web server) è un'applicazione software che, in esecuzione su un server, è in grado di gestire le richieste di trasferimento di pagine web di un client, tipicamente un web browser. La comunicazione tra server e client avviene tramite il protocollo HTTP, che utilizza la porta TCP 80 (o 8080), o eventualmente la versione sicura HTTPS, che utilizza invece la 443. Tra i principali server web coprono particolare rilievo:
IIS, Microsoft Internet Information Services è un complesso di servizi server Internet per sistemi operativi Microsoft Windows. Inizialmente distribuito come Option Pack per il sistema operativo Windows NT, venne poi integrato in Windows 2000 e Windows Server 2003. La versione corrente, integrata in Windows Server 2012 R2, è la 8.5 ed include i servizi server per i protocolli FTP, SMTP, NNTP e HTTP/HTTPS.
Apache, server web libero, sviluppato dalla Apache Software Foundation. È la piattaforma server Web modulare più diffusa, in grado di operare su una grande varietà di sistemi operativi, tra cui UNIX/Linux, Microsoft Windows e OpenVMS. È un software che realizza le funzioni di trasporto delle informazioni, di internetwork e di collegamento, ed ha il vantaggio di offrire funzioni di controllo per la sicurezza come quelle effettuate da un proxy.
nginx (pronunciato come "engine-x") è un web server/reverse proxy leggero ad alte prestazioni; è anche un server proxy di posta elettronica (IMAP / POP3), distribuito sotto licenza BSD-like. Funziona su Unix, Linux, varianti di BSD, macOS, Solaris e Microsoft Windows.

Su un server web risiedono dunque i siti web tramite hosting. L'insieme di tutti i web server interconnessi a livello mondiale dà vita al World Wide Web.
Avendo maturato diversi anni come sviluppatore, un po’ per praticità, un po’ per opportunità, ho fatto uso di Apache per la creazione del software.

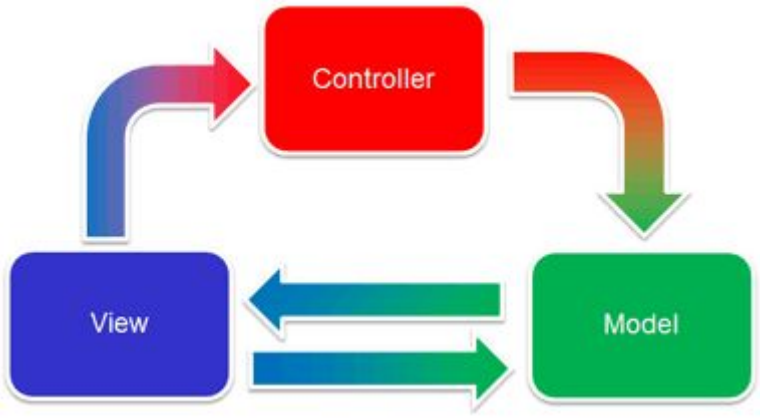
Generalizzando possiamo definire un’applicazione web come un classico programma stand-alone che non fa uso di una singola macchina, ma il cui contenuto è decentralizzato su un server remoto, sul quale avvengono i

calcoli necessari ad ottenere funzionalità e dati richiesti dall'utente. Pertanto per definizione l'applicazione web genera contenuti dinamici. Per far ciò è necessario l'utilizzo di un processore di contenuti per server web. Avendo scelto come server web Apache ho limitato la scelta a linguaggi di programmazione opensource. Tra i più utilizzati troviamo PHP e Python.
Uno dei linguaggi più completi che negli ultimi anni ha riscontrato una crescita esponenziale nel suo utilizzo è Python. Lo si può utilizzare sia per il web, che per ambienti desktop o anche solo per semplici script. Multipiattaforma, è semplice ed intuitivo sia comprenderlo che impararlo. Linguaggio ben organizzato che obbliga il programmatore ad essere ordinato.
PHP è invece molto diffuso per la sua semplicità di utilizzo. Nell'ambito web è sicuramente il più diffuso. PHP è diffuso in ambito web da molto tempo ed è ampiamente utilizzato anche oggi. Di conseguenza i sistemi. Per quanto possa essere più performante e moderno python, per ragioni prettamente commerciali e per la maggior facilità a creare un team di sviluppo ho preferito utilizzare PHP.
Altro componente a completare l'architettura è il DBMS. Scelta obbligata per aver utilizzato Apache e PHP è l'uso del database relazionale più diffuso in ambito opensource: Mysql. E' un relational database management system (RDBMS) composto da un client a riga di comando e un server, entrambi disponibili sia per sistemi Unix e Unix-like sia per Windows; le piattaforme principali di riferimento sono Linux e Oracle Solaris. Software libero rilasciato a doppia licenza, compresa la GNU General Public License, sviluppato per essere il più possibile conforme agli standard ANSI SQL e ODBC SQL.
Ecco pertanto delineata la classica architettura LAMP, acronimo che indica l'utilizzo su ambienti Linux di Apache, Mysql e Php.

3.4 Framework Model-View-Controller

MVC, acronimo di Model-View-Controller, è un pattern architetturale pensato per lo sviluppo della User Interface. Storicamente introdotto nel 1979 in Smalltalk, ma diventato famoso probabilmente per la sua implementazione Java con il framework Struts, permette lo sviluppo di client, restituendo al presentation layer il compito per il quale è concepito: l'interazione con l'utente.

Molto diffuso nello sviluppo di sistemi software, in particolare nell'ambito della programmazione orientata agli oggetti e in applicazioni web, in grado di separare la logica di presentazione dei dati dalla logica di business. Questo pattern si posiziona nel livello logico o di business e di presentazione in una architettura multi-tier.



Il pattern è basato sulla separazione dei compiti fra i componenti software che interpretano tre ruoli principali:

- il model fornisce i metodi per accedere ai dati utili all'applicazione;
- il view visualizza i dati contenuti nel model e si occupa dell'interazione con utenti e agenti;
- il controller riceve i comandi dell'utente (in genere attraverso il view) e li attua modificando lo stato degli altri due componenti.

Questo schema, fra l'altro, implica anche la tradizionale separazione fra la logica applicativa (in questo contesto spesso chiamata "logica di business"), a carico del controller e del model, e l'interfaccia utente a carico del view.

Originariamente impiegato dal linguaggio Smalltalk, il pattern è stato esplicitamente o implicitamente sposato da numerose tecnologie moderne, come framework basati su PHP (Symfony, Laravel, Zend Framework, CakePHP, Yii framework, CodeIgniter), su Ruby (Ruby on Rails), su Python (Django, TurboGears, Pylons, Web2py, Zope), su Java (Spring, JSF e Struts), su Objective C o su .NET.

La scelta di un framework che facesse uso di un pattern MVC usando architettura LAMP è ricaduta su Laravel, framework open source di tipo MVC scritto in PHP per lo sviluppo di applicazioni web, creato nel 2011 da Taylor Otwell come derivazione di Symfony.

Distribuito con licenza MIT, mantiene tutto il codice disponibile su GitHub, servizio di hosting per progetti software (il nome deriva dal fatto che "GitHub" è una implementazione dello strumento di controllo versione distribuito Git). Laravel è ad oggi il framework PHP più popolare. Alcune delle caratteristiche sono: un sistema di gestione dei pacchetti modulare con un gestore delle dipendenze dedicato, differenti modalità di accesso ai database relazionali, strumenti che aiutano la distribuzione e la manutenzione dell'applicazione, e la sua disposizione al Syntactic sugar, termine coniato dall'informatico inglese Peter J. Landin per definire costrutti sintattici di un linguaggio di programmazione che non hanno effetto sulla funzionalità o sull'espressività del linguaggio, ma ne rendono più facile ("dolce") l'uso per gli esseri umani; sostanzialmente i programmatori utilizzano una modalità di scrittura del codice spesso più pratico e produttivo e che genera programmi più facili da leggere e mantenere.

3.5 Observer Pattern

Il pattern Observer si basa su uno o più oggetti, chiamati osservatori o observer, che vengono registrati per gestire un evento che potrebbe essere generato dall'oggetto "osservato", che può essere chiamato soggetto.
Oltre all'observer esiste il concrete Observer, che si differenzia dal primo in quanto implementa direttamente le azioni da compiere in risposta ad un messaggio; riepilogando, il primo è una classe astratta, il secondo no.
Uno degli aspetti fondamentali è che tutto il funzionamento dell'observer si basa su meccanismi di callback, implementabili in diversi modi, o tramite funzioni virtuali o tramite puntatori a funzioni passati quali argomenti nel momento della registrazione dell'observer, e spesso a questa funzione vengono passati dei parametri in fase di generazione dell'evento.
Il framework Laravel implementa Observer Pattern per attivare alcuni eventi, che possono essere ascoltati per agganciarsi, quando vengono eseguite varie azioni su un modello.
Questo pattern mi ha permesso di gestire alcune situazione createsi nel progetto PrimoUP in itinere, aiutandomi in alcuni casi in cui era necessario far scattare alcune azioni al fronte del verificarsi di determinate azioni: ad esempio alla chiusura di un preventivo / piano di cura inviare alcune

notifiche a predeterminati ruoli, utilizzatori del software, inviando SMS o E-Mail.

3.6 Programmazione Lato Client

L'esigenza di rendere il Web sempre più interattivo e dinamico mi ha imposto l'utilizzo di una programmazione front-end, lato client, in grado di offrire all'utilizzatore un esperienza utente moderna e veloce. Le esigenze erano diverse:

- far in modo che l'utente possa visualizzare porzioni di pagine web, con i dati costantemente aggiornati, senza doverla ricaricare l'intera pagina
- visualizzare contenuti multipli, ad esempio utilizzando finestre sovrapposte
- aggiornare il sistema memorizzando informazioni relative all'interazione utente, utile nell'elaborazione di report come ad esempio gli ultimi pazienti visitati, le azioni più frequenti...

JavaScript è uno dei linguaggi di programmazione per questo scopo. L'enorme diffusione di JavaScript è dovuta principalmente al fiorire di numerose librerie nate allo scopo di semplificare la programmazione sul browser, ma anche alla nascita di framework lato server e nel mondo mobile che lo supportano come linguaggio principale (come ad esempio NodeJS).
Il rinnovato interesse verso il linguaggio con le nuove potenzialità applicative ha fatto nascere il cosiddetto Web 2.0 ed ha fatto fiorire numerose librerie con lo scopo di semplificare alcune delle attività più comuni e di bypassare le differenze che ancora c'erano tra i Browser, favorendo una programmazione unificata e più rapida.
La concorrenza tra componenti esterni e JavaScript è durata diversi anni e vide Flash predominante sul fronte dell'interazione utente e dei formati per l'advertising, a discapito di JavaScript, che sembrava essere destinato ad un lento declino.
Nel corso degli anni sono nate infatti diverse tecnologie concorrenti client-side, con il compito di eseguire piccoli programmi da parte del browser dell'utilizzatore: Flash, ActiveX e gli Applet Java. Queste tecnologie fornivano la possibilità di realizzare funzionalità ed effetti grafici di maggior

impatto rispetto a quanto possibile con JavaScript, ma richiedevano specifici runtime o, come i controlli ActiveX, giravano solo su uno specifico browser.
La mia esperienza sull'utilizzo di Javascript risale a circa 15 anni fa, quando non esistevano librerie per facilitarne l'uso e la manipolazione delle pagine web prevedeva una conoscenza approfondita dell'argomento (come viene generato il DOM, quali eventi vengono scatenati prima e quali dopo, quali funzioni sono compatibili sui diversi browser). Con il passare degli anni mi è stato di aiuto studiare la libreria JQuery. Nasce con l'obiettivo di semplificare la selezione, la manipolazione, la gestione degli eventi e l'animazione di elementi DOM in pagine HTML, nonché semplificare l'uso di funzionalità AJAX, la gestione degli eventi e la manipolazione dei CSS.
Le sue caratteristiche permettono di astrarre le interazioni a basso livello con i contenuti delle pagine HTML. L'approccio di tipo modulare di jQuery consente la creazione semplificata di applicazioni web e contenuti dinamici versatili.

3.6.1 Programmazione Reattiva

Con il susseguirsi degli anni si sono sviluppate innumerevoli librerie per facilitare l'uso di javascript. Alcune di queste sono nate per venire incontro alle esigenze di rendere le pagine HTML reattive ai dati che li contengono. Spesso si ha bisogno di modificare il contenuto di una pagina HTML cambiando alcuni valori, concettualmente abbinati ad un modello di dato.
Per meglio comprendere il funzionamento si può far riferimento ad un esempio pratico: è nata la necessità di voler realizzare nel software PrimoUp una tabella che mostrasse l'elenco di files, ognuno dei quali avrà come attributi un nome, un formato ed una dimensione. Supponendo di voler modificare il nome di un file, con l'approccio pre-reattivo l'operazione da compiere sarebbe stata quella di recuperare il nodo contenente il nome del file modificato e cambiarne il valore (attributo innerHTML in javascript).
Con una programmazione reattiva si ha invece la possibilità di associare ad un nodo HTML un oggetto Javascript, il contenuto HTML non sarà semplicemente un testo, bensì il valore del singolo attributo. Cambiano il valore dell'attributo dell'oggetto in questione la pagina HTML reagirà al cambiamento sostituendo automaticamente il contenuto, diventando pertanto "reattivo" AI cambiamenti del modello a cui è associato.

Un'importante framework Javascript per gestire contenuti reattivi è Vue.js, in grado di adattarsi perfettamente ad ogni tipo di applicativo frontend grazie ad un intero ecosistema di componenti (plugin). Sono diversi i suoi punti di forza tra cui, senza dubbio, la versatilità, la facilità di utilizzo e soprattutto quella di installazione. Vue.js è un framework poco invasivo che lavora benissimo sia da solo che con applicativi molto utilizzati come jQuery e Laravel, motivo per il quale ho deciso di farne uso in PrimoUP.

Vue.js gode al giorno d'oggi di una grande popolarità tra gli sviluppatori frontend, grazie anche ad un'attiva community di supporto.

Ecco un esempio di come è stata creata una tabella HTML reattiva, aggiornata automaticamente ogni qualvolta si andrà a modificare l'array docs contenente gli attributi sopra descritti.

```
<template>
    <div>
        <div class="tl list-upload-documents">
            <div v-for="doc in docs" class="list-upload-document">
                <a target="_blank" :href="doc.url"><img class="filetype" :src="doc.image_filetype">{{ doc.name }}</a>
                <button v-if="!readonly" type="button" class="btn btn-sm btn-list-danger pull-right" @click="deleteFile(doc.id)">
                    <i class="fa fa-trash"></i>
                </button>
                <button v-if="!readonly" type="button" class="btn btn-sm btn-list-secondary pull-right" @click="renameFile(doc.id,doc.name)">
                    <i class="fa fa-pencil"></i>
                </button>
            </div>
        </div>
        <div class="cb col-xs-12 missingAttachment" v-if="showandatoryError()">
            <div class="alert alert-danger"><i class="fa fa-warning"></i> {{ mandatoryMessage }}</div>
        </div>
    </div>
</template>

<script>
    export default {
        props: {
            docsprop: null,
            funprop: null,
            groupname: null,
            layoutprop: null,
            mandatoryMessage: null,
            mandatory: null,
            readonly:null
        },
        data() {
            return {
                docs: [],
                fun: "",
                counter: 0,
                layout: 'table',
            }
        },
        mounted() {
            var self = this;
            this.docs = this.docsprop;
            this.fun = this.funprop;
            self.counter = self.docs.length;
            if (this.groupname == undefined) {
                $(".add-documents").click(function() {
                    self.$bus.$emit('upload-start', self.fun);
                });
```

Documenti

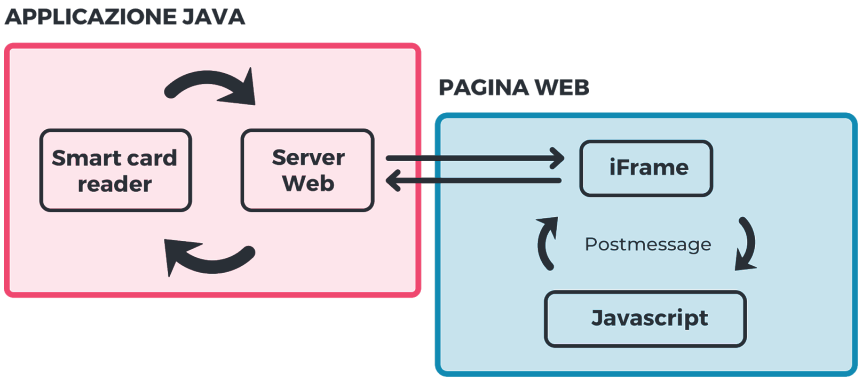
AGGIUNGI +

Data	Descrizione	Categoria	Visualizza	
02/05/2019	Ricostruzioni Anteriori	Conservativa	Ricostruzioni-Anteriori (1,53 MB)	Notifica Modifica Elimina
02/05/2019	Ricostruzioni Posteriori	Conservativa	Ricostruzioni-Posteriori (2,49 MB)	Notifica Modifica Elimina
02/05/2019	Adesione	Conservativa	Adesione (1,48 MB)	Notifica Modifica Elimina

3.7 PostMessage e WebSocket

Altro tassello importante è stato l'utilizzo di alcune tecniche che mi hanno permesso di stabilire una comunicazione tra diversi hardware connessi al client con il server.
Normalmente, due pagine diverse possono solo comunicare direttamente tra loro utilizzando JavaScript quando sono sotto la stessa origine, anche se una di esse è incorporata in un'altra (ad es. iframes) o una è aperta dall'altra (es. window.open()).
I browser limitano la condivisione dei dati tra le pagine Web, rispettando la policy di domain separation, ovvero impedire l'esecuzione di software javascript di un dominio all'interno di un altro dominio. Per eludere questa problematica è possibile utilizzare la politica di "same origin", che è alla base del modello di sicurezza delle applicazioni web. Viene utilizzata per consentire agli script in esecuzione su pagine originate dallo stesso sito di accedere ai reciproci dati, ma impedisce agli script di accedere ai dati offerti da un dominio diverso.
Se un hardware espone su un dominio specifico (ovvero rende disponibile un servizio permettendo la comunicazione su una specifica porta per consentire lo scambio dati) sarà impossibile comunicare con esso in quanto appartenente ad un altro dominio, il browser in uso bloccherà la comunicazione rispettando la policy chiamata "domain separation": nessun altro dominio può accedere al codice Javascript di un altro dominio. La domain separation impedisce che uno script malevolo in una pagina ottenga l'accesso a dati sensibili su un altro attraverso il DOM (Document Object Model) di quella pagina.
Se così non fosse, un attacker avrebbe la possibilità di creare uno script per comunicare con la pagina web vittima, ad esempio di un istituto di credito, eseguirla sul client (browser) vittima ed ottenere cookie, one time password e operare senza alcuna autorizzazione.
L'utilizzo di PostMessage rende possibile la comunicazione tra origini diverse in modo sicuro.. È molto simile ad una comunicazione asincrona, che fa uso dell'oggetto XMLHttpRequest (chiamate ajax), ma con funzionalità cross-domain.

Nella pratica l'esigenza era poter mettere in comunicazione alcuni hardware con il software PrimoUP. La comunicazione è avvenuta in prima battuta realizzando programmi stand-alone in Java. Il software ha una doppia comunicazione: da un lato comunica con l'hardware utilizzando librerie e protocolli definiti dal produttore, dall'altro riceve ed invia informazione mediante protocollo HTTP. Il programma attiva piccolo server web in attesa di ricevere connessioni, una volta arrivata la chiamata, il server web decodifica il messaggio ricevuto interpretando la richiesta e abilitando la comunicazione con l'hardware. La risposta ottenuta dall'hardware sarà poi inviata come risposta HTTP.



Si può facilmente notare come il client scambi informazioni per ottenere i dati della smart-card. All'interno della pagina web è inserito un blocco iFrame, contenente il riferimento ad una nuova pagina, creando pertanto una doppia pagina. La comunicazione tra le due istanze di javascript avviene mediante l'utilizzo di chiamate PostMessage: da un lato l'iFrame richiama la pagina web esposta dal server web del software java (ricevendo la risposta della smart-card), dall'altro la pagina "padre" scambierà informazioni con l'iFrame mediante comunicazione PostMessage.

4 PREPARAZIONE ALLO SVILUPPO

Data la complessità del progetto sarebbe stato impossibile svolgere l'intero lavoro personalmente, in tempi ragionevoli. E' stato pertanto necessario nominare un team che portasse avanti la progettazione e lo sviluppo di PrimoUP, predisporre di classici ambienti di sviluppo per condividere il progetto ed utilizzare un sistema di gestione del codice sorgente.

4.1 Team

A livello tecnico è stato necessario nominare figure che potessero svolgere egregiamente le canoniche lavorazioni di:

- creazione dei wireframe;
- produzione mockup;
- programmazione;
- coordinamento;

Il team è stato composto interamente da personale interno di Bitboss Srl, società appaltatrice per la realizzazione del software, di cui sono co-fondatore e socio ed è stato così composto:

- due programmatori senior, tra cui io, per lo sviluppo effettivo del software;
- terzo sviluppatore junior con esperienza pluriennale nella creazione di piccoli gestionali;
- un web designer per la realizzazione dei wireframe e successivi mockup;

Oltre ad essere parte integrante del team di sviluppo, ho ricoperto il ruolo di lead developer, coordinando le attività tra noi tre sviluppatori con attività di controllo e tutoring, mentre il secondo sviluppatore senior ha coordinato l'avanzamento del progetto ricoprendo anche il ruolo di project manager e coordinandosi quanto più possibile con il cliente.

4.2 Pianificazione degli ambienti

La pianificazione di un software deve essere tale da dare il tempo necessario a tutte le parti interessate di identificare e sviluppare i requisiti di sicurezza, realizzarli e verificarli.
Spesso i tempi delle verifiche (test) sono compressi per poter consegnare il sistema il prima possibile ai clienti. Questo ha conseguenze ben note, tra queste l'abbassamento del livello di qualità e sicurezza del sistema, aggravato, successivamente, da continue modifiche (spesso in emergenza).
Quando si pianificano i test è opportuno considerare non solo i tempi necessari per condurli, ma anche quelli per affrontare le carenze identificate e per verificare nuovamente il sistema dopo le correzioni.

Particolare attenzione, quando si adottano metodi di tipo Agile, va posta nella cosiddetta Definition of Done o DoD, ossia i criteri, stabiliti dagli sviluppatori, per determinare se un prodotto è finito. La DoD deve includere la buona riuscita dei test, inclusi quelli di sicurezza.

Quando si adotta una metodologia Agile, ancor di più nel caso di Scum, fin dai primissimi sviluppi vanno progettati (e poi realizzati) gli ambienti di sviluppo e test. Per essi vanno identificati i requisiti di sicurezza:

- controllo accesso agli ambienti, limitandolo il più possibile;
- gestione delle versioni del software (inclusi script e schemi) sviluppato;
- backup di quanto sviluppato;
- logging delle modifiche e dei loro autori;
- aggiornamento degli strumenti di sviluppo (vulnerabilità identificate negli strumenti di sviluppo possono avere degli impatti sui sistemi prodotti);
- quali dati usare per i test (sicuramente non quelli di produzione e, per quanto possibile, neanche delle loro copie se i dati di produzione hanno caratteristiche di riservatezza);
- quali software di terze parti (inclusi quelli free, open source e le librerie) usare e come assicurarne l'aggiornamento in caso siano identificate vulnerabilità.

4.2.1 Identificazione dei requisiti di sicurezza

I requisiti di sicurezza vanno identificati fin dalle prime fasi dello sviluppo e questo deve essere previsto in fase di pianificazione. È infatti noto che, se i requisiti di sicurezza sono identificati in un secondo tempo, la loro realizzazione ha maggiori carenze perché l'architettura adottata non è adeguata per supportarli o la loro integrazione con le parti del sistema già sviluppate presenta difficoltà.
Questo principio è quello ben noto in ambito di sicurezza informatica, ed è stato adottato anche in ambito privacy e dal GDPR (regolamento generale sulla protezione dei dati convertito in legge dall'Unione Europea nel 2018) come privacy-by-design.

I requisiti di sicurezza sono stati identificati sin dall'inizio nel backlog in modo che l'assegnazione delle priorità di sviluppo li tenga sempre presenti.

I requisiti di codifica sono stati invece inclusi nella DoD (un prodotto può essere considerato finito se il suo sviluppo ha adottato i requisiti di sicurezza nella fase di codifica).

4.2.2 I test

Per ogni funzionalità sviluppata è stato necessario effettuare diversi test, nello specifico unit test, integration test e staging test.

Le unit testing consistono nel isolare una parte del codice e comprovare che funziona alla perfezione. Sono piccoli test che valutano il comportamento dell'oggetto e della logica.
I singoli unit test sono stati creati in Laravel utilizzando PHPUnit. La loro creazione è molto semplice: si scrivono piccoli pezzi di codice atti a testare che il comportamento di una determinata funzionalità sia quello atteso; successivamente si aggiungono i nuovi test a quelli precedenti e si eseguono tutti insieme attraverso specifici tool, diversi a seconda del linguaggio di programmazione utilizzato, nel nostro caso PHPUnit. All'esecuzione di un test PHPUnit restituisce i test falliti e quelli riusciti.
Grazie all'utilizzo degli unit testing è stato possibile garantire che:

- la logica del codice è valida in qualsiasi caso;
- aumenta la leggibilità del codice e aiuta gli sviluppatori a conoscere il codice di base e ciò che aiuta a fare modifiche rapidamente;
- aiutano a realizzare una documentazione del progetto;
- i tempi di realizzazione sono minimi, sicuramente minori rispetto alla ricerca di un bug una volta presentatosi.

Successivamente sono avvenuti gli integration test, test ad alto livello con i quali ci si immedesima nell'utente utilizzatore provando il software, cercando di ottenere un risultato al compiersi di una determinata azione.
Questo genere di test verifica non solo il corretto comportamento di ogni singolo oggetto, ma anche le relazioni con gli altri componenti dell'applicazione. Nella pratica per eseguire gli integration test ci si avvale di un browser per compiere, sulle pagine, un determinato numero di azioni in maniera programmatica (come per esempio la submission di un form), per poi verificare uno specifico output visivo al termine del test. Seguendo l'esempio della submission del form, si aspetterà un messaggio di successo.
Sebbene il test d'integrazione sia senza dubbio il più completo e affidabile, dato che durante la sua esecuzione percorre l'intero stack risulta essere anche il più lento, soprattutto in software grandi come PrimoUP, dove i componenti utilizzati sono molteplici (laravel, javascript, vuejs..).

Infine, una volta che il software è stato testato mediante unit test e integration test, il nuovo componente è stato sottoposto a staging test.

Per far ciò è necessario dapprima configurare un ambiente ad-hoc un ambiente clone di quello ufficiale, con dati fittizi ma che rispecchino i dati ufficiali. L'ambiente di staging è una configurazione che replica l'ambiente di produzione. Normalmente un ambiente di staging fa riferimento ad una copia di backup dei dati ufficiali e l'accesso avviene in un'area non pubblica.

Un'ultima modalità di testing, poco utilizzata in questo progetto, è chiamata penetration test, ovvero test condotti sui sistemi finiti in ambiente di produzione, sono da considerarsi l'ultima delle tecniche di verifica della sicurezza di un sistema informatico.

4.2.3 Environment e deploy

La definizione dei test utilizzati nel progetto PrimoUP mi ha reso obbligata la creazione di diversi ambienti nei quali effettuare le operazioni di replicazione del software, la pubblicazione di una distribuzione in un predeterminato server, ovvero il deployment.

Una volta ultimato lo sviluppo e compiuti gli unit test e integration test il software contenente la nuova funzionalità viene pubblicato nell'ambiente di staging, da noi denominato "develop". L'ambiente di staging è pertanto formato da un programma non ancora ufficiale, che fa uso di un database di prova. E' in questo ambiente che sono avvenuti gli staging test, direttamente dal responsabile IT dell'azienda cliente.

Una volta testato il software e corretti eventuali bug, avviene il deployment su un nuovo ambiente, chiamato beta. La peculiarità di beta è quella di contenere il software che dovrà essere rilasciato, non ancora ufficiale, ma connesso direttamente all'ambiente ufficiale. Ciò ci ha permesso di testare le funzionalità con i dati effettivi.

Infine la pubblicazione del software avviene con il rilascio di una versione definitiva e il deploy nell'ambiente chiamato di produzione: dati e software ufficiali, effettivamente utilizzati.

Un ultimo ambiente utilizzato è sandbox. Questo non è nient'altro che un ambiente di produzione che fa uso di un database fittizio. Il senso è quello di avere un accesso al software ufficiale potendo liberamente modificare qualunque dato presente. E' il caso dell'ambiente dove è possibile testare integrazione con software terzi, o da utilizzare per presentazione del software in varie convention o meeting.

I passaggi di ambiente vanno controllati e sottoponendoli ad autorizzazione. Le persone che possono autorizzare il passaggio non dovrebbero essere né gli stessi sviluppatori, né gli stessi addetti ai test, in modo da evitare conflitti di interesse (spesso gli errori non sono commessi volontariamente, ma perché le persone sono orientate a soddisfare i requisiti di costo e di tempo dei progetti, non quelli di sicurezza). Chi autorizza il passaggio di ambiente deve verificare che siano state condotte tutte le fasi di progettazione e test previste e che includano anche gli aspetti di sicurezza.
I metodi di tipo Agile non prevedono il ruolo di addetto ai test o di "autorizzatore" al passaggio di ambiente. Per questo deve essere ben definita la Definition of Done e ne deve essere verificata l'applicazione.
I passaggi di ambiente sono avvenuti interamente con processi automatizzati (utilizzando opportuni script), in modo da ridurre al minimo l'errore umano in occasione dei passaggi in produzione.
Anche in caso di correzione di eventuali bug bloccanti, da correggere nel tempo minore possibile, il software non è stato modificato in ambiente di test o produzione, ma solo in quello di sviluppo. Questo perché, in caso contrario, gli sviluppatori si dimenticano spesso di riportare la modifica in ambiente di sviluppo. Questo ha come conseguenza che il successivo cambiamento, prodotto in ambiente di sviluppo, cancella quanto modificato nei soli ambienti di test o produzione e quindi le vulnerabilità non corrette in ambiente di sviluppo sono riportate nuovamente in ambiente di produzione.

4.3 Gestione del Repository

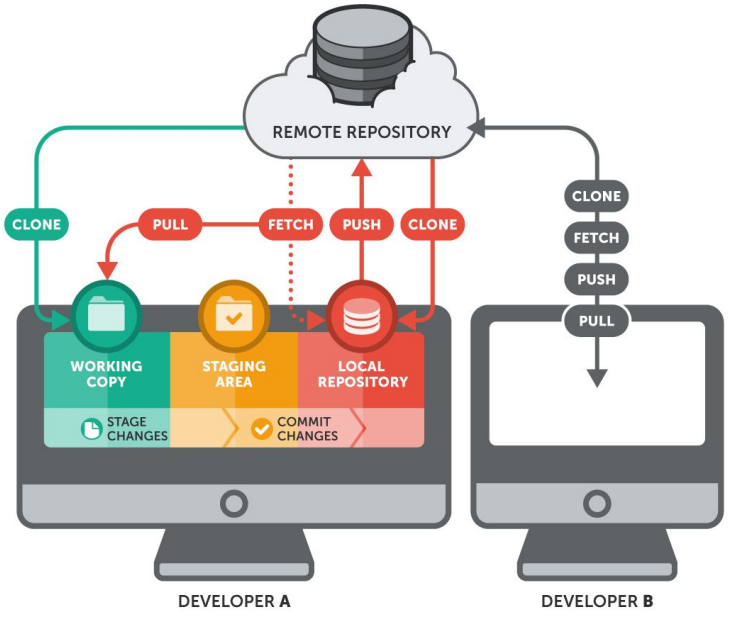
Il version control è una tecnica (implementata usando uno specifico software) utilizzata per gestire le modifiche ai file sorgenti di un'applicazione, organizzandoli nel tempo e permettendo la sincronizzazione delle modifiche da parte di più sviluppatori. È possibile utilizzare il version control per

controllare file sorgenti, file binari o risorse (immagini, archivi...) risorse digitali.

È un componente importante della gestione della configurazione del software. Il version control consente a più sviluppatori, progettisti e membri del team di lavorare insieme sullo stesso progetto. Man mano che lo sviluppo diventa più complesso, c'è una maggiore necessità di gestire più versioni di intere porzioni di software. Può infatti accadere che una porzione di software venga modificata nel tempo e solo successivamente si presenti la necessità di tornare alla versione precedente, mi sono trovato ad esempio a dover recuperare alcune pagine web allo stato di modifica ad una certa data. Grazie al version control ho potuto ottenere la versione di una stessa pagina ad una certa data.
Un'altra motivazione che rende importante l'utilizzo di un version control è l'individuazione di chi ha modificato una certa risorsa, in quale data e con quale motivazione. Ciò rende possibile un controllo, oltre che sul codice, anche sullo stato del progetto, permettendo una comparazione tra i vari sviluppatori.
Inoltre senza l'utilizzo di un version control non sarebbe possibile ricordarsi quali files sono stati modificati in un certo lasso di tempo e quali discrepanze esistono negli vari ambienti utilizzati (staging, production...).

Ma come funziona tecnicamente un version control, quali sono le operazioni che si possono svolgere e come fa uno sviluppatore ad avere il codice sorgente costantemente aggiornato?
Innanzitutto il version control prevede che tutto il codice sia salvato in un'area comune su Internet pubblica, quest'area è il Repository (remote repository). L'applicazione di version control dispone di alcuni comandi in grado di controllare il codice di programmazione. Attraverso l'operazione CLONE il codice viene copiato dal repository alla macchina che esegue il comando, creano una copia locale del software e un repository locale, clone del repository remoto. Ogni qualvolta si apporteranno modifiche al codice potranno essere salvate attraverso l'operazione COMMIT, che prevede la possibilità di aggiungere commenti per dare una descrizione al lavoro svolto.

Il codice così salvato rimane sulla macchina locale. In caso di rottura dell'hard disk della macchina locale tutte le modifiche saranno perse. Inoltre gli altri sviluppatori non hanno alcuna informazione relativa alle modifiche effettuate. Una volta eseguite una o più operazione di commit, il repository remoto può essere aggiornato, allineando il sistema alle modifiche locali. Questa operazione avviene mediante il comando PUSH. Nel caso in cui altri sviluppatori abbiano apportato altre modifiche al repository remoto l'operazione non andrà a buon fine. Infatti, in questo caso, deve essere preceduta dal PULL, ovvero l'aggiornamento del repository locale controllando il codice presente sul repository remoto.
Pertanto, le uniche operazioni che verranno svolte sui server che riproducono i vari ambienti sono CLONE iniziale e, man mano che saranno effettuate le modifiche, PULL. Tutte le altre operazioni sono fortemente sconsigliate.



Esistono due protocolli di version control attualmente utilizzati: SVN e GIT.
Git viene definito come sistema distribuito, mentre svn è detto sistema centralizzato. In pratica questo significa che in un sistema GIT tutti gli utenti hanno la loro copia personale di codice in locale che possono mettere online in un repository per renderla fruibile ad altri membri del team. In SVN invece Ciascun membro ha una working copy e le modifiche sono inviate a un server centrale (central repository) che di volta in volta "cede" i cambiamenti agli altri membri del team.

In GIT il "master" ovvero il flusso principale di codice è in locale e non in un server centralizzato remoto come accade in SVN. In pratica questo si traduce in maggiore velocità, in quanto si lavora essenzialmente in locale e si invia online la working copy solo quando siamo soddisfatti delle modifiche. In SVN invece operazioni anche basilari, necessitano di una connessione al server centrale.

Ho deciso di utilizzare GIT per questioni prettamente pratiche, in quanto in GIT tutte le operazioni sono estremamente semplici. Inoltre GIT crea copie molto più leggere di SVN. Quest'ultimo infatti crea ben 2 copie di ciascun file: una copia usata dall'utente e una copia nascosta per operazioni come status, diff e commit. Ragion per cui, nel corso degli anni, ho progressivamente abbandonato SVN in favore di GIT.

4.3.1 Gitflow

Un Branch non è altro che un puntatore a un commit. Man mano che viene creato un nuovo commit, il branch corrente si sposta e punta al nuovo commit creato. Git usa il puntatore speciale HEAD per capire qual è il branch corrente. In condizioni normali, HEAD contiene un riferimento al branch corrente. Quando creiamo un nuovo repository, Git crea in automatico il branch di default che prende il nome di master.

Gitflow è una modalità di lavoro Git che definisce un modello di ramificazione rigoroso progettato attorno alla versione del progetto. Ciò fornisce una solida struttura per la gestione di progetti più grandi.
Gitflow è ideale per i progetti come PrimoUP che hanno un ciclo di rilascio pianificato. Questo flusso di lavoro non aggiunge nuovi concetti o comandi oltre a quanto richiesto per il flusso di lavoro standard.
L'approccio GitFlow utilizza due rami principali per gestire il controllo di versione (versioning) del progetto. Il primo, master che conserva tutte le varie versioni, release. Il secondo è il ramo develop, che è fondamentale per lo sviluppo delle prossime versione e serve come base per le future integrazioni che vedremo nei vari capitoli. È inoltre conveniente taggare tutti i commit (o merge) verso il master con un numero di versione progressiva.

Tutta la (futura) vita del nostro progetto ruoterà quindi attorno ai due rami master e develop ed essi saranno il punto di partenza e di arrivo degli altri rami e saranno l'ossatura vera e propria del repository.

Allo stato iniziale il ramo develop è un clone di master.
Per implementare una nuova funzionalità si crea un nuovo ramo (branch) a partire da develop. Una volta completato lo sviluppo della funzionalità, il ramo deve essere unito (merge) di nuovo in develop. Non deve in nessun caso essere unito direttamente al ramo master. Questa operazione può essere ripetuta più volte, fin quando si decide di rendere convalidate tutte le features implementate. Una volta che il ramo develop ha accumulato, tramite unioni (merge), abbastanza feature (o anche solo una), siamo pronti per approcciare una nuova release del progetto. Per farlo si crea un nuovo branch a partire da develop. Su questo nuovo ramo, che per prassi avrà un nome del tipo release/xx.yy, non dovremmo scrivere nuove integrazioni, ma limitarci a quei piccoli compiti che porta dietro ogni release, come fixare gli ultimi bug, aggiungere documentazione, pulire o aggiungere commenti, minificare il codice. Quando il ramo è pronto, dovremmo unirlo nel ramo master e poi nuovamente nel ramo develop.

Utilizzare questo approccio permette agli sviluppatori di lavorare e finalizzare una release mentre è possibile ancora lavorare parallelamente su altre funzioni (cioè su altri rami feature/*) che non sono state ancora rilasciate. Inoltre, utilizzando una struttura dei nomi formale e condivisa, si può capire subito, a colpo d'occhio, lo stato attuale del progetto.

I rami (branch) di hotfix vengono usati per correggere bug che affliggono la release attualmente in produzione. Sono fondamentali perché ci permetteranno di gestire bug che al momento i nostri utenti stanno riscontrando oppure di tappare falle di sicurezza zero-day. Per questo motivo i rami hotfix hanno un flusso più rapido rispetto ai rami visti in precedenza. Visto che fixano bug già distribuiti, un ramo hotfix/nomebug verrà creato a partire direttamente da master e, una volta corretto il problema, dovrà essere unito (merge) sia in master che develop (oppure nella release che attualmente si sta preparando). Un approccio di questo tipo ci permetterà

quindi di fare bug fix senza interrompere i lavori al progetto su altri rami rendendo però questo nuovo fix disponibile anche per le future release.

4.3.2 Branches

Descritti i vari ambienti di deploy del software, analizzati gli aspetti di gitflow, è stato facile identificare i vari branches fissi che ho creato per adottare un gitflow modificato allo scopo di rendere validi i vari ambienti.
La modifica al flusso gitflow che ho implementato è stata l'aggiunta di un branch intermedio tra develop e master chiamato beta. Tutte le features sono state create a partire da develop e chiuse sullo stesso branch.
La creazione della realease è avvenuta in due step. In un primo momento la creazione di uno specifico ramo da develop chiuso su beta ha definito la versione di staging.
In un secondo momento è stata creata un'ultima tipologia di release che unisce beta in master.
Possiamo quindi tradurre il passaggio da develop a master come due passaggi distinti: da develop a beta (versione di staging) e da beta a master (versione production).

Delineando questi branch è stato semplice gestire l'aggiornamento degli ambienti di sviluppo:

- develop: codice aggiornato sul branch develop, database di test;
- beta: codice aggiornato sul branch beta, database di produzione;
- produzione: codice aggiornato sul branch master, database di produzione;
- sandbox: codice aggiornato sul branch master, con database di test (un ulteriore database diverso da develop, contenente il clone del database di produzione ma con valore dei singoli dati modificati attraverso un programma da me realizzato).

5 IL SOFTWARE

Il software è stato scritto senza il supporto di alcun builder o interfaccia di sviluppo, sostanzialmente partendo da file vuoti, scrivendo riga dopo riga, siamo riusciti a creare un software di vaste dimensioni, ricco di svariati programmi, con un elevato livello di configurazione e di autorizzazione.
Possiamo definire PrimoUP un ERP/CRM con le caratteristiche di un CMS, in grado di essere completamente configurabile dall'amministratore.
Il software è utilizzabile esclusivamente previa autenticazione. Non esistono infatti pagine web visibili da utente non loggato se non quelle di login e di password dimenticata.
Per minimizzare i tempi di sviluppo abbiamo acquistato un tema web online. Il tema racchiude canonici elementi grafici utilizzando specifiche colorazioni per rendere la user interface gradevole e di piacevole utilizzo da parte dell'utente, user-friendly. Consiste in una collezione di files css e javascript da utilizzare per la generazione delle pagine html.
Il tema, chiamato Metronic Live dal produttore, è stato creato dalla società Keenthemes, che lo rilascia a pagamento qualora sia utilizzato per fini commerciali, come in questo caso. Il layout è composto da una topbar, una sidebar verticale, collapsable, ovvero richiudibile, e pagina effettiva al suo fianco. Il tema fa uso di bootstrap, definito spesso come front-end framework, è una collezione di classi css racchiuse in un unico file, che rendono molto semplice la creazione di un layout html, dando ai tag html uno stile moderni. Il tema è stato oggetto di numerose modifiche dovute all'applicazione del brand cliente, con l'utilizzo di colori specifici, modificando elementi grafici e aggiungendone di nuovi.

5.1 Le funzionalità di PrimoUP

Il software è utilizzato da una delle più grandi catene di cliniche odontoiatriche italiane, al fine di gestire la cura del paziente, a partire dalla prima visita finalizzata alla preventivazione di un piano di cura, passando per l'esecuzione delle varie cure fino ad arrivare a curare la soddisfazione del cliente con feedback e visite di controllo.

Posso pertanto elencare in modo molto approssimativo solo alcune delle funzionalità presenti nel software, tra le più importanti troviamo la gestione del paziente, del magazzino, prima nota e l'area di configurazione.

5.1.1 Gestione del paziente

Il calendario di clinica è un complesso software che può essere visualizzato in calendario giornaliero o settimanale. Il calendario giornaliero ha una rappresentazione tabellare che riporta tante righe quanti sono gli slot della giornata (configurabili in minuti, da 10min in avanti, limitato agli orari lavorativi 8.30 - 22.00). A seconda di come si decida di definire le varie colonne, sono state create due versioni differenti di calendario giornaliero:

- calendario per medico: ogni colonna rappresenta la disponibilità del singolo medico nella clinica. Saranno pertanto presenti tante colonne quanti sono i medici disponibili in quella giornata. Questo significa che il numero di colonne non è predeterminato, ma dinamico in funzione del giorno, in background sono presenti gli orari di disponibilità dei singoli medici e in sovrapposizione gli appuntamenti dei pazienti. Questa viene rappresentata con un'area di sfondo, sulla quale sarà possibile inserire gli appuntamenti dei pazienti;
- calendario per stanza: prevede la gestione delle singole stanze/riuniti, ogni colonna identifica una particolare stanza,, in background le disponibilità dei medici sulla singola stanza e in sovrapposizione gli appuntamenti dei pazienti. E' pertanto possibile che una singola stanza sia assegnata, in orari diversi, a medici diversi.

Il calendario è stato realizzato per poter gestire il numero maggiore possibile di operazioni sulla stessa pagina, senza doversi spostare su altri programmi. Mediante l'utilizzo di schermate modali multiple (sovrapposte alla pagina principale) avviene la creazione dei pazienti, l'assegnazione di particolari cure nel singolo appuntamento, definizione delle operazioni da effettuare (pagamenti, fatturazione...), cambio di stato dell'appuntamento, triage telefonico, triage fisico e molto altro ancora. Ad ogni medico (e per ogni stanza definita) viene assegnato un calendario di presenza settimanale, al quale verranno aggiunte o eliminate le variazioni di presenza nei singoli giorni (come presenze o assenze eccezionali). E' di gran lunga il programma

utilizzato dal gruppo di persone più ampio di utenti (receptionist, call center, assistenti...).

Un'altra area comprendente svariati programmi è la scheda paziente, strutturato a tab verticali per gestire i dati anagrafici del paziente, tessere di sconto, relazioni di parentela con altri pazienti... Il triage viene creato direttamente in questa schermata, un elenco di domande finalizzate a conoscere lo stato di salute del paziente, elenco appuntamenti presenti, passati e futuri, visite ambulatoriali con gestione degli eseguiti e fatturazione, creazione e visualizzazione consensi informati, elenco piani di cura e preventivi, upload documentale e tutto ciò che riguarda gli aspetti generali del paziente.
Ogni paziente deve rilasciare l'autorizzazione in clinica all'utilizzo dei propri dati personali, consenso al trattamento dei propri dati personali, anche detto consenso privacy. A seconda delle prestazioni che il paziente dovrà effettuare dovranno inoltre essere rilasciate altre autorizzazioni.

Infine, il terzo macro-software relativa al paziente riguarda la gestione del singolo piano di cura, con la possibilità di modificare le singole prestazioni in corso d'opera, gestione dei singoli consensi per il piano di cura, piano pagamenti, fatturazione, gestione eseguiti, progetti di finanziamento, download di tutta la cartella clinica e molto altro. Attraverso questo programma sarà possibile definire se una prestazione sia stata pagata e fatturata, programmata o eseguita. L'operazione di marcare come eseguita una prestazione farà maturare un compenso al medico che lavorato.

5.1.2 Le prescrizioni

Le attività svolte in clinica fanno riferimento alle attività dirette con i pazienti, dalla registrazione dei dati anagrafici fino alla gestione delle cure più o meno complesse. Tutte le attività di tipo odontoiatrico sono svolte direttamente in clinica, dall'ablazione tartaro fino ad interventi ricostruttiiv. Le prestazioni odontotecniche, ovvero la creazione di manufatti come protesi dentali, vengono inviate a laboratori esterni, attualmente un unico grande laboratorio appartenente alla società stessa. Il software è stato però

progettato per prevedere l'utilizzo di più laboratori per la realizzazione dei manufatti.
Il funzionamento su PrimoUP parte dall'esecuzione di una prestazione a cui sono associate una serie di fasi di laboratorio. Per poter completare l'esecuzione è necessario inviare la prescrizione in laboratorio. Una volta ricevuta la prescrizione, il laboratorio inizierà la lavorazione, suddivisa per fasi di lavorazione. L'incarico passa tra i vari addetti e, una volta ultimato, il manufatto sarà reinviato in clinica e la prescrizione potrà considerarsi chiusa. Ciò darà la possibilità di completare l'esecuzione della prestazione iniziale.

5.1.3 Il magazzino

Ogni clinica si trova a dover gestire il materiale utilizzato (guanti, siringhe, anestetici, garze...). Il software permette di monitorare il materiale presente in clinica, amministrando le giacenze di magazzino, con valorizzazione nelle scorte mediante le classiche modalità LIFO (last-in, first-out, oppure first-in, first-out).
Per ogni articolo è possibile definire una quantità minima di stoccaggio. In questo modo è possibile la creazione automatica degli articoli sotto-scorta.
Man mano che la merce viene utilizzata la quantità dei singoli articoli diminuirà. La registrazione sul software avviene mediante l'operazione di scarico articolo. Il sistema, leggendo la configurazione, capirà se scaricare gli articoli in modalità singola (un pezzo alla volta), in confezioni (formate da un numero di unità preconfigurate) o in pacchi (definiti come un insieme di confezioni).
Il sistema gestisce una diversa unità di misura per l'acquisto della merce e per l'operazione di scarico, è stato possibile, ad esempio, contabilizzare l'utilizzo di una singola siringa e automaticamente ricaricare un magazzino conseguentemente ad un ordine di scatole di siringhe, gestendo il riordino con quantità di scatole e non di singole siringhe. Attraverso un calcolo delle unità di misura di vendita, con gestione di quantità minime ed eventualmente articoli sostitutivi, in clinica è possibile ordinare il materiale con un semplice click nel software.
E' stata inoltre implementata la gestione dei singoli lotti, con numero e scadenza differente. Questo rende possibile la tracciabilità dei vari prodotti.

5.1.4 Prima Nota

Non poteva mancare la gestione della prima nota, ovvero delle entrate e uscite economiche per singola clinica. Attualmente, per politica aziendale, non è possibile registrare manualmente incassi, in quanto essi sono frutto di emissione e pagamento di una fattura. Le entrate sono pertanto gestite automaticamente attraverso la creazione delle fatture.
I costi rappresentano invece spese sostenute dalle cliniche, comprovate da documentazione fotografica da allegare obbligatoriamente.
Inoltre gli incassi contanti saranno soggetti alle operazioni di versamento su conto corrente.
Al fine di garantire la quadratura contabile, giornalmente il sistema gestisce il controllo di cassa al fine di impedire (o comunque limitare il più possibile) eventuali ammanchi per comportamenti errati o addirittura fraudolenti.

5.1.5 Impostazioni

Una delle caratteristiche fondamentali di PrimoUP è la configurazione, anche avanzata, di ogni aspetto riguardante il software. Il tutto avviene mediante l'utilizzo di programmi specifici. La configurazione del sistema avviene impostando alcune variabili. Allo stesso modo avviene la configurazione di tutti i sistemi di notifiche per i vari eventi derivanti dall'utilizzo del software: notifiche web, via email, invio sms e notifiche push per applicazioni mobile.

5.2 Lo sviluppo delle funzionalità

Il software è stato sviluppato sfruttando il pattern architetturale MVC (Model-View-Controller) predefinito in Laravel.
MVC è particolarmente diffuso nello sviluppo di sistemi software, legata ad una programmazione orientata agli oggetti, in grado di separare la logica di presentazione dei dati dalla logica applicativa e dalla memorizzazione dei dati.
Il protocollo HTTP viene utilizzato principalmente per lo scambio di pagine web tra client e server. Nella comunicazione client/server, la parte attiva, il browser, inoltra la richiesta di ottenimento della risorsa voluta al server. Il

server esegue il programma PHP elaborando la risposta in funzione dei dati presenti nel database. Il risultato sarà tradotto in una serie di tag HTML.
Riunire tutte queste operazioni in un unico blocco di codice, in un unico file, creerebbe molta confusione, generando un codice poco leggibile (o impossibile) e pregiudicando la manutenzione futura. Nel caso in cui decidessimo di cambiare la base dati e volessimo utilizzare banalmente un altro database (semplicemente cambiando il nome), saremo costretti a modificare la singola pagina, e con essa tutte le pagine che fanno accesso al database (tipicamente tutte). Altro scenario è il caso in cui decidessimo di implementare una grafica particolare per l'ambito mobile. Anche in questo caso dovremmo stravolgere il contenuto delle pagine web. Tanta complessità rende inoltre praticamente impossibile il debugging in caso di problemi.
Il pattern MVC mi ha permesso pertanto di sviluppare separatamente i modelli, controller e viste. I modelli sono classi usate per accedere al database, per leggere o scrivere dati. Tipicamente il software prevede tanti modelli quante sono tabella del database (escludendo le tabelle relazioni molti-molti, gestite attraverso semplici metodi della classe stessa). La creazione delle tabelle e le sue eventuali successive modifiche degli attributi, foreign-key, indici o relazioni, è avvenuta facendo uso di specifiche classi software chiamate migrations, classi create allo scopo di eseguire specifiche query. Ciò permette di tener traccia di ogni modifica sulla struttura e/o sui dati del database e di averne pertanto un controllo distribuito tra i vari programmatori, che sono in grado di capire quali classi sono da invocare per allineare il database. Il tutto è gestito da uno dei programmi del framework Laravel.

I controller sono invece il motore del software. Le operazioni tipiche gestite nei controller sono l'ottenimento dei dati (ottenendoli dai modelli), normalizzati, racchiusi in variabili che saranno inviate alle viste. L'altro compito fondamentale dei controller è il controllo dei dati pervenuti dal cliente. Ad esempio l'invio da parte del client di una collezione di dati per l'inserimento di un nuovo record in una tabella del database. Questi dati sono sottoposti al controllo da parte del controller, prima di essere inviati al modello specifico, responsabile dell'inserimento effettivo nel database. Il

controller è anche responsabile di tutto ciò che riguarda l'autorizzazione all'accesso (verifica dei privilegi per la singola operazione). I vari controller sono stati racchiusi in due distinti package: uno per tutte le richieste client avvenute per cambio pagina o per sottomissione di form, l'altro per tutte le richieste eseguite in background dal client mediante chiamate asincrone Ajax.

Le viste contengono invece l'effettivo output html che sarà mostrato al client. All'interno delle singole viste vengono utilizzate le variabili ricevute dal controller (ad esempio un array per la generazione di una tabella html). Tutte le viste partono da un generico layout, che contiene il template grafico. All'interno del layout vengono "iniettate" le singole sezioni definite nella vista. Ogni vista produce un risultato di tipo responsive, adattabile al dispositivo che la visualizza, rendendo il software fruibile da computer desktop, tablet o smartphone.

All'interno delle viste è stata definita un'importante sezione javascript, utilizzata per rendere più user-friendly il software, più grazioso a livello visivo ma anche più funzionale, eseguendo una buona parte di chiamate via ajax per accesso ai dati, senza rallentare il client che ha la percezione di ottenere la pagina web molto velocemente, nonostante la grande mole di dati gestita su PrimoUP.
Per quei casi in cui è necessario modificare pochi elementi html, il javascript è scritto in jQuery o javascript nativo. Quando invece si ha l'esigenza di modificare diversi elementi HTML, modificare dati presenti in un tab e modificarne un altro, diventa utile un sistema di scambio dati tra porzioni di pagina HTML. Mediante l'utilizzo del framework Vue.JS è stato possibile definire codice javascript che legasse al contenuto HTML un oggetto dati, rendendo possibile la interconnessione di diverse sezioni della pagina.

Il codice javascript presente nei file Vue (che fanno uso di vue.js) è stato scritto in ECMAScript, diversamente, se presente nelle viste, attraverso javascript nativo. Il codice js viene compresso in un unico file per rendere più veloce lo scambio via HTTP.

Il codice css è stato scritto facendo uso di scss (Syntactically Awesome StyleSheets), un'estensione del linguaggio CSS che permette di utilizzare variabili, di creare funzioni e di organizzare il foglio di stile in più file.
Il linguaggio Sass si basa sul concetto di preprocessore CSS, il quale serve a definire fogli di stile con una forma più semplice, completa e potente rispetto ai CSS e a generare file CSS ottimizzati, aggregando le strutture definite anche in modo complesso. Anche in questo caso il codice viene compresso in un unico file (preventivamente convertito in css standard).

Il software è basato composto di pagine dinamiche, gestendo le classiche Create, Read, Update e Delete.
Primo controllo frontend attraverso javascript, successiva validazione backend.

Le classiche operazioni eseguite sui vari modelli sono racchiuse nelle operazioni chiamate CRUD (create, read, update, delete), avvenute tipicamente in due modalità, classica o single-page. Nella modalità classica, per ogni funzionalità realizzata, ho realizzato diverse viste:

- index, contenente l'elenco dei record della tabella a cui si fa riferimento, suddiviso in pagine. Ogni index prevede la gestione della ricerca per filtrare i risultati lato backend: data la grossa mole di dati da gestire, la vista non ottiene i dati completi a cui applicare le ricerche mediante javascript, bensì vengono filtrati direttamente dal controller che li invia alla vista già selezionati. In alcuni casi questa vista prevede la possibilità di eliminare direttamente la riga, inviando al controller specifico la richiesta di cancellazione.
- create, una vista contenente i tag input, select, option e tutto ciò necessario per far si che il client invii i dati al server, effettuando un primo controllo front-end lato javascript, mediante l'utilizzo di un sistema di controllo tramite espressioni regolari. Una volta validato il form verrà inviato al controller, che effettuerà un ulteriore verifica prima di passare i dati al modello per l'inserimento.
- edit, una vista molto simile alla create, che contiene un form con dati già valorizzati, per rendere possibile l'operazione di modifica. Create e edit spesso includono una stessa vista contenente il form, un unico

pezzo di vista (chiamato partial) evitando la replicazione del codice, con la differenza che il submit del form avverrà su rotte diverse, eseguendo pertanto metodi del controller distinti. La vista di modifica prevede anche la possibilità di eseguire l'operazione di delete.

- show, vista utilizzata sporadicamente, molto simile alla vista di edit nella quale non si ha però la possibilità né di modificare, né di eliminare il record a cui fa riferimento.

Al fine di evitare continui cambi pagina, ho realizzato una gestione CRUD attraverso una single page. Si tratta di una pagina index, dalla quale è possibile inserire nuove righe, modificare direttamente le singole celle della tabella, eliminare la singola riga. Una volta che l'utente ha apportato le dovute modifiche, il tutto viene inviato al server. Il controller sarà in grado di capire quali righe sono state cancellate, eliminate o inserite. Il salvataggio avviene mediante chiamate asincrone javascript usando ajax.

5.2 Gestione errori

Ad oggi il software è utilizzato costantemente da circa 2500 utente. Essendo un software aziendale l'utilizzo è intenso per tutta la giornata lavorativa. In una prima fase ho dovuto prestare assistenza al software essendo responsabile dell'intero sviluppo. Spesso mi sono trovato nella situazione di dover gestire problematiche non note, non sapendo se si trattasse di utenti che utilizzavano il software compiendo operazioni errate, dati inseriti male o bug del software. In una prima fase, per analizzare le problematiche, ho dovuto controllare i files di log del server web, con notevole dispendio di tempo e spesso senza riuscire a capire quale fosse il problematica.
Ho pertanto deciso di realizzare un sistema personalizzato per la gestione degli errori. Ad ogni ricezione di errore HTTP (tipicamente errore 500) ho inserito in una specifica tabella del database un record contenente url che ha generato l'errore, referral, metodo di richiesta (get, post o delete), eventuali variabili ricevute, id utente connesso al sistema. Il tutto completato con una vista custom per l'errore 500, contenente un generico messaggio, comprensivo dell'ID del record appena inserito in database. In questo modo, in caso di errore, ho potuto analizzare la problematica e capire se vi fosse un problema nei dati ricevuti, nella richiesta o addirittura riprodurre l'errore per

creare un fix al bug riscontrato. Un semplice piccolo programma che mi ha permesso di risparmiare tantissimo tempo nell'analisi dei problemi utente.

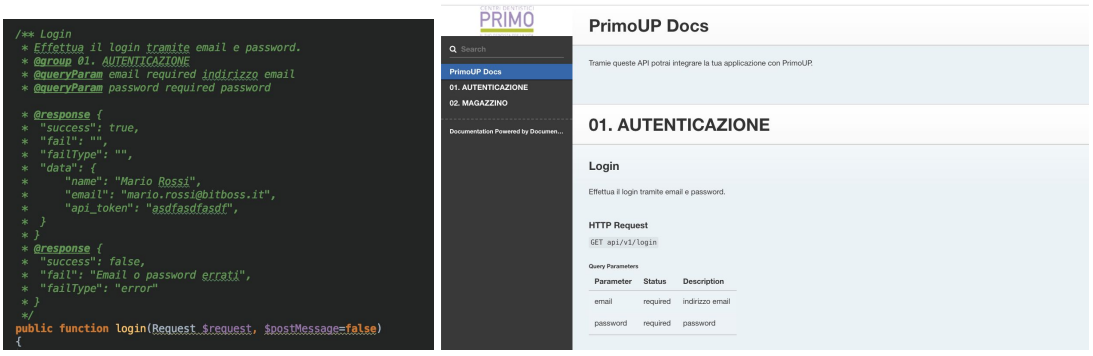
5.3 API e documentazione

Lo sviluppo delle API REST sono servite per predisporre l'integrazione con altre applicazioni e servizi.
Un'API REST è un'interfaccia di programmazione che usa HTTP per gestire dati remoti. Consiste in:

- definire delle risorse (dati) accessibili via Web, ed identificarle con degli URL;
- mappare le operazioni CRUD (Create, Retrieve, Update, Delete) su tali risorse ai verbi del protocollo HTTP (POST, GET, PUT, DELETE);
- dare la possibilità di gestire la content negotiation, cioè la possibilità per il client di chiedere al server una rappresentazione in uno specifico formato invece del classico JSON che diamo per scontato;

Attraverso l'autenticazione per mezzo di un token (ricevuto dal server in fase di login), un software di terze parti può interagire con PrimoUP inviando su determinati endpoint i dati richiesti.

Lo sviluppo delle API si sintetizza nella creazione di un unico controller contenente diversi metodi che il client potrà utilizzare. Le risposte che PrimoUP invia sono json che rappresentano uno o più record del database.
Per poter effettuare i test di funzionamento della API ho fatto uso del software PostMan, applicazione sviluppata internamente al browser Chrome.
La documentazione in questi casi è fondamentale, è infatti l'unico strumento per le aziende terze per avere il chiaro funzionamento delle API. Grazie all'utilizzo di un programma sviluppato per PHP, ho potuto generare una documentazione automaticamente. ApiDoc è un software in grado di leggere i commenti di programmazione inseriti in un controller e produrre un risultato HTML. Mi è bastato pertanto studiare il suo funzionamento per capire come commentare il codice sviluppato.



Come si può vedere in questo esempio sono bastate poche righe di commento per generare una pagina HTML semplice e intuitiva.

5.6 Privacy e GDPR

Il 19 settembre 2018 sono entrate in vigore le nuove regole in materia di trattamento e libera circolazione dei dati personali. E' con il Decreto legislativo del 10 agosto 2018, n. 101 che l'Italia ha recepito il regolamento Ue 2016/679 (GDPR), che abroga la direttiva 95/46/CE (regolamento generale sulla protezione dei dati).
Il GDPR è un regolamento dell'Unione europea in materia di trattamento dei dati personali e di privacy, adottato il 27 aprile 2016 e diventato operativo a partire dal 25 maggio 2018.
Con questo regolamento, la Commissione europea si è posta l'obiettivo di rafforzare la protezione dei dati personali di cittadini dell'Unione europea (UE) e dei residenti nell'UE, sia all'interno che all'esterno dei confini dell'UE, restituendo ai cittadini il controllo dei propri dati personali, semplificando il contesto normativo che riguarda gli affari internazionali, unificando e rendendo omogenea la normativa privacy dentro l'UE.

Esso copre molti ambiti, tra i quali il settore dei siti web, sui quali sono stati necessari interventi di adeguamento per adempiere alle norme che regolamentano il trattamento dei dati personali attraverso un sito internet.
Il regolamento si basa su due principi:

- il controllo dei dati da parte dell'interessato;
- la garanzia di maggior sicurezza possibile nella fase del trattamento da parte delle imprese, con una forte responsabilizzazione del titolare del trattamento.

Quindi le aziende hanno dovuto adeguare l'organizzazione dell'impresa, compreso il sito web, mettendo al centro la tutela dei dati personali.
Prima del regolamento europeo del 2018 adeguarsi alla normativa sulla privacy era molto più semplice: spesso bastava utilizzare un fac-simile per la privacy policy. Attualmente invece ogni sito internet deve redigere la privacy policy in base a come raccoglie i dati degli utenti e all'uso che ne fa. Una volta che il sito web si è conformato al GDPR, gli obblighi non finiscono: la protezione dei dati prosegue lungo tutta la durata del trattamento, fino alla loro cancellazione.
Il regolamento europeo è complesso e, pur definendo cosa si deve fare, spesso non dà indicazioni sul come farlo.
Nel GDPR per siti web non entrano in gioco solo i moduli che riguardano il consenso e che appaiono sul front-end del sito, ma anche i plug-in installati, i codici di tracciamento, il CMS utilizzato e la piattaforma hosting su cui è ospitato.
Ragion per cui è stato necessario avvalermi di un esperto in materia.

L'informativa è il documento più importante, da cui si deduce se il titolare del sito ha investito nel conformarsi al GDPR, o se ha copiato il documento da qualche parte, senza adeguarlo alle sue esigenze.
La privacy policy descrive le modalità di gestione del sito in riferimento al trattamento dei dati personali degli utenti che consultano il contenuto e usufruiscono dei servizi proposti dal sito stesso.
Normalmente nella policy c'è anche il richiamo alle specifiche norme di riferimento con relativo link.

Il regolamento europeo è molto chiaro sulle informazioni minime che devono essere presenti e garantite dalla policy. Relativamente al progetto PrimoUP abbiamo dovuto rispettare alcuni obblighi. Tra i più salienti troviamo:

- Informativa sui dati del titolare: il titolare dei dati è colui che tratta i dati, quindi corrisponde all'impresa titolare del sito internet. Nel nostro caso il titolare è PrimoGroup SpA, proprietario del software.
- Garantire il diritto degli interessati: i pazienti hanno diritto, in qualunque momento, di chiedere la conferma dell'esistenza dei dati,

di conoscerne contenuto e origine. Possono anche chiedere la rettifica o l'aggiornamento degli stessi. Possono altresì chiedere la cancellazione, la trasformazione in forma anonima o il blocco se il loro trattamento viola la legge. Devono essere specificati gli indirizzi e-mail a cui rivolgere le richieste. Di conseguenza è stato necessario creare alcuni software per l'estrazione dei dati al fine di essere inviati al paziente qualora li richiedesse.

L'articolo 9 del GPDR ci dice che i dati particolari (ex-sensibili) non devono essere trattati – salvo consenso esplicito dell'interessato o in caso di necessità per assolvere ad alcuni obblighi ben codificati - e ci dice anche quali sono:

- l'origine razziale o etnica;
- le opinioni politiche, le convinzioni religiose o filosofiche;
- l'appartenenza sindacale;
- i dati genetici e i dati biometrici intesi a identificare in modo univoco una persona fisica;
- i dati relativi alla salute o alla vita sessuale o all'orientamento sessuale della persona;

Ciò riguarda i dati personali che riconducono ad un singolo individuo attraverso le sue caratteristiche, relazioni, abitudini, stile di vita e così via. E quindi sono anche:

- tutte le informazioni identificative, dai dati anagrafici alle immagini che ritraggono la persona;
- i dati precedentemente definiti sensibili e quindi sottoposti a tutela particolare;
- le informazioni giudiziarie che possono rivelare l'esistenza di determinati provvedimenti giudiziari a carico della persona;
- i dati relativi alle comunicazioni elettroniche via telefono o internet come, per esempio, un indirizzo IP;
- i dati che consentono di geolocalizzare una persona e da cui è possibile capire dove è andata, quando e a volte anche con chi ;
- i dati genetici e i dati biometrici;

L'attuale quadro normativo affida al titolare il compito e la responsabilità di decidere quali misure di sicurezza possano essere considerate idonee al fine di garantire un'adeguata protezione ai dati personali trattati. Tale libertà di scelta ha portato, nel corso dell'ultimo anno – il primo da quando il GDPR è divenuto pienamente applicabile – a considerare la cifratura quale misura cardine atta a garantire riservatezza, integrità e disponibilità dei sistemi e dei servizi di trattamento.
Pertanto, oltre a impedire trattamenti di dati non conformi al GDPR, è stato necessario garantire sicurezza dei dati mediante cifratura.

5.6.1 Cifratura dei dati sensibili

Dovendo il software utilizzare dati sensibili, ogni paziente viene informato dell'utilizzo dei dati con l'esclusiva finalità riferita al trattamento delle cure prestate. Il modulo viene accettato mediante firma grafometrica e conservato nel sistema. Il documento pdf così accettato rappresenta il nulla osta al trattamento dei dati da parte di PrimoUP. Avvenuta l'accettazione della privacy policy, il paziente può essere inserito nel database del software, rispettando così le direttive imposte dal GDPR.

A questo punto i dati sensibili memorizzati devono essere memorizzati all'interno del database dividendo i dati non sensibili da quelli sensibili. Per far ciò abbiamo scelto di rendere impossibile, se non al sistema, l'associazione dei dati clinici ai singoli pazienti. Gli attributi come nome, cognome, ragione sociale, telefono, email, codice fiscale e sesso vengono cifrati e memorizzati.
Coinvolgendo la cifratura e la decifratura lo stesso attore (il software) diventava pressoché inutile la cifratura asimmetrica. Abbiamo pertanto optato per una cifratura simmetrica mediante l'utilizzo dell'algoritmo AES con controllo CBC.
La cifratura simmetrica prevede l'utilizzo di una chiave per la cifratura e la decifratura di messaggi. Il nome simmetrica deriva dal fatto che la stessa chiave deve essere utilizzata sia per le operazioni di trasformazione del testo in chiaro (plain text) in testo cifrato, sia per l'operazione opposta. La chiave utilizzata viene decisa a priori e può avere lunghezza da 128bit a 256 bit.

Il framework Laravel utilizzato per lo sviluppo di questo progetto prevede la possibilità di cifrare alcuni dati a livello software. Non da però la libertà di gestione della cifratura/decifratura direttamente dal dbms. Questo è stato un limite a cui ho dovuto far fronte, per necessità da parte di PrimoGroup. Mi è stato pertanto impossibile l'utilizzo del sistema di cifratura standard fornito in Laravel ed è stato necessario creare un sistema ad-hoc, che permettesse la cifratura e la decifratura attraverso l'esecuzione di query.
Questa esigenza nasce dall'utilizzo di altri software di business intelligence, sui quali sono stati sviluppati numerosi programmi per generare reportistica interna.
Fortunatamente il dbms mysql mette a disposizione due funzioni atte a tal scopo, aes_encrypt e aes_decrypt sono state fondamentali per la riuscita del sistema di cifratura dei dati sensibili.

5.7 Covid-19

Il 31 dicembre 2019, le autorità sanitarie cinesi hanno notificato un focolaio di casi di polmonite ad eziologia non nota nella città di Wuhan (Provincia dell'Hubei, Cina). Il 9 gennaio 2020, il China CDC (il Centro per il controllo e la prevenzione delle malattie della Cina) ha identificato un nuovo coronavirus (provvisoriamente chiamato 2019-nCoV) come causa eziologica di queste patologie. Le autorità sanitarie cinesi hanno inoltre confermato la trasmissione inter-umana del virus. L'11 febbraio l'Organizzazione Mondiale della Sanità (OMS) ha annunciato che la malattia respiratoria causata dal 2019-nCoV è stata chiamata COVID-19 (Corona Virus Disease).
I primi due casi della pandemia di COVID-19 del 2020 in Italia sono stati confermati il 30 gennaio 2020, quando due turisti provenienti dalla Cina sono risultati positivi al virus SARS-CoV-2 a Roma. Un focolaio di infezioni di COVID-19 è stato successivamente rilevato il 21 febbraio 2020 a partire da 16 casi confermati in Lombardia, a Codogno, in Provincia di Lodi, aumentati a 60 il giorno successivo con i primi decessi segnalati negli stessi giorni.
Alla data del 6 settembre 2020 sono stati registrati 277 634 casi positivi, tra cui 210.015 persone dimesse e guarite e 35.541 persone decedute, e sono stati effettuati 9.219.152 tamponi per il virus, rendendo l'Italia il diciannovesimo paese al mondo per numero di casi totali e il sesto per numero di decessi.

Tutto ciò ha reso necessaria l'adozione di politiche di contenimento che hanno portato all'obbligo di chiusura di tutte le cliniche coinvolte nel progetto PrimoUP. Il 4 maggio 2020, in seguito all'allentamento delle misure di contenimento, è stata consentita la riapertura delle cliniche, adottando accorgimenti per prevenire i rischi di contagio. Ciò ha reso necessaria la creazione di alcune funzionalità nel software: triage telefonico, cleaning slot e autocertificazione stato di salute pazienti e dipendenti.

Per combattere quanto più possibile la diffusione del virus, abbiamo implementato una single page bloccante tramite la quale ogni utente utilizzatore del software viene costretto ad autocertificare il proprio stato di salute. Il certificato, sottoposto al personale di clinica, ha una scadenza, pertanto la richiesta di compilazione avviene ciclicamente ad ogni intervallo di tempo definibile da impostazioni.
Al fine di limitare quanto più possibile i contagi e gli accessi in clinica di pazienti con possibili sintomi da Covid-19, è stato implementato nel software un modulo di anamnesi telefonica (triage) con informazioni sullo stato di salute del paziente, per poter rinviare l'appuntamento in caso di risposte positive al questionario. Lo stesso modulo viene poi ripresentato per il triage alla presenza fisica del paziente, che lo ha accettato confermando la veridicità delle informazioni riportate.
Un altro accorgimento è stato quello necessario a limitare la permanenza dei pazienti in sala d'attesa, attraverso l'adozione di un sistema che permettesse l'accesso in clinica ad un solo paziente alla volta, evitare la presenza simultanea di più pazienti in sala d'attesa. Ciò è avvenuto mediante la creazione di slot bloccanti prima e dopo ogni appuntamento. Questi periodi, fissati a 15 minuti ma di durata configurabile, sono stati chiamati cleaning slot, intervallo di tempo necessario alla sanificazione degli ambienti.

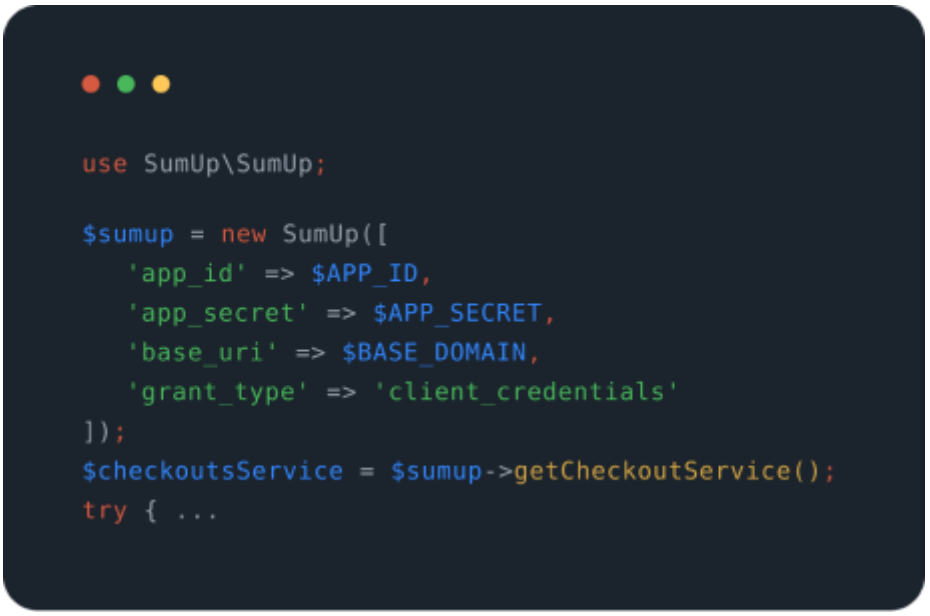
6. HARDWARE INTEROPERABILITY

Tutto ciò che è stato descritto finora fa parte di un insieme di tecnologie classiche nello sviluppo di applicazioni web, seppur il software ha previsto lunghe e complesse fasi di progettazione e realizzazione, ho fatto uso di tecnologie standard nel mondo web.
La sfida più grande non era rappresentata dal classico paradigma di programmazione client/server, bensi dall'utilizzo di diversi hardware da parte del client, controllati dal software via remoto, limitando quanto più possibile la configurazione specifica da parte dell'utente utilizzatore.
Su applicazioni web che fanno uso di sistemi client/server il software javascript viene eseguito direttamente dal client. L'interprete javascript presente nei browser non permette l'interazione con l'hardware connesso direttamente alla macchina. Questo però non significa che sia impossibile controllare risorse hardware lato client via web. Esistono infatti particolari tecniche e protocolli di comunicazione che i vari device possono utilizzare per comunicare via Internet. Si pensi ad esempio all'Internet of Things, ovvero quell'area dell'informatica riferita a piccoli dispositivi hardware controllabili appunto via Internet. Questi fanno spesso uso di protocolli specifici per lo scambio di messaggi. Altro esempio è l'utilizzo di microfono e webcam per videoconferenze via Internet. Questo è reso possibile da specifici driver e autorizzazioni che permettono lo scambio di un flusso audio/video via web. Il chè non significa però poter controllare il dispositivo via web (non è infatti possibile ad esempio cambiare la risoluzione delle immagini o incrementare il frame rate dello steam.
La sfida era pertanto mettere in comunicazione il software PrimoUP con dispositivi poco, o per nulla utilizzabili via web. Adottando opportuni accorgimenti ho realizzato un sistema di interoperabilità tra il software (mediante l'utilizzo di javascript) e i vari sistemi hardware: lettore smart card, dispositivo di pagamento POS, tavoletta per la firma grafometrica, software di scansione immagini (panoramici).

6.1 Integrazione terminale POS

Il POS è uno strumento che consente di accettare pagamenti con carte di credito, di debito o prepagate, attraverso chip o banda magnetica.
L'esigenza era quella che il software PrimoUP controllare i pagamenti mediante strumenti elettronici. L'obiettivo era permettere direttamente al software web di controllare il POS, inviando l'importo richiesto senza bisogno che l'operatore dovesse digitarlo sul dispositivo POS. Riuscire ad eseguire questa integrazione avrebbe portato diversi vantaggi.
riducendo il più possibile l'intervento umano dell'operatore per diverse ragioni.
In primo luogo automatizzare il pagamento direttamente dal software avrebbe ridotto drasticamente la probabilità di errore umano. Il paziente che si presenta in clinica ha obbligatoriamente un appuntamento, al quale sono abbinate una o più prestazioni con un determinato prezzo (in funzione del paziente e della prestazione stessa). La fattura relativa all'intervento da eseguirsi è pertanto precalcolata, di conseguenza, in caso di pagamento mediante l'uso di POS, l'importo è già conosciuto. Integrare l'utilizzo del pos inviandogli direttamente l'importo avrebbe pertanto migliorato la user experience.
Un altro aspetto che avrebbe reso vantaggioso l'utilizzo del pos via web è il fatto che in questo modo il pagamento può essere considerato certificato: alla ricezione del messaggio di conferma avrebbe reso il pagamento "trusted", effettivamente incassato. Ciò avrebbe permesso una riduzione dei tempi di lavoro dovuti al controllo incrociato degli incassi avvenuti mediante pagamenti elettronici.
Infine questa integrazione poteva minimizzare i possibili errori o comportamenti fraudolenti da parte del personale di clinica, impedendogli la stampa della fattura se non dopo aver ricevuto l'esito di pagamento corretto.

Durante la fase di analisi dell'integrazione ho valutato se continuare a far utilizzare i POS di marchio Ingenico nelle cliniche o passare all'utilizzo di una più moderna tecnologia, SumUp.



```php
use SumUp\SumUp;

$sumup = new SumUp([
    'app_id' => $APP_ID,
    'app_secret' => $APP_SECRET,
    'base_uri' => $BASE_DOMAIN,
    'grant_type' => 'client_credentials'
]);
$checkoutsService = $sumup->getCheckoutService();
try { ...
```

SumUp sembrava fare proprio al caso nostro, un software moderno, utilizzo via web, chiara e dettagliata documentazione per l'integrazione. SumUp permette di effettuare i pagamenti in diverse modalità:

- fisica, digitando l'importo sul tastierino, non era il caso nostro, l'obiettivo era proprio evitare ciò.
- applicazione mobile: mediante l'uso di app per dispositivi iOS o Android, app ufficiale o mediante sviluppo propria app con SDK proprietaria, è possibile armare il terminare inviandogli l'importo voluto. Anche questo non faceva al caso nostro perché la finalità era usare un unico software (quello web) e non far installare ad ogni operatore un'app per richiedere il pagamento, con immissione comunque manuale dell'importo
- api web: mediante l'immissione di numero carta di credito, scadenza carta e codice CVV è possibile inviare l'importo al terminale, questi dati però devono essere immessi dall'utente e il pagamento avviene come nei circuiti web (non pagobancomat, ad esempio)

SumUp non permette di effettuare pagamenti fisici gestendoli via web: o attraverso l'utilizzo di un tastierino numerico o App nativa, con il controllo dello stato del pagamento tramite api web. E' risultato impossibile utilizzare questa tecnologia per lo scambio importo ed emissione del pagamento attraverso chiamate API.

Ho ritenuto pertanto opportuno scartare questa opzione e sono passato all'analisi di Ingenico.

Il mercato dei terminali di pagamento tradizionali è ormai dominato da Ingenico, i cui POS si trovano nella maggior parte delle attività commerciali. Quando si valuta l'utilizzo di un terminale per ricevere pagamenti con carta si pensa in primo luogo a Ingenico. I terminali Ingenico non sono commercializzati direttamente dal produttore bensì dalle banche o dai rivenditori che si occupano della soluzione monetica e della manutenzione. Sono previsti più passaggi per la sottoscrizione del servizio, hanno più vincolo ma offrono più opportunità.
L'integrazione con i dispositivi Ingenico è possibile facendo uso di una libreria proprietaria fornitami dal produttore, utilizzabile solo su ambienti Microsoft.

I dispositivi Ingenico possono inoltre comunicare mediante l'utilizzo porta seriale RS-232 o tramite protocollo TCP/IP. Attraverso l'utilizzo di una DLL messa a disposizione dal fornitore, ho potuto creare un programma che comunicasse con il POS, permettendomi di:

- "armare" il POS, attivarlo in modalità di richiesta pagamento inviandogli l'importo da ricevere
- ricevere l'esito del pagamento
- inviare la richiesta di storno dell'ultima transazione

Il programma utilizzato per testare il funzionamento del dispositivo è stato scritto in C#. Avrei potuto ottimizzare tale software integrando uno scambio dati con il server web permettendo la comunicazione con PrimoUP ma la strada di utilizzare la comunicazione TCP avrebbe permesso il funzionamento senza l'installazione di alcun software aggiuntivo. Un po' per semplificare l'utilizzo, un po' per sfida personale, ho deciso di studiare il protocollo di comunicazione TCP.

6.1.1 Il protocollo 17 Ingenico

Il formato dei messaggi scambiati tra POS e client che ne richiede l'utilizzo non è documentato in quanto i terminale POS utilizzano un protocollo di comunicazione proprietario, chiamato Protocollo 17.
Non avendo alcuna documentazione per conoscere i pacchetti scambiati durante la comunicazione, è stato necessario un lavoro di reverse

engineering del sistema di comunicazione, finalizzato alla decodifica dei pacchetti scambiati via TCP/IP.
Mediante l'utilizzo di Tcpdump, un semplice software di monitoring e debugging di rete, ho analizzando il traffico di rete creato dal POS verso il router. Il protocollo è molto semplice e prevede lo scambio di alcune tipologie di messaggio. Quelle che facevano al caso mio erano solo 3:

- informazione sullo stato del POS;
- richiesta invio importo, con attesa sincrona dell'esito;
- richiesta storno operazione identificata tramite codice;

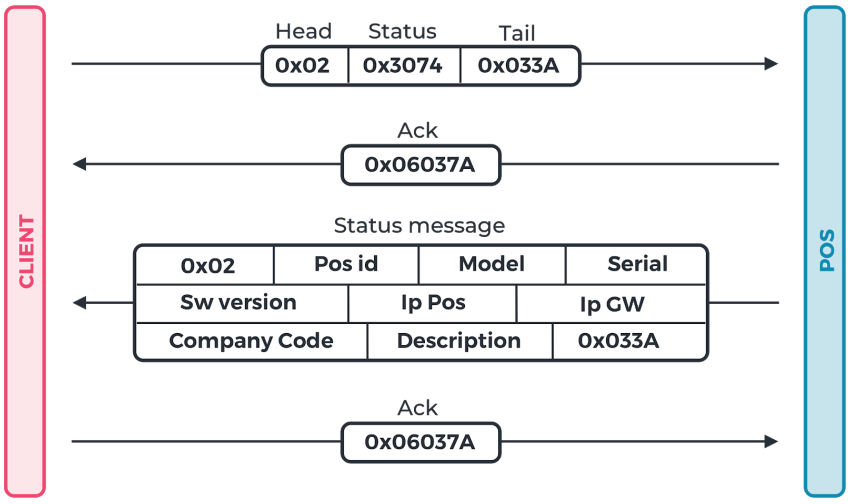
Analizzando il traffico di rete ho constatato che ad ogni invio di richiesta ricevevo due pacchetti. Il primo è un messaggio di conferma, con funzionamento analogo all'acknowledge del TCP. Il messaggio successivo conteneva la risposta al messaggio inviato. Infatti, una volta ricevuta la risposta, viene mandato lo stesso messaggio di conferma in direzione opposta, ovvero dal software al terminale POS.
I messaggi ack scambiati in entrambe le direzioni sono a lunghezza fissa e sempre contenenti il valore "0x06037A".

Il primo messaggio analizzato è il messaggio di status del pos. Si compone di un byte fisso in apertura e due byte in chiusura. I pacchetti iniziano con il byte 0x02 e termina con 0x033A.
Il contenuto è di lunghezza 10 byte e rappresenta la codifica del valore ascii "0t" con padding a sinistra di 0. Ne consegue che, essendo "0t" rappresentato in esadecimale come 0x3074, l'intero messaggio è sempre 0x020000000000000000003074033A
La risposta ricevuta, preceduta dall'ack 0x033A, presenta gli stessi tre byte per apertura (0x02) e chiusura del messaggio (0x033A). Il corpo del messaggio contiene in codifica ascii i dati relativi al pos. Tutti i campi sono posizionali, di lunghezza fissa così identificati:

- ID POS - 9 byte;
- MODELLO - 53 byte;
- SERIAL NUMBER - 15 byte;
- VERSIONE SOFTWARE - 56 byte;
- INDIRIZZO IP POS - 12 byte;

- INDIRIZZO IP GATEWAY - 12 byte;
- CODICE AZIENDA - 5 byte;
- DESCRIZIONE AZIENDA - 200 byte;

Alla ricezione del messaggio ho dovuto inviare l'ack, pena la continua ricezione del messaggio di stato.



Il messaggio di "scambio importo" con il POS, per armarlo e fare in modo che rimanesse in attesa di una carte di pagamento, contiene una struttura analoga al messaggio di stato, primo byte 0x02, ultimi due byte 0x03 seguito da un byte di checksum dell'intero messaggio. Il contenuto è la codifica di una stringa ascii contenente P seguita da 10 zeri, 2 seguita da 13 zeri e 8 cifre che identificano l'importo senza virgola, con padding a sinistra 0. Se ad esempio volessimo inviare l'importo 1,32 il messaggio risulta così composto

- P0000000000
- 20000000000000
- 00000132

Scoprire come venisse calcolato il byte di checksum non è stata cosa semplice. Per scoprirlo ho dovuto effettuare una serie di chiamate con importo diverso per scoprire quale fosse il valore calcolato in funzione dell'importo richiesto. In questo modo, con una collezione di circa 1000 importi diversi, ho capito che il byte di controllo era il risultato della sottrazione tra 127 e il risultato dell'operazione di xor byte per byte del messaggio.

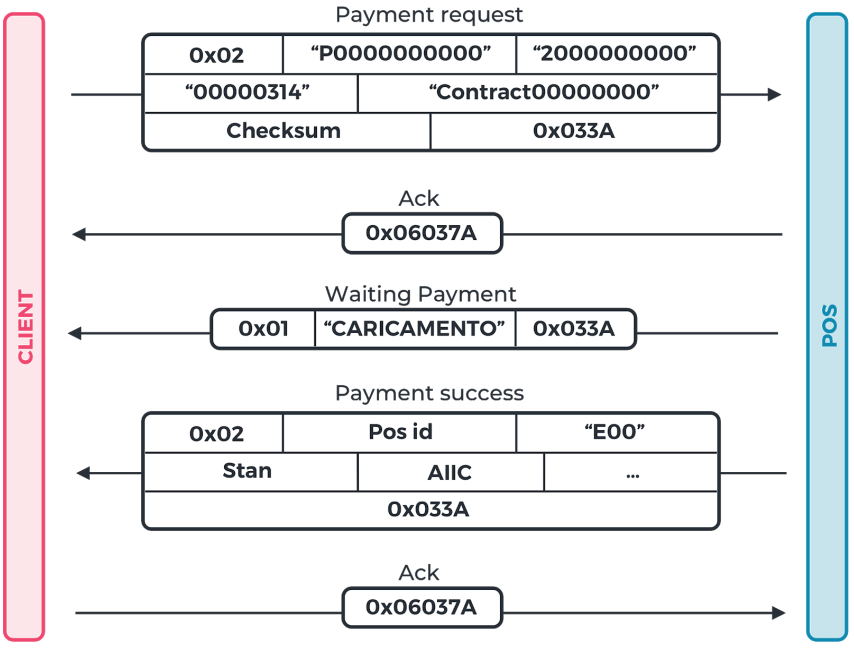
Al messaggio di ack da parte del pos, si ricevono due messaggi in sequenza. Entrambi i messaggi sono composti da un'intestazione e dal corpo. L'intestazione presenta:

- POS ID - 9 byte;
- STATO POS - 3 byte il cui valore è "E00" per ok, "E01" in caso di errore;
- STAN - 6 byte numero progressivo che identifica la transazione
- AIIC - 11 byte codice che identifica l'acquirente all'interno di quel sistema

Nel corpo del primo messaggio ricevuto subito dopo l'invio dell'ack è un campo descrittivo lungo 24 byte contenente la stringa "OPERAZIONE IN CORSO", che indica che il pos è armato ed è in attesa del passaggio della carta.

Il messaggio successivo contiene invece tutti i dati relativi alla transazione, tra i quali

- CODICE PAN - 10 byte;
- CODICE BC - 3 byte;
- CODICE AUTORIZZAZIONE - 6 byte;
- GIORNO DELL'ANNO - 3 byte;
- ORA PAGAMENTO - 4 byte, concatenazione di ore e minuti.



6.1.2 Comunicazione tra POS e PrimoUP

La comunicazione tra POS e PrimoUp è avvenuta in un primo momento tramite connessione lato server. Una volta che l'utente crea una fattura inserendo come modalità di pagamento quelle impostate per la comunicazione tramite pos (bancomat, carta di credito...) il sistema presenta una schermata modale di attesa e avviene la richiesta di pagamento via ajax al server. Il server effettua una connessione mediante l'uso di un socket TCP, attende l'esito della transazione e restituisce i dati del pagamento al client.
Per far ciò è stato necessario aprire le porte TCP dei router ed effettuare il forwarding della porta sull'IP del terminale. Per ovvie ragioni di sicurezza il port forwarding viene effettuato solo se proveniente da IP autorizzato, ovvero l'ip del server.
Una volta stabilito il corretto funzionamento, appurato che tutti i dispositivi POS fossero funzionanti, ho deciso di modificare la comunicazione separandola dal flusso standard. Ho pertanto realizzato su un nuovo dominio il sistema di comunicazione, chiamato apipos. PrimoUP, attraverso l'uso di API Rest, ha continuato a funzionare regolarmente e apipos è stato delegato alla comunicazione diretta con i terminali POS. Questa scelta deriva in primo luogo per separare i server, in modo tale da avere un sistema monitorabile esclusivamente per le comunicazione ai terminali. Altro aspetto che mi ha portato a separare gli ambienti è stato quello di creare un sistema indipendente, svincolato da PrimoUP per essere un domani utilizzato da qualunque altro sistema voglia implementare la comunicazione con POS Ingenico.

6.1.3 Problematiche di sicurezza POS

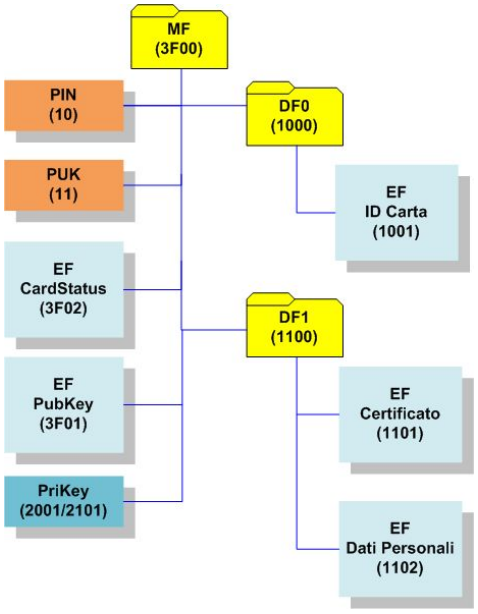
Devo ammettere che prima di analizzare il traffico inviato sulla rete per la comunicazione tra DLL e POS, ritenevo impossibile l'utilizzo della comunicazione TCP su rete, avendo ricevuto dai tecnici Ingenico contattati per l'integrazione disposizione sull'utilizzo delle DLL, unico strumento per poter utilizzare la connessione via rete.
Conoscendo il funzionamento TCP, sono rimasto incuriosito da quale fosse il sistema di cifratura implementato da una multinazionale del calibro di Ingenico, quali accorgimenti avesse adottato per la comunicazione.

Con grande stupore ho constatato l'assenza di qualunque forma di autorizzazione alla comunicazione con il POS, tanto da rendere possibile l'integrazione direttamente decodificando i messaggi scambiati. Tutti i messaggi sono infatti inviati e ricevuti in chiaro e senza necessità di fornire un'autenticazione.
Ritengo pertanto fortemente insicuro l'utilizzo di questa tecnologia, non tanto per l'utente utilizzatore quanto per il sistema in sé, facilmente attaccabile se connessi alla stessa rete locale. Si pensi infatti a cosa potrebbe succedere accedendo alla rete locale di un supermercato (a volte addirittura condivisa via wifi) che utilizza le casse automatiche per il pagamento: con un attacco di tipo MITM, nel quale l'attacker finge di essere il pos per la cassa e la cassa per il pos, è possibile controllare i messaggi inviando una notifica di pagamento corretto causando danni economici incalcolabili.

6.2 SMART Card Reader

Ogni paziente che si presenta in clinica per effettuare cure deve essere registrato sul software PrimoUP. Nella maggior parte dei casi è necessario che vengano forniti i dati identificativi ai fini fiscali (indirizzo di residenza, codice fiscale, data e luogo di nascita). Da diversi anni a questa parte è obbligatorio redigere una certificazione dei prestati ad ogni paziente, i quali potranno inserire le spese mediche in detrazione dai propri redditi, in molti casi mediante l'utilizzo del sistema di precompilazione della dichiarazione dei redditi. Per la corretta comunicazione delle certificazione all'Agenzia delle Entrate è necessario che tutti i dati inviati siano privi di errori. Può capitare che si presentino errori di digitazione del codice fiscale, ciò causa non pochi problemi in fase di compilazione delle dichiarazioni. Al fine di ridurre il più possibile gli errori umani era necessario acquisire il codice fiscale leggendolo automaticamente dal tesserino sanitario fisico del paziente, attraverso l'utilizzo di un comune smart card reader.
Per far ciò ho realizzato un software ad-hoc. Come ambiente di sviluppo ho scelto Java, in modo tale da garantire la portabilità del software su diversi sistemi operativi.

Il software è una semplice applicazione Java con alcune classi per gestire i dati e l'accesso alla lettura del filesystem presente all'interno della carta tessera sanitaria / codice fiscale.



I dati riguardanti il titolare della carta si trovano all'interno della cartella "EF Dati Personali".

E' possibile accedere ai dati presenti su una smart card utilizzando il protocollo APDU (application protocol data unit), definito come standard ISO. La comunicazione avviene mediante opportuni comandi. Per l'implementazione mi sono avvalso della libreria package javax.smartcardio, all'interno della quale è presente la classe CommandAPDU per l'invio dei comandi e relativa ricezione delle risposte.

L'accesso al file avviene percorrendo l'intero filesystem mediante l'invio di una sequenza di dati trasmessi alla carta. L'invio della stringa 00A40000023F00 indica il comando di cambio cartella (00A4000002) seguito dal nome della cartella in oggetto (3F00). Di conseguenza, per accedere al file dei dati personali, basterà inviare in sequenza i comandi:

- 00A40000023F00
- 00A40000021100
- 00A40000021102

Infine con il comando 00B0000000 si andrà a leggere il dei dati personali, tra i quali verranno restituiti i seguenti dati:

- codice fiscale
- nome
- cognome
- data di nascita
- luogo di nascita
- domicilio

Sono presenti inoltre dati relativi alla residenza e al domicilio ma, non essendo obbligatori, sono presenti solo in alcune smart card. E' stato pertanto deciso di non utilizzare tale dato.
Oltre alle funzionalità di lettura della tessera sanitaria, il software espone un server web per l'interrogazione e l'ottenimento dei dati presenti nella carta inserita nel lettore.
Il software viene caricato all'avvio del sistema operativo del computer utilizzato in clinica e ha la caratteristica di un normale server web. Attraverso normali richieste HTTP il server mette in funzione l'hardware, leggendo la card inserita e restituendo i dati letti. Di conseguenza, conoscendo la porta preconfigurata (nel mio caso 8022), basterà avviare il programma di lettura direttamente via web accedendo all'indirizzo http://localhost:8022. La risposta restituita è un json contenente i dati letti dalla carta.

Nella pagina web che permette la gestione della smart card viene inserito un iFrame nascosto che effettua la richiesta della risorsa remota. La risposta ottenuta, se in formato JSON, fornisce i dati del paziente, che saranno utilizzati per popolare i campi del form utente e inviati al sistema server mediante chiamata ajax.
Possiamo pertanto dire che il javascript responsabile della lettura dei dati della smart card fa "polling" sul software java, invia pertanto una richiesta ciclica al server esposto dal software che controlla la smart card: ad intervalli regolari aggiorno l'iFrame ottenendo le informazioni ricevute e, nel caso di risposta json, utilizzo i dati per popolare i campi.

6.3 Firma grafometrica Wacom

Uno dei primi obiettivi che si voleva raggiungere con il software PrimoUP era rendere il lavoro in clinica paperless, ogni file presente su un sistema centrale senza richiedere al paziente alcun documento da archiviare in modalità cartacea.
Uno dei primi vincoli era pertanto apporre le firme sui vari documenti: il cliente per accettazione di un preventivo, il medico per certificare un referto o compilare i diari clinici, i responsabili di area per approvare politiche di sconto.
Attualmente il quadro normativo prevede la possibilità di apporre le firme in formato elettronico dandone la stessa valenza del cartaceo purché la firma sia applicata ogni qualvolta richiesto. Per intenderci in caso di firma multipla, per accettazione ad esempio di un preventivo e delle sue clausole vessatorie, la firma dovrà essere digitata dall'utente più volte.
In un primo momento ho implementato un sistema che tramite javascript intercettava la firma su un box modale, sovrapposto alla pagina web, valorizzato tramite disegno su tablet o mediante l'uso del mouse. Questa soluzione era funzionale ma poco user-friendly, sia per i pazienti ai quali bisognava consegnare un tablet fisicamente, sia per la poca praticità nell'uso del mouse per realizzare una firma.

Successivamente abbiamo optato per l'utilizzo di una tavoletta grafica Wacom, da collocare una per ogni clinica. una tavoletta wacom per ogni clinica. Il sistema è sicuramente una soluzione più valida e confortevole per l'utente che farà uso di un dispositivo dedicatogli esclusivamente alla firma.

Realizzare il software per l'integrazione è stato molto semplice. Il produttore rilascia infatti un SDK da utilizzare su vari ambienti di programmazione. Innanzitutto è stato necessario installare i driver e l'SDK, in modo tale da caricare le DLL necessarie al funzionamento. L'installazione abilita un servizio per l'accesso all'hardware sulla porta 9000. Attraverso una semplice connessione WebSocket via javascript ho realizzato un programma per la gestione della firma.

```
ws = new WebSocket("wss://localhost:" + port.toString() + "/ws");
```

Per prima cosa è necessario avviare il software proprietario per abilitare il servizio WebSocket. L'accesso avviene mediante l'esecuzione della funzione di connessione, avviandola in tutte quelle pagine in cui è prevista l'apposizione di una firma elettronica.
In sistema riceverà una callback qualora si riuscisse a stabilire correttamente la connessione (fallisce ad esempio quando la tavoletta wacom è sconnessa dalla porta USB), e si memorizzerà il risultato ottenuto.
Senza apportare troppe modifiche al codice già scritto, ho realizzato semplicemente un bottone "wacom" all'interno della finestra per effettuare la firma normale tramite SignturePad, visibile solo in caso di esito di connessione avvenuta. Il click provoca la visualizzazione del layout custom sulla wacom. Il layout definito è suddiviso in tre aree orizzontali, la prima contenente il logo della società, la seconda contiene il box necessario per apporre la firma e il terzo contiene i tre bottoni per confermare la firma, annullare o riniziare l'operazione. Una volta confermato la Wacom invia una semplice immagine. Usando alcune funzioni native di javascript avviene il crop dell'immagine limitatamente alla sola area della firma, producendo un file jpg da inviare al server per la memorizzazione su database.

Grazie alle librerie già presenti nell'SDK non ho dovuto far altro che inserire creare codice javascript all'interno dell'applicazione. Pertanto non sono state necessarie tecniche particolari per permettere la comunicazione con PrimoUP se non l'esecuzione del software. L'unico svantaggio che possiamo riscontrare in questa integrazione è dato dal vincolo di possedere un sistema Microsoft per poter riconoscere l'hardware.

6.4 Macchinari panoramici

L'ortopantomografia, detta più comunemente panoramica dentale, è un'esame radiologico che permette di ottenere un'immagine dei denti, delle arcate dentarie e delle ossa mascellari e mandibolari. Osservando tale

immagine il dentista è in grado di valutare l'allineamento delle arcate e di individuare eventuali carie sfuggite all'esame visivo, granulomi, cisti, fratture radicolari, riassorbimenti ossei, denti inclusi ecc.
La panoramica dentale può essere quindi considerata un esame di primo livello, utile per ottenere un quadro generale sullo stato di salute dell'apparato stomatognatico, in modo da formulare una diagnosi corretta e definire il percorso terapeutico più appropriato, indispensabile per la cura del paziente.

Ogni clinica possiede un macchinario per effettuare le ortopantomografie, connesse fisicamente ad un computer per la memorizzazione delle immagini. I software proprietari utilizzati sono di tre tipi diversi a seconda del macchinario utilizzato. Nonostante ogni software memorizza le informazioni in maniera differente, il salvataggio avviene in una cartella diversa a seconda del paziente, il cui nominativo viene inserito manualmente durante l'operazione di esecuzione. Utilizzando in alcuni casi uno specifico software installato, in altri semplicemente configurando il software esistente, i dati vengono inviati su un'area ftp con relativo file contenente il matching tra paziente e relativa cartella.

Con una frequenza prestabilita molto bassa (10 minuti) avviene il controllo, da parte di PrimoUP, di tutti i file presenti nell'area ftp dedicata all'upload dei dati dei software panoramici, identificando quelli nuovi. Essendo le immagini molto grandi, prima di effettuare il salvataggio il programma si occupa di comprimere e ridurre le immagini ottimizzandole. L'operazione di ricerca delle nuove immagini, con l'aumentare delle scansioni trovate, aumenta dovendo controllare se il contenuto dell'immagine è cambiato o meno.
Al fine di limitare lo scambio di immagini molto pesanti alle stesse vengono abbinati dei digest calcolati in sha1. Confrontando gli hash AWS e quelli salvati, il sistema verifica la presenza o meno dell'immagine e in caso di assenza effettua il download.
L'alta frequenza di aggiornamento è finalizzata a rendere disponibile su PrimoUP l'immagine scansionata. Una volta effettuata la scansione, il medico potrà controllare sulla pagina del paziente i documenti acquisiti, e di conseguenza anche un'immagine scattata da pochi minuti.

7 REMOTE SERVER

I provider che mettono a disposizione server validi, ricchi di servizi sono molto e spesso hanno tutti un'affidabilità elevata, con percentuali di uptime superiori al 99%.
La scelta dei server da utilizzare per il software in produzione è stata guidata dalla mia esperienza acquisita nel corso degli ultimi anno e mi ha portato a propendere per l'utilizzo di DigitalOcean.
DigitalOcean, Inc. è un provider di infrastrutture cloud americano con sede a New York City e data center in tutto il mondo. A partire da gennaio 2018, DigitalOcean era la terza società di hosting al mondo in termini di computer collegati al web.
La console messa a disposizione da DigitalOcean è intuitiva e funzionale: i link ed i bottoni visualizzati dall'utente consentono di effettuare tutte le operazioni che uno sviluppatore si aspetta di poter compiere. Altri provider, con il passare degli anni, hanno reso abbastanza confusionaria l'interfaccia (Amazon AWS, anche per via delle numerose feature disponibili) o hanno preferito curare maggiormente l'aspetto grafico rispetto all'usabilità (Microsoft Azure).

Configurare un server può essere un'operazione lenta e può richiedere diverse ore di lavoro. Al fine di ottimizzare i tempi necessari allo startup di un software, Laravel mette a disposizione un servizio ci configurazione server remoto.
Laravel Forge è un servizio di gestione e distribuzione di un'applicazione Laravel su un server. Utilizzando un pannello semplice ed intuitivo è possibile connettere il server (nel mio caso digitalocean) a Forge. Automaticamente verranno installati i pacchetti più utilizzati e utili all'uso del framework Laravel, installando e configurando diversi software tra i quali quelli da me utilizzati:

- Nginx
- PHP
- MySQL / Postgres / MariaDB (se selezionato)
- Logrotate

L’accesso al server via SSH utilizzando la password è disabilitata di default. L’accesso dovrà essere effettuato utilizzando l’autenticazione tramite chiave asimmetrica. L'autenticazione con chiave asimmetrica si basa su coppie di chiavi pubbliche/private. L'autenticazione con chiave pubblica può essere utilizzata sia per l'autenticazione del server (host) che per quella del client (utente). Una volta generate le coppie chiave pubblica /chiave privata è possibile utilizzare la chiave privata per cifrare grantendo autenticità (sono effettivamente chi dichiaro di essere), in questo modo la chiave pubblica decifrerà il messaggio che, in caso di successo, indica che l'unico attore in grado di cifrare il messaggio è il proprietario della chiave privata (mentre tutti coloro che possiedono la chiave pubblica potranno decifrare).

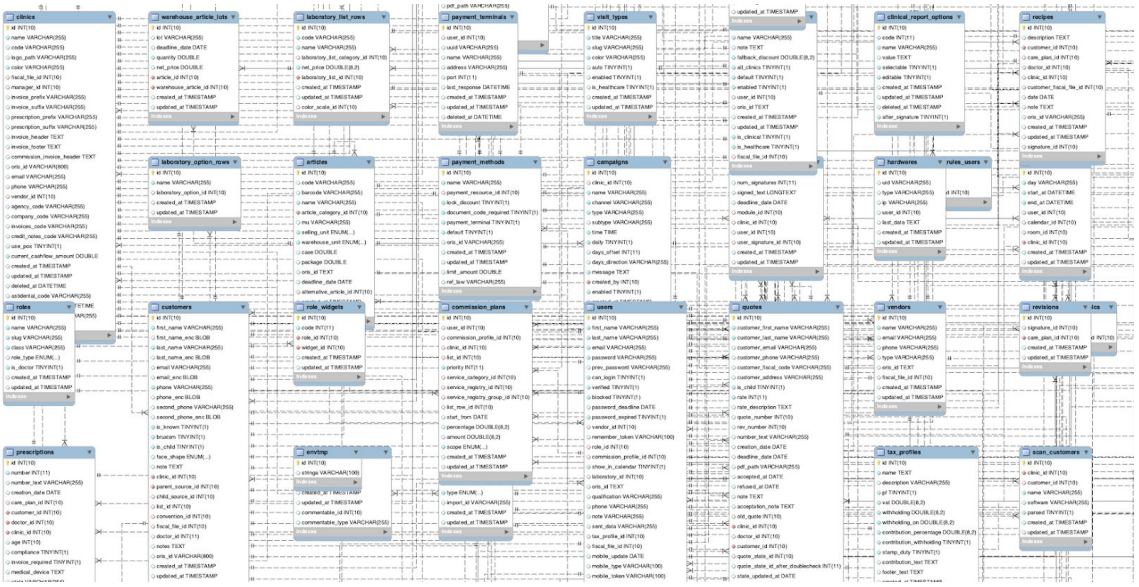
Un’importante funzionalità che offre forge è la possibilità di abilitare l’auto-deploy. Una volta sviluppata una specifica funzionalità verrà effettuato il push sul repository remoto. A questo punto il server forge, sul quale è stato configurato il repository e attivato l’auto-deploy, eseguirà uno script personalizzabile, che prevede l’effettuazione dell’operazione di push per aggiornare il sistema.
Fondamentale per il progetto è la configurazione e la gestione dei workers, demoni che eseguono specifiche operazioni evitando che il sistema sia bloccato: inviare centinaia di email è un’operazione che può richiedere diversi minuti. L’esecuzione dell’invio senza l’uso di un job (processo eseguito dal worker) provocherebbe il blocco del sistema, anche in caso di esecuzione tramite chiamata asincrona il sistema nella migliore delle ipotesi fallirà per timeout. L’esecuzione tramite worker permette l'utilizzo del software senza che l’utente venga bloccato.

7.1 Configurazione DBMS

Il sistema PrimoUP gestisce circa 200 tabelle organizzate in un unico database. La creazione e la modifica delle singole tabelle è avvenuta facendo uso del sistema di migration di Laravel. Creare una migration significa definire una classe contenente due metodi, up e down, contenenti rispettivamente una o più query necessarie alla modifica voluta e le query opposte per cancellare la modifica effettuata. In questo modo la

manipolazione della struttura del database avviene mediante l'esecuzione delle migrations non ancora caricate, eseguendo per ognuna di esse il metodo up.
E' stato fondamentale l'utilizzo di un'applicazione client per l'accesso al database, in grado di darmi il controllo dei dati, eseguire query di lettura ed eventualmente modificare direttamente singoli campi al verificarsi delle relative problematiche.

Data la grandezza del database, lo schema ER (entity-relationship) non è dei più semplici e posso pertanto riportare solo una piccola parte



In un database relazionale, le relazioni consentono di evitare dati ridondanti. Per assicurarsi che i dati siano sincronizzati, è possibile applicare l'integrità referenziale tra le tabelle. Le relazioni di integrità referenziale consentono di fare in modo che le informazioni contenute in una tabella corrispondano alle informazioni in un'altra. Ad esempio, la chiave esterna clinic_id presente nella tabella customers è valorizzata con la primary key della tabella clinics. Non è possibile definire un record in customers inserendo come attributo clinic_id un valore non presente nella tabella clinics.
Per ogni chiave esterna è stato pertanto definito un vincolo di integrità referenziale. Anche in presenza di poche tabelle è sempre opportuno procedere definendo tutte l'entità referenziali correttamente e lavorare in

maniera molto strutturata, in alternativa ci si troverà in situazioni troppo complesse nelle quali gli errori vengono propagati di tabella in tabella.

7.1.1 Stored procedure, trigger e indici

Una stored procedure è un programma scritto in SQL, mantenuto nel database stesso, archiviato nel cosiddetto database data dictionary.
A seconda delle loro caratteristiche si distinguono diversi tipi di sottoprogrammi:

- Funzioni, restituiscono un singolo valore oltre ad accettare parametri di ingresso e/o uscita;
- Procedure, non restituiscono valori ma accettano parametri di ingresso e/o uscita;
- Trigger, sono attivati da eventi;

Non ho definito alcuna funzione personalizzata, al contrario ho fatto uso di alcune procedure e trigger.

Le procedure sono state create per ottenere alcuni report utilizzati per il controllo dell'integrità dei dati. Sostanzialmente le procedure sono state definite per due finalità:

1. controllare che alcuni dati siano corretti, ho creato alcune query in grado di restituire l'elenco dei record concettualmente errati. Qualora il risultato sia un insieme non vuoto di righe è il segnale di un'anomalia da correggere. Un programma di monitoraggio avviato giornalmente esegue tutte le procedure al fine di riscontrare eventuali problemi, inviandomi la segnalazione. Questo mi ha permesso di risolvere alcuni bug verificatisi durante l'utilizzo del software.
2. correggere i dati, output dei controlli. In attesa della creazione di un fix che corregga il problema, viene utilizzata una query specifica di correzione che sarà pertanto via via meno usata.

Il trigger, nelle basi di dati, è una procedura che viene eseguita in maniera automatica in coincidenza di un determinato evento, come ad esempio la cancellazione di un record di una tabella. In questo modo si ha a disposizione una tecnica per specificare e mantenere vincoli di integrità anche complessi.

La maggior parte dei trigger viene definita per essere eseguita quando vengono apportate modifiche ai dati di una tabella. È possibile definire trigger per l'esecuzione al posto o dopo le azioni DML (Data Manipulation Language) come INSERT, UPDATE e DELETE.
I trigger aiutano a garantire che determinate azioni, come la gestione di un file di controllo, siano completate indipendentemente dal programma o utente che apporta modifiche ai dati.

Esistono due tipologie di trigger differenti: before e after. Con i trigger before rendiamo possibile la modifica di una query prima che essa sia eseguita dal dbms. Il trigger viene invocato al verificarsi di un'azione DML su una specifica tabella. Un esempio pratico di utilizzo di trigger before che ho definito è stata quella che valorizza un campo nella tabella dates. Un appuntamento a calendario viene memorizzato nella tabella dates, valorizzando i campi start_at e end_at come la data e ora di inizio appuntamento e la data e ora fine appuntamento. La creazione degli appuntamenti avviene in diverse parti del software (calendario di clinica, calendario del dottore specifico...).
Mi sono trovato nella condizione di dover aggiungere un nuovo campo alla tabella dates valorizzato con il giorno dell'appuntamento, senza orario. Ogni appuntamento viene infatti fissato con orario inizio e fine ma sempre riferiti allo stesso giorno. La definizione di un trigger mi ha permesso di indicare al dbms che in caso di inserimento nella tabella dates, il sistema debba valorizzare anche il campo date, contenente la data del campo start_at (che come spiegato sarebbe potuto essere anche end_at).

Un trigger after è invece eseguito successivamente all'esecuzione dell'operazione DML. Di conseguenza, a differenza del trigger before, non è possibile modificare il valore del record prima che avvenga l'effettiva esecuzione. E' il caso di un trigger usato per riempire una tabella di log. Un esempio di trigger after che ho implementato è stata la creazione di una nuova riga all'interno di una tabella di log degli eventi che memorizzasse il record della tabella dates prima della modifica. Di conseguenza nella tabella dates_log saranno presenti tutti gli appuntamenti prima che siano modificati, in modo da tracciare tutte le operazioni effettuate sul singolo record ai fini di controllo in caso di problematiche o di controllo.

Un indice database è una struttura di dati che migliora la velocità delle operazioni in una tabella. Ogni volta che un'applicazione web invia una query a un database, il database cercherà in tutte le righe della tua tabella per trovare quella che combacia con le clausole di ricerca. L'utilizzo degli indici è molto importante per ottenere un risultato in tempi ragionevoli. Fin quando una tabella contiene poche migliaia di righe l'utilizzo degli indici è ininfluente, le query sono talmente veloci da rendere inutile l'utilizzo di un indice, anzi, in alcuni casi potrebbe essere controproducente la creazione di indici su piccole quantità di dati: in questi casi il sistema spenderebbe più tempo nelle operazioni di aggiornamento degli indici (tramite inserimento o cancellazione record dalla tabella) rispetto al tempo guadagnato nell'utilizzo (mediante query di lettura).
Mi sono trovato a dover gestire una grossa mole di dati, parliamo di alcuni milioni di righe, che se messe in join con altre tabelle provocano il lock del database e il blocco dell'utilizzo del software da parte degli utenti. E' stata quindi una scelta obbligata la ricerca di tutte quelle query "lente" che necessitavano di indici per ottimizzare le operazioni di lettura.
La definizione degli indici è un'operazione relativamente veloce di per sé, il problema non è stato nell'esecuzione ma nella ricerca, ho dovuto dedicare diverse giornate di lavoro per capire quali richieste risultavano essere problematiche in termini di prestazioni, quali campi venivano utilizzati nella selezione dei dati e di conseguenza trovare gli attributi da usare per definire gli indici in maniera appropriata.

7.2 Esternalizzazione risorse

In PrimoUP vengono memorizzati diverse tipologie di file, a partire dalle scansioni di documenti dei pazienti, alle fotografie dei macchinari panoramici fino ad arrivare ai documenti prodotti dal sistema stesso, quali ad esempio i preventivi, i piani di cura, le fatture, i certificati medici, le ricette mediche e molto altro ancora. Tutti questi documenti vengono salvati in un'area chiamata Storage.

In un primo momento ho utilizzato come storage un disco dedicato connesso alla macchina server, in modo tale da rendere le risorse indipendenti dal software.

Con il proseguire delle attività lo spazio su disco è andato in esaurimento, ho pertanto provveduto a raddoppiare lo spazio ma era chiaro che la soluzione non era quella giusta. In primo luogo perché anche raddoppiando lo spazio questo si sarebbe prima o poi esaurito con relativa velocità. La diffusione di smartphone e tablet con fotocamere sempre più tecnologicamente avanzate, l'aumento delle risoluzioni delle immagini scattate e conseguentemente l'aumento delle dimensioni dei relativi file, è molto semplice saturare il disco di storage. Molto spesso infatti le immagini vengono caricate direttamente dagli smartphone dei medici connessi al software PrimoUP, utilizzando il proprio smartphone.
Un'altra motivazione che mi ha portato a rivedere la decisione di memorizzazione dello storage su filesystem locale è stata dettata dalla sicurezza in caso di danneggiamento dei dischi. Tramite operazioni di sincronizzazione e backup incrementali, differenziali e totali più volte al giorno i dati vengono salvati su dispositivi dedicati garantendone l'integrità in caso di danneggiamento. I file di storage erano anch'essi inclusi nel backup, ma raggiunta una mole di circa 200GB di dati diventava difficile il trasferimento su altri device per garantire sicurezza in caso di rottura del disco.
Per queste motivazioni ho creato un integrazione con il servizio Amazon AWS S3, con upload diretto sui server Amazon. E' stato necessario creare un piccolo software che recuperasse tutte le risorse di storage locali e le trasferisse sui server di Amazon.

7.3 Scalabilità Verticale e Orizzontale

Il primo avvio del sistema software è avvenuto utilizzando come server web una macchina dedicata, utilizzata per l'esecuzione del software applicativo, per la gestione dei dati e per la memorizzazione dello storage.
La gestione su un unico server dell'applicazione Laravel, del DBMS e dello storage rendeva il sistema poco flessibile e adattabile al cambiamento del traffico.

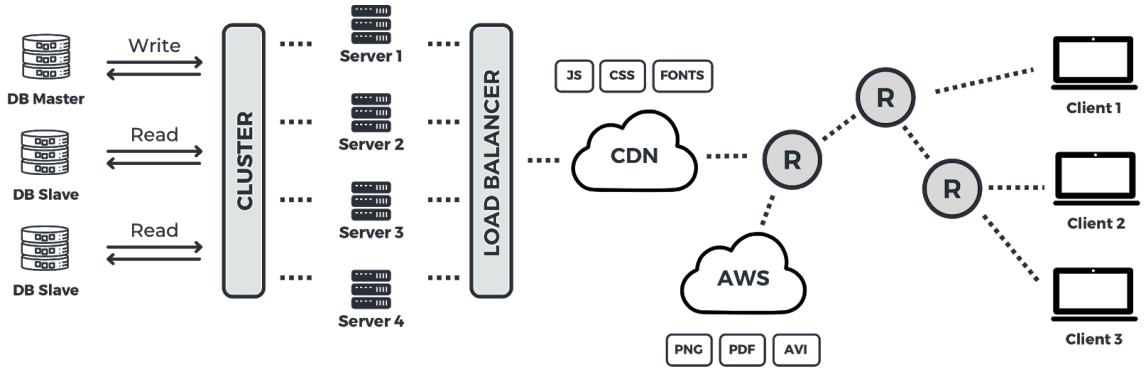
La prima modifica architetturale effettuata è stata dividere il server database dal server applicativo. Con due server distinti, uno per la memorizzazione dei dati e l'altro per la gestione delle chiamate HTTP è stato possibile assegnare prestazioni adeguate in funzione del traffico proveniente sull'applicazione.

Differenziare i sistemi su più macchine in relazione alle funzionalità che dovranno ricoprire è alla base del principio di scalabilità di un sistema informatico.
La scalabilità è la capacità di un software o di un hardware di adattarsi a un aumento di domanda o di carico di lavoro. Dunque indica se un sistema è portato a crescere o meno. Nel mondo del Web 2.0 troviamo numerosi esempi di attività scalabili, primi fra tutti i social network (Facebook, Twitter, Pinterest, ecc.), ovvero dei siti web studiati per essere espandibili a un costante aumento di traffico.
Esistono due differenti tipologie di scalabilità:

- Scalabilità verticale: significa crescere in altezza, ovvero implementare la capacità del sistema lievitando le sue risorse. Ad esempio l'aggiunta di più RAM alla propria memoria, oppure più CPU al server.
- Scalabilità orizzontale: crescere in larghezza, cioè l'aggiunta di nuovi nodi (server) in parallelo, in modo che funzionino come un sistema unico. Riguarda sia l'aspetto hardware che software, e ne sono un esempio i sistemi distribuiti e il Clustering.

La prima tipologia di scalabilità applicata al software PrimoUP è stata quella verticale. In diversi momenti ci siamo trovati a dover incrementare le risorse fornite dal provider per far fronte all'aumento delle lamentele per problemi di velocità del server. Analizzando i report di utilizzo dell'hardware è stato semplice adattare la singola risorsa: l'operazione di raddoppiare lo spazio di storage è un'operazione resa possibile dal fatto che il server utilizzabile è appunto scalabile. Attraverso l'utilizzo dell'interfaccia web per la gestione del server è stato possibile aumentare il disco fisso sul quale era mappato lo storage.
Allo stesso è stato necessario incrementare la RAM all'aumentare delle richieste di pagine web responsabili dell'esecuzione di diverse query con molti dati in output.

Un altro approccio utilizzato per rendere il sistema scalabile è stata l'adozione di politiche di scalabilità orizzontale, la prima delle quali è stata la scissione del server applicativo responsabile del software Laravel / PHP dal server DBMS responsabile dell'accesso in lettura e in scrittura del database. Non è stato però l'unico cambiamento. In considerazione del carico di CPU sempre più elevato, dovuto alla generazione dei PDF, archivi da inviare via mail, calcoli interni relativi alle provvigioni, abbiamo lavorato sulla scalabilità orizzontale, integrato nel sistema un load balancer e un database cluster.
L'immagine che segue riproduce l'architettura utilizzata per garantire la massima scalabilità: load balancer, database cluster, utilizzo di rete CDN e storage dei dati salvati su Amazon AWS S3



7.3.1 Load Balancer

Il load balancer è di fatto la distribuzione efficiente del traffico di rete in entrata su un gruppo di server back-end, noto anche come server farm o pool di server. Per far fronte a questi aumenti di carico, le best practice informatiche moderne richiedono generalmente l'aggiunta di più server.
Un load balancer funge da router, oscurando i server a cui è connesso. Funge da server web di frontend, ricevuta la richiesta dei client la inoltra ad un server, scelto utilizzando un predeterminato algoritmo di scheduling, che ne elabora la risposta. Il load balancer di fatto massimizza la velocità e l'utilizzo della risorse dei server e garantisce che nessuno di essi sia sovraccaricato, evitando che il sistema degradi per mancanza di risorse (cpu, ram...). Se un singolo server si arresta, il sistema di bilanciamento del carico reindirizza il traffico ai restanti server in linea.

Quando un nuovo server viene aggiunto al gruppo di server, il sistema di bilanciamento del carico inizia automaticamente a inviargli richieste.
Così facendo, un load balancer esegue le seguenti funzioni:

- distribuisce le richieste dei client o il carico di rete in modo efficiente su più server;
- assicura un'elevata disponibilità e affidabilità inviando richieste solo ai server online;
- fornisce la flessibilità di aggiungere o sottrarre server a seconda della domanda.

Abbiamo pertanto utilizzato un load balancer al quale abbiamo collegato due server applicativi velocizzando così le operazioni di calcolo. Gli algoritmi di scheduling utilizzati per la scelta del server applicativo a cui passare la chiamata sono configurabili in funzione delle proprie esigenze (ridizione del numero di connessioni, minimizzazione dei tempi di risposta, hash ip, random, round robin). Il sistem da me scelto è Round Robin, ovvero l'assegnazione ciclica, la scelta del "prossimo server" è sequenziale.

L'utilizzo di una catena di application servers ha però prodotto alcuni problemi da risolvere. Il primo era dovuto alle risorse presenti in un unico server. Replicare un server applicativo implica la replicazione di tutte le risorse utilizzate (immagini, documenti pdf...) con notevole spreco di tempo necessario alla sincronizzazione e risorse. Per questo motivo ho infatti migrato ogni risorsa di storage su un servizio esterno, nella fattispecie Amazon AWS S3.
Un altro problema è conosciuto con il termine "Session Persistence", e riguarda la necessità di condivisione della connessione client. Essendo HTTP un protocollo di comunicazione stateless, le informazioni relative all'utente connesso sono memorizzate sul server, identificate dall'ID di sessione. Il browser invia l'identificativo ad ogni richiesta in modo tale da rendere noto al server chi sia l'utente che ha effettuato la richiesta. Il problema nasce quando un client (browser) invia il proprio identificativo non al server che ha istanziato la connessione, bensì ad uno inconsapevole, che non conosce alcun dato associato, o ancora peggio dove l'ID di sessione si riferisce ad altri utenti. Si creerebbe pertanto un data breach, la divulgazione involontaria di

informazioni private. Fortunatamente questa problematica è facilmente risolvibile abilitando la gestione della sessione su un unico server mediante l'utilizzo del "clustering session replication".
Infine l'ultimo problema riscontrato è stata l'impossibilità di effettuare il logging delle chiamate ricevute. Ogni server NGINX ha un proprio file log per gli accessi e gli errori. Per monitorare una specifica interazione sarà necessario effettuare il merge di tutti gli access.log o gli error.log dei vari servers.

7.3.2 Database Cluster, lettura scrittura

Il database è una risorsa critica, un collo di bottiglia a cui prestare molta attenzione quando tutti gli accessi avvengono in un unico punto. Le operazioni di scrittura su database possono provocare dei blocchi sql (operazioni di lock) che rendono la risorsa inaccessibile fino al completamento dell'operazione, la chiamata successiva di lettura o scrittura dovrà attendere il termine del lock per poter essere eseguita.

Tipicamente un sistema software effettua un'intensa attività di lettura del database rispetto alle meno numerose operazioni di scrittura. Inoltre le operazioni di lettura sono molto più veloci di quelle di scrittura, responsabili degli aggiornamenti degli indici, controllo dei vincoli di integrità referenziale, esecuzione di trigger after e before.
E' pertanto evidente l'esigenza di ottimizzare le query di lettura separandole da quelle di scrittura. Per permettere ciò ci viene in aiuto il Database Cluster.
Un cluster è costituito da un gruppo di server indipendenti che interagiscono tra di loro per migliorare la disponibilità di servizi e applicazioni. I server inclusi nel cluster (detti nodi) sono connessi tra loro e si aggiornano automaticamente con processi di sincronizzazione. Se in uno dei nodi si verifica un errore, il servizio verrà garantito da un altro nodo.
Attraverso opportune configurazioni al database server, è stato possibile definire un nodo database master responsabile della scrittura su database e della propagazione delle informazioni ai server connessi e due slave, responsabili della lettura dei dati. In questo modo le operazioni di scrittura vengono effettuate su un unico punto e quelle di lettura in più server,

minimizzando pertanto i tempi di esecuzione. Automaticamente il server master effettua la sincronizzazione sui nodi slave.
Le operazioni di lettura producono un forte incremento in termini di prestazioni. Infatti, accedendo al singolo nodo slave, non sarà necessaria nessuna sincronizzazione. Per le operazioni di lettura, avendo più nodi a disposizione, è necessario utilizzare un algoritmo di scheduling per la selezione. Il sistema è attualmente impostato a funzionare con scheduling Round Robin.
Questa modifica ha portato alla creazione di un bug nel sistema. In alcune parti del software avviene una preventiva lettura per controllare se il record che si desidera inserire sia già presente a sistema o meno. Se ciò avviene per un gruppo di record l'operazione causa un problema di inconsistenza del dato. Alla prima lettura non è presente il record pertanto verrà inserito. Alla seconda lettura, se il dato fosse lo stesso del precedente la lettura su database dovrebbe produrre la riga inserita nell'iterazione precedente. Ma cosa capita quando lo slave selezionato per la lettura non si è ancora aggiornato con i cambiamenti del master? Questo problema causa la ripetizione di un record non voluto in database.
In Laravel esiste un'opzione da inserire in configurazione chiamata sticky.
L'opzione sticky è un valore opzionale che può essere utilizzato per consentire la lettura immediata dei record che sono stati scritti nel database durante il ciclo di richiesta corrente. Se l'opzione è abilitata ed è stata eseguita un'operazione di "scrittura" sul database durante il ciclo di richiesta corrente, ogni ulteriore operazione di "lettura" utilizzerà la connessione di "scrittura". Ciò garantisce che tutti i dati scritti durante il ciclo di richiesta possano essere immediatamente riletti dal database durante la stessa richiesta.

7.4 Ottimizzazione delle pagine

Il protocollo HTTP prevede lo scambio di richiesta e risposta. La richiesta tra client e server è molto ridotta in termini di byte inviati. Gran parte del delay tra l'invio della richiesta e la ricezione della risposta dipende sostanzialmente da quanto grande sarà tale risposta. Più grande sarà più byte sarà necessario inviare.

Al fine di ottenere una latenza più bassa possibile ho utilizzato tre tecniche di ottimizzazione, compressione, esternalizzazione e uso CDN.

La compressione consiste nell'invio di tutti quei file interpretabili quali html, css e javascript compressi, senza l'utilizzo dei caratteri di ritorno a capo, tabulazione, ottimizzazione delle variabili con sostituzione in nomi brevi. Questa operazione prende il nome di minification, operazione eseguita mediante l'uso di appositi software.

Con esternalizzazione viene inteso quell'operazione di utilizzo di link a risorse esterne, non presenti sul server applicativo. Analogamente alla gestione dello storage avvenuta utilizzando un servizio esterno (Amazon AWS S3), tutti i file standard javascript e css inclusi del layout di default dell'applicazione vengono caricati tramite link esterni. Questo comporta una partizione del carico tra domini differenti e conseguente aumento delle prestazioni.

Infine è stato fatto uso di una rete CDN per ottimizzare ulteriormente il software. Una rete CDN (Content Delivery Network) è una piattaforma di server altamente distribuita che aiuta a minimizzare il ritardo nel caricamento dei contenuti delle pagine web riducendo la distanza fisica tra il server e l'utente. In tal modo, gli utenti in tutto il mondo possono visualizzare gli stessi contenuti di alta qualità senza rallentare i tempi di caricamento.
In assenza di una CDN, i server di origine dei contenuti devono rispondere a ogni singola richiesta degli utenti finali, il che si traduce in un traffico notevole verso l'origine e un conseguente carico che possono aumentare le probabilità di un guasto del server di origine in caso di picchi di traffico estremamente elevati o di un carico persistente.
Rispondendo alle richieste degli utenti finali al posto dell'origine e da un'ubicazione fisica e di rete molto vicina a questi ultimi, una CDN riduce il traffico per i server di contenuti, migliorando la web experience con conseguenti vantaggi sia per il provider di contenuti che per gli utenti finali.
Il caching dei contenuti sulla rete CDN minimizza il bisogno di estrarlo dall'origine.

8 MIGRAZIONE DATI

Il software PrimoUP è stato sviluppato senza alcuna base, partendo da un così detto “blank project”, un nuovo progetto con l’unico vantaggio di partire dall’utilizzo del framework Laravel. Il cliente per il quale è stato progettato questo software esercita l’attività da diversi anni. di conseguenza ho dovuto progettare un sistema di migrazione e di conversione dei dati sul nuovo sistema.

8.1 OrisdentEvo

Circa la metà delle cliniche che attualmente utilizza il software PrimoUP arriva da OrisdentEvo, un vecchio programma stand-alone installato sulle macchine presenti in clinica.
OrisdentEvo è un software stand-alone, compatibile esclusivamente con sistemi Microsoft. Il software necessita pertanto di essere installato sulle singole macchine per poter funzionare. La gestione dei dati avviene mediante l’utilizzo di un database Microsoft MSSQL.
Il software è nato diversi anni fa. Ciò ha reso complicato l’utilizzo in smart-working. Grazie alla sua architettura, l’utilizzo è possibile all’interno di una stessa rete, installando su più postazioni il software, connesso allo stesso database di rete. L’utilizzo fuori dalla rete locale è ostico: bisognerebbe infatti abilitare alcune funzionalità di routing e comunque installare il software su ogni dispositivo sul quale si intende lavorare.
Pertanto ciò rende impossibile l’utilizzo del software su ambienti diversi da Microsoft (Mac o Linux ad esempio) e su dispositivi non desktop (tablet, smatphone...). Per ovviare a questo problema la scelta è stata quella di accedere via RDP (Desktop Remoto) su un server connesso alla rete locale e lavorare connessi ad una macchina remota.

Tra tutte queste problematiche la più importante è stata l’incompatibilità con sistemi non Microsoft. Essendo il progetto PrimoUP sviluppato in Laravel, fa uso di tecnologia web, utilizzabile con qualunque sistema operativo.

Seppur svincolato da un particolare sistema operativo, Laravel nasce per essere performante su ambienti base unix. In contrasto con ciò, il software MSSQL (utilizzato da OrisdentEvo) è difficilmente integrabile con ambienti unix. Il linguaggio PHP permette la connessione con ambienti MSSQL mediante l'utilizzo dei driver ODBC, driver con il quale ho riscontrato alcuni problemi di compatibilità con il computer utilizzato per lo sviluppo. La soluzione per ovviare a questo problema è stata l'adozione di un server Windows remoto, connesso via VPN alla rete locale dove era presente il server Sql Orisdent.
E' stato quindi creato un semplice software in PHP per effettuare una connessione ad un database specifico ed effettuare una query. Il programma attende in input i dati di connessione del database (ip, nome utente, password, porta...) e la query da eseguire. Il server web Windows, ricevuta la richiesta, si connette via ODBC al server MSSQL, esegue la query, ottiene la risposta normalizzata in json e la invia al client Laravel.
Questa comunicazione mi ha permesso di interagire con il sistema Orisdent in modo molto semplice e veloce.

L'operazione successiva è stata l'analisi del database, creando uno schema ER in grado di fornire l'idea di come fossero memorizzate le informazioni. Nonostante le numerose tabelle devo constatare la corretta progettazione di gran parte dei dati di questo software. Questo mi ha permesso di realizzare un sistema di conversione dei dati in linea con quelli progettati su PrimoUP.
Non tutte le funzionalità presenti sul vecchio software sono state riportate su PrimoUP. Il lavoro suddiviso a sprint, la metodologia Agile utilizzata, mi ha permesso di progettare un software cucito sul cliente, senza pensare a come fosse progettato il suo vecchio software. Nel corso degli sprint abbiamo trattato quelle funzionalità chiave, tralasciando alcuni programmi presenti nel vecchio software ma non utilizzato, in favore della realizzazione di nuovi programmi non presenti su Orisdent.

Il software realizzato per la conversione dei dati da Orisdent a PrimoUP è avvenuta a step, modulo per modulo, tabella per tabella, partendo ai modelli più generici privi di chiavi esterne fino ai modelli con chiavi esterne multiple, con riferimento a tabelle già acquisite (prime conversioni cliniche, medici,

pazienti e successivamente modelli ad essi connessi, come preventivi, fatture, piani di cura...).
La prima conversione ha previsto il passaggio a PrimoUP di 40 cliniche da eseguire durante due giorni di chiusura per festa nazionale. Non sarei riuscito a eseguire per tempo l'operazione se non avessi parallelizzato la conversione di più cliniche contemporaneamente. In questo modo, elaborando 5 o 6 cliniche alla volta, è stato possibile rispettare le tempistiche.

Dopo alcuni mesi, in considerazione ad una successiva fusione con un'altra catena di cliniche dentali, ho effettuato alcune modifiche al sistema di import creato per adattarlo al caso. Fortunatamente anche questa seconda azienda utilizzava il software OrisdentEvo, con alcune funzionalità in più. Ho provveduto a creare nuovi programmi per la conversione delle funzionalità mancanti, adattando anche il software PrimoUP.

In conclusione è stato un lavoro molto impegnativo e meticoloso, sia nella sua progettazione, nell'implementazione e ancor di più nell'esecuzione effettiva, operazione complessa che si è però conclusa in modo molto positivo.

8.2 DBMedica

L'ultima conversione di cui mi sono occupato è dbmedica. DBMedica è un software web, anch'esso come PrimoUP sviluppato in Laravel. Dall'analisi durante gli sprint è apparso fin da subito una conversione molto semplice, stesso tipo di database, stesso framework, funzionalità simili davano l'idea di una strada in discesa. Purtroppo la realtà dei fatti è che alcune software house si dimostrano poco collaborative. E' sempre complicato collaborare con software house per effettuare il cosiddetto passaggio di consegne: seppur scarsamente professionale, spesso non si ha interesse a dedicare tempo per un'attività in fase di cessazione.

E' importante sottolineare che l'articolo 20 del GDPR indica che "l'interessato ha il diritto di ricevere in un formato strutturato, di uso comune e leggibile da dispositivo automatico i dati personali che lo riguardano forniti a un titolare del trattamento e ha il diritto di trasmettere tali dati a un altro titolare del

trattamento senza impedimenti da parte del titolare del trattamento cui li ha forniti".
Inoltre i dati devono essere forniti "senza ingiustificato ritardo e, comunque, al più tardi entro un mese dal ricevimento della richiesta stessa" (art. 12.3 GDPR). In tal modo le persone possono spostarsi da un fornitore di servizi ad un altro, così impedendo il formarsi di fenomeni di lock-in (blocco all'interno di un servizio).
In pieno contrasto con quanto appena chiarito ho dovuto far fronte ad inconsistenti giustificazioni sull'impossibilità di ottenimento i dati richiesti.

L'unica funzionalità che ho potuto utilizzare è stata l'export in formato CSV di alcune tabelle (pazienti, medici, listini). Purtroppo, o per fortuna, ho constatato che il software in questione aveva notevoli criticità e diversi errori di configurazione ed implementazione, tralasciando alcune policy di sicurezza che Laravel mette a disposizione. Per poter estrarre i dati salvati sul sistema DBMedica, ma comunque di proprietà del cliente Primo Group, ho sfruttato una falla di sicurezza nota come Sql Injection. Tengo a precisare che la falla è stata utilizzata con l'autorizzazione della software house in questione.

SQL Injection è una tecnica di hacking molto utilizzata in ambito web, che sfrutta lacune di mancanza del controllo dei dati ricevuti dall'utente, basata sull'inserimento (injection) di codice SQL che sarà eseguito dal sistema. Spesso viene definito erroneamente come codice sql eseguito ingannando il sistema. Ritengo non sia un'affermazione corretta in quanto il sistema si limita ad eseguire ciò che viene inviato, è tuttalpiù il programmatore ad essere ingannato dall'utente.
E' un attacco reso possibile alla leggerezza commessa dai programmatori che per errore, e spesso per pigrizia, non rispettano un dogma valido in ambito web: "mai fidarsi dell'utente". Ogni input proveniente dall'utente deve infatti essere controllato e normalizzato.
Un attacco SQL injection prevede l'inserimento, da parte di un attacker, di termini e caratteri SQL specializzati nel campo di un modulo Web al fine di eseguire codice sql non previsto in fase di progettazione del software. E' una

falla molto grave, l'attacker ha infatti la possibilità di modificare o distruggere completamente i dati memorizzati nel database.

In questo esempio si può notare come un campo con un "name" usato per comporre una query possa essere utilizzato per creare una query diversa da quella attesa:

```
// codice atteso --> danilo
SELECT * FROM users WHERE name='danilo'


// codice malevolo --> danilo' or 'claudia
SELECT * FROM users WHERE name='danilo' or name='claudia
```

La falla riscontrata su DBMedica deriva dalla possibilità di effettuare un ordinamento tramite chiamate ajax. Le tabelle HTML possono essere rigenerate con l'interazione utente mediante l'utilizzo di filtri per la ricerca o per l'ordinamento. Ciò produce l'invio di una chiamata asincrona con i dati inseriti direttamente nella URL (get). Esistono molteplici modalità per poter effettuare sql injection, due delle quali sono:

- utilizzare un campo atteso nella clausola where per comporre condizioni multiple, come nell'esempio precedente;
- aggiungere una condizione impossibile aggiungendo in union la query che si vuole eseguire, terminata con il doppio meno per commentare tutto ciò che segue, come nell'esempio che segue, utilizzato per l'ottenimento dei dati ai fini della conversione.

```
// codice atteso --> 'order by name'
(SELECT * FROM users order by name' limit 10)


// codice malevolo
(SELECT * FROM users order by name) union
(select * from privileges) -- limit 10)
```

Per poter utilizzare in modo versatile e comodo il bug riscontrato ho creato una semplice funzione parametrizzata con la query da eseguire.

La query, iniettata sfruttando l'errore, viene inviata utilizzando un piccolo client web che ho generato facendo uso delle librerie cURL.

CURL è un progetto software che fornisce una libreria (libcurl) e uno strumento a riga di comando (curl) per il trasferimento di dati utilizzando vari protocolli di rete. Esistono funzioni native in PHP per poter utilizzare le libcurl.
Il browser così creato è stato utilizzato per ottenere altri dati accedendo ai software via HTTP per ottenere tutti quelle informazioni non presenti in database, estratto mediante le query custom.
E' il caso ad esempio dei dati cifrati mediante algoritmo AES, con chiave a 40bit e Initialization Vector. La software house responsabile dello sviluppo e della manutenzione di DBMedica mi ha fornito la chiave simmetrica, ma non gli IV, indispensabili per la decifratura dei messaggi, quali ad esempio codice fiscale paziente e dati delle cartelle cliniche. Per ovviare a questo ho utilizzato il browser così costituito, effettuando il parsing della pagina web per normalizzare i dati creando i record nel database PrimoUP. Questo programma prende il nome di crawler, o spider. E' una tipologia di software molto utilizzata in ambito web per analizzare i contenuti di una pagina web in maniera automatizzata.

9 INTEGRAZIONE SOFTWARE

Alcune volte nasce l'esigenza di integrare software di terze parti, un po' per impossibilità tecnologica nell'implementazione o per mancanza di competenza in un settore specifico, un po' per evitare di reinventare sistemi già creati da altri, fruibili (gratuitamente o a pagamento) e integrabili perfettamente con ogni sistema, risparmiando tempo e risorse.
I servizi integrati nel software sono serviti per inviare email, messaggi SMS e inviare ordini automatici ai fornitori di magazzino.

9.1 Transactional email

Un'email transazionale è una normale email inviata in automatico in risposta all'azione di un utente. E' un tipo di posta elettronica automatica tra un mittente e un destinatario. È diversa dall'email di marketing in quanto l'email transazionale è attivata da eventi, interazioni o preferenze all'interno di un servizio o di un'applicazione piuttosto che dalla campagna di marketing di un'azienda. Per questo motivo, l'e-mail transazionale viene talvolta chiamata e-mail "attivata".
Alcuni esempi pratici sono l'invio di un messaggio di benvenuto dopo una registrazione su un sito web, una conferma di reset della password, un auto-responder quando si è fuori dall'ufficio, una ricevuta di acquisto.
Le email transazionali hanno i tassi di apertura più alti in assoluto, proprio perché sono una risposta diretta (e attesa) ad un'azione: sempre per lo stesso motivo, anche i tassi di click-through di tali messaggi sono estremamente alti.
Pertanto, dato che le email transazionali sono così importanti per qualsiasi impresa digitale, è essenziale assicurarsi che siano correttamente consegnate ai destinatari e analizzarne le interazioni.
I due sistemi che ho integrato in PrimoUP per l'invio di questi messaggi sono mailtrap per lo sviluppo e maillgun per il sistema in produzione.

Mailtrap è un servizio web che si puoi utilizzare per "intercettare" le email, senza effettivamente inviarle al destinatario previsto. La lettura dei messaggi avviene accedendo ad un portale web, visualizzandole come in una normale

webmail. Il funzionamento è molto semplice, basta creare un account (gratuito o a pagamento in funzione di quante email si vuole gestire) per ottenere uno specifico server smtp, username e password da utilizzare per l'invio delle email. Questo servizio è utile in fase di test e di debug del software, da utilizzare esclusivamente in fase di sviluppo.

Mailgun è invece un servizio professionale per inviare email transazionali. Oltre alla classica modalità di utilizzo via SMTP, può essere utilizzato mediante API. Può essere usato per diversi scopi:

- come server di posta in uscita da integrare alla normale posta elettronica;
- come server SMTP per le email transazionali del tuo sito internet;
- come server di posta per l'invio di newsletter.

Le peculiarità che mi hanno convinto all'utilizzo di questo servizio sono diverse. In primo luogo la possibilità di controllare i log delle email inviate, verificare se l'email appena inviata sia stata consegnata al destinatario corretto o meno. Inoltre si ha la possibilità di avere una reportistica delle email aperte fornendo così un tasso di apertura utile ai fini comunicativi.

9.1 Gestione acquisto ordini

Come già descritto sommariamente nella descrizione delle funzionalità, PrimoUP prevede programmi specifici per la gestione del magazzino. Le operazioni che si andranno a svolgere durante le normali attività lavorative saranno in sequenza ciclica:

1. caricamento di un ordine in magazzino, significa aver ricevuto della merce da un fornitore, ciò aumenterà la quantità presente nella giacenza di magazzino (gestito per singola clinica);
2. scaricare singoli articoli man mano che vengono utilizzati;
3. richiedere l'ordine di articoli terminati o sottoscorta (per i quali è presente una quantità ma inferiore ad un limite preimpostato);
4. inviare l'ordine al fornitore (via email, sms o tramite sistema integrato).

L'obiettivo era quello di fornire uno strumento comune ai vari fornitori per poter gestire automaticamente il riordino della merce. Per fare ciò è stata necessaria la creazione di un sistema integrazione software e API.

Analizzando le fasi sopra descritte, al fine di migliorare il lavoro in clinica, ridurre gli errori dovuti a selezioni sbagliate di quantità, fornitori e quant'altro, si è deciso con il cliente di automatizzare il punto 4, l'inoltro degli ordini ai fornitori.

L'invio dell'ordine automatico è stata però un'operazione da effettuarsi creando programmi ad-hoc per ogni fornitore, comunicando con i loro sistemi. Le integrazioni sono state finora due, rivolte a due fornitori distinti.
Una volta ricevuto l'ordine, il fornitore elabora le richieste e nel corso delle giornate successive aggiornerà lo stato dell'ordine su PrimoUP. Per fare ciò il fornitore utilizza la comunicazione via API, con specifica chiamata di aggiornamento ordine.

I due fornitori oggetto dell'integrazione utilizzano tecnologie differenti. Il primo un ambiente PHP, il secondo SAP.
L'integrazione con PHP è stata semplice e veloce, non tanto per l'utilizzo del linguaggio in sé, quanto per la possibilità di avere un sistema già funzionante su protocollo HTTP. L'integrazione con SAP, sistema di Enterprise Resource Planning (ERP) che sostiene e automatizza la maggior parte dei processi aziendali, è stata molto complessa e farraginosa.
In entrambe le occasioni è stata necessaria un'intensa attività di collaborazione, tra comunicazioni email, telefonate, video-call, produzione documentazioni, integration test, attività necessarie per garantire un corretto successo nell'integrazione.

9.3 Gestione SMS

Un'altra importante attività di integrazione è stata l'utilizzo di un sistema di invio SMS automatico. Questo servizio è dedicato alla fidelizzazione del paziente, ad esempio tramite l'invio di SMS per il reminder degli appuntamenti prenotati o auguri di compleanno, ma anche per fini commerciali, come la notifica di sconto su preventivi non accettati.

I provider per l'invio di SMS sono molti, un po' per la semplicità di utilizzo, un po' perché già utilizzato in altri progetti, ho deciso di integrare il sistema Skebby.

Skebby opera nel settore dei servizi SMS online e si è distinta nei primi anni 2000 come una delle primissime realtà a sviluppare un'app per inviare SMS via internet dal telefonino: un servizio definito dalla stampa come lo “Skype per gli SMS”.
Da allora, la società ha sviluppato una piattaforma innovativa per la fornitura di soluzioni marketing & service via SMS accessibili sia direttamente da applicazioni e server web di terzi tramite API sia da qualunque computer tramite un'applicazione web proprietaria.

10 STATO DELL'ARTE

Attualmente il software è utilizzato in più di 150 cliniche odontoiatriche, con oltre 5.000 utenze per il personale medico, assistenti e operatori di call center. I pazienti gestiti sono circa 500.000, con oltre 700.000 piani di cura tra conclusi e da concludere.
Le cure prestate nelle cliniche sono principalmente in ambito odontoiatrico, una buona parte delle attività è coperta dalle visite specialistiche e ambulatoriali.
Raggiunti questi numeri possiamo affermare di aver concluso una fase di startup e possiamo considerare il progetto PrimoUP partito con successo. Seppur con alcune problematiche ancora in essere, il software viene utilizzato giornalmente con successo e svolge il compito prefissato: la completa sostituzione documentale e trasformazione dei processi aziendali in un sistema paperless. Questo però non significa che PrimoUP è da considerarsi concluso, sono infatti ancora tantissimi i fronti aperti. Tra sviluppo di nuove funzionalità, progettazione e integrazione nuovi sistemi e ottimizzazione della user experience il lavoro che può essere svolto è ancora tanto.

10.1 Assistenza

In considerazione dei numeri degli utilizzatori, dell'attuale crescita aziendale e dell'esigenza dei pazienti sempre più numerosi, è stata necessaria un'attività di assistenza full-time per risolvere le problematiche che si vengono a creare ogni giorno.
Come spiegato nei precedenti capitoli, il software è stato oggetto di diversi import provenienti da altri gestionali. I dati importati in alcuni casi sono frutto di ulteriori precedenti importazioni. Questo causa un'attività di monitoraggio e assistenza immediata e continua da fornire al personale di clinica.
Le figure professionali che attualmente ricoprono l'attività di assistenza su PrimoUP sono diverse e sono suddivise a livelli.

L’assistenza di primo livello è fornita da un team IT interno alla società PrimoGroup SpA, riguarda tutte le problematiche riscontrate per mancanza di conoscenza delle corrette procedure da seguire per poter ottenere il risultato richiesto. Sono quelle problematiche che scattano nel momento in cui un utente (un medico, un assistente alla poltrona, un operatore del callcenter...) non ha chiara la procedura da seguire per effettuare una determinata operazione. Alcune volte è dovuto a mancanza di conoscenza del software, delle scorciatoie o degli step per giungere al risultato voluto, queste problematiche sono facilmente risolvibili conoscendo man mano il software. Altre volte un utente si trova a dover eseguire operazioni più elaborate di quanto si aspetti. Per queste problematiche è invece necessario un aggiornamento per migliorare la user experience.

Quando la problematica non è relativa al mero funzionamento del software ma ad un errore presente nella pagina, l’assistenza passa al personale IT, sempre interno a PrimoGroup S.p.A., ma con un livello di esperienza e competenza elevato. Molto spesso sono problemi relativi a dati errati in database, un po’ a causa delle diverse operazioni di migrazioni tra i vari software, un po’ per configurazioni errate, sono spesso risolvibili modificando direttamente i dati errati presenti nel database. L’aggiornamento dei dati avviene mediante l’utilizzo di un client sql (ad esempio tramite l’applicazione SequelPro).

Alcune volte il personale IT, esperto nel settore, non riesce a risolvere la problematica. L’assistenza è affidata alla Bitboss S.r.l. che ha sviluppato il software. Ad oggi dedico gran parte della mia attività lavorativa all’assistenza sul software, risolvendo i casi più complessi e correggendo codice e dati errati.
Questo tipo di attività è destinata ad essere affidata al personale interno di PrimoGroup S.p.A. Sono infatti presenti diversi sviluppatori tra junior e senior, tra ingegneri e programmatori, e si sta formando un importante team tecnologico all’interno dell’azienda.

10.2 Sviluppi futuri

Se buona parte della mia giornata lavorativa è dedicata all'assistenza, un'altra attività di cui sono responsabile è la progettazione e lo sviluppo di nuove funzionalità, progettate come di consueto, facendo uso della metodologia Agile. Sprint dopo sprint, il software non si è mai fermato ed è in continua evoluzione.

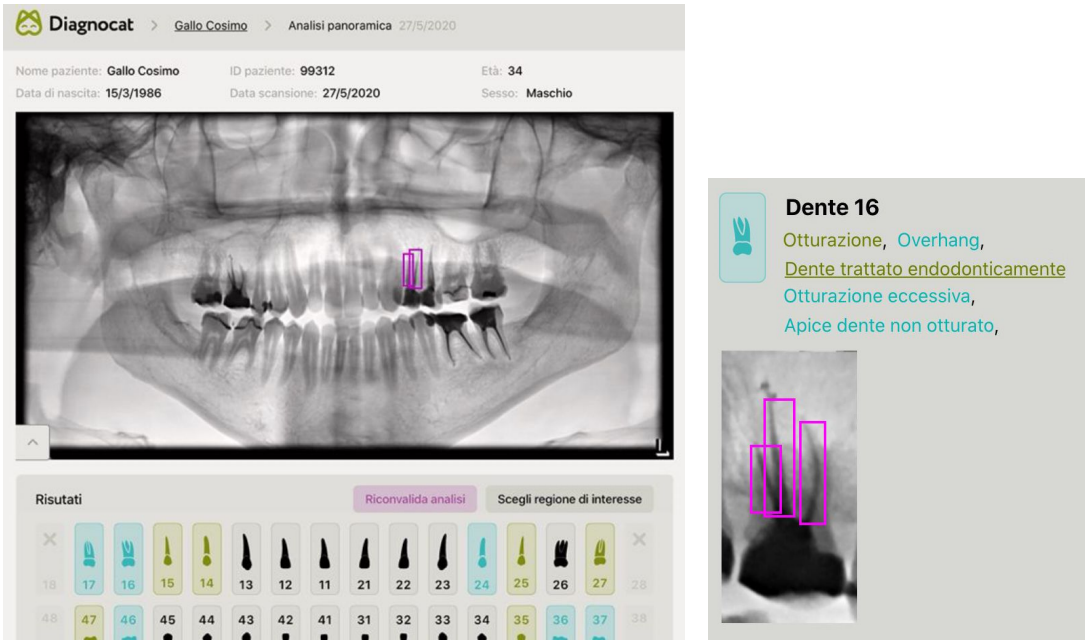
10.2.1 Call Center Nazionale

Grazie alla collaborazione con un'importante società di telecomunicazioni, è attualmente in fase di sviluppo un sistema di integrazione che consentirà di ricevere e effettuare chiamate telefoniche attraverso PrimoUP. La comunicazione avviene mediante un protocollo VOIP, utilizzando un software proprietario che permette l'interazione mediante una libreria javascript: implementando opportune callback è possibile controllare le singole chiamate.
Le chiamate potranno essere inbound (in ingresso per assistenza pazienti, informazioni...) o outbound (finalizzate alla fidelizzazione del cliente o per marketing). Gli operatori potranno lavorare in modo semplice svolgendo esclusivamente attività di evasione delle chiamate in coda, grazie ad una gestione delle chiamate automatizzate, inviate per mezzo di un configuratore. Un'altra funzionalità in fase di sviluppo è la registrazione delle chiamate e un'importante area di reportistica (chiamate effettuate, utenti in pausa, chiamate in arrivo...).

Al termine di questa integrazione, verrà attivato un servizio di assistenza nazionale, centralizzato, che permetterà di semplificare il lavoro in clinica: si pensi infatti a quanti vantaggi in termini di risparmio di tempo e risorse porterà un unico punto nazionale, in grado di aiutare il paziente che necessita spesso di semplici informazioni o assistenza. In questo modo in clinica potranno arrivare le sole chiamate che necessitano del supporto del personale medico o responsabili di clinica, ottimizzando così il lavoro e dando al paziente un servizio più efficiente.

10.2.1 Intelligenza artificiale

Attraverso l'analisi delle immagini panoramiche dei pazienti, è in corso lo sviluppo di un sistema in grado di riconoscere automaticamente le patologie dei pazienti e automatizzare la gestione dei piani di cura. L'obiettivo è da un lato fornire uno strumento di supporto ai medici, in grado di suggerire le problematiche riconosciute, dall'altro monitorare il lavoro dei medici e garantire pertanto uno standard molto elevato.



La diagnosi avviene in modalità bidirezionale: se da un lato il software è in grado di riconoscere particolari patologie automaticamente, dall'altro il medico potrà correggere il risultato, aggiungendo le cure e il software mediante auto-apprendimento modificherè l'algoritmo di ricerca.

10.2.2 IOT

Con il termine Internet Of Things (IoT, in italiano Internet delle Cose) si intendono tutti quei dispositivi hardware connessi ad Internet, utili a controllare un dispositivo fisico. L'idea di fondo è controllare un oggetto elettronico mediante comunicazione Internet. Da diversi anni a questa parte i produttori di elettrodomestici e dispositivi elettronici in genere hanno aggiunto sempre più spesso la possibilità di controllare i device attraverso siti web o applicazioni per smartphone.

Una integrazione che sarà oggetto di nuovo studio, progettazione e sviluppo è l'automazione del laboratorio.
Come accennato nei capitoli precedenti, la creazione dei manufatti avviene presso laboratori esterni alle cliniche. Attualmente il processo che parte dall'arrivo della prescrizione fino al suo completamento è gestito a livello informatico mediante software proprietari, creati ad-hoc per la gestione del processo di creazione del manufatto. L'avanzamento avviene manualmente, tramite l'aggiornamento da parte dei singoli operatori.
L'idea è quella di integrare il sistema di laboratorio con PrimoUP in modo tale che l'avanzamento della lavorazione avvenga in modo automatico, senza l'intervento dei singoli operatori.
Per permettere ciò potranno essere utilizzati piccoli dispositivi hardware connessi alle singole postazioni. Ogni dispositivo hardware collegato ai vari punti della catena di creazione, memorizzerà l'arrivo del manufatto e attenderà la lavorazione della fase per registrarne la terminazione, rendendo disponibile il manufatto per la prossima fase di lavorazione.
Così facendo sarà semplice monitorare il singolo manufatto, dando la possibilità da PrimoUP di conoscere l'esatto punto della lavorazione in cui ogni manufatto si trova.

10.3 Conclusioni

L'obiettivo del progetto PrimoUP era progettare un software in grado di semplificare i processi aziendali, facilitando quanto più possibile la gestione del lavoro da svolgere giornalmente nelle cliniche, che rendesse possibile la creazione e la conservazione di tutti i documenti trasformando l'archivio cartaceo - comprendente fatture, autorizzazioni mediche, diari clinici, cartelle cliniche - in archivio digitale.

L'utilizzo del nuovo software aveva come presupposto la conversione dei dati esistenti già presenti nei precedenti software utilizzati. Per fare ciò è stata necessaria una conoscenza molto approfondita di diversi sistemi informatici, realizzando programmi ad-hoc per recuperare i dati da un sistema operativo Windows con database Microsoft SQL e convertirli nella struttura dati relativa al nuovo software, basato su sistema operativo Linux e database Mysql. Grazie alla formazione universitaria acquisita nel corso di studi intrapreso

presso il Politecnico di Torino e grazie alla pluriennale esperienza maturata in ambito lavorativo sono stato in grado di progettare e implementare un software specifico per le operazioni di migrazione dei dati.

Un altro scoglio relativo alla progettazione del software riguardava la creazione di un sistema in grado mettere in comunicazione diversi device hardware connessi ad un client con il sistema software remoto, basato su tecnologia web. Anche in questo caso sono riuscito a giungere all'obiettivo grazie alle competenze acquisite nel corso degli studi: senza una conoscenza dettagliata di tutto ciò che riguarda la comunicazione via Internet, della struttura di ogni singolo pacchetto riguardante i protocolli ISO/OSI, non sarei riuscito a decifrare la comunicazione proprietaria tra client e dispositivi POS, a permettere lo scambio dati delle tessere sanitarie e tantomeno a creare un sistema di comunicazione con le tavolette grafiche Wacom.

Inoltre uno studio dettagliato di tecnologie di sviluppo moderne e l'utilizzo della metodologia Agile mi ha permesso di creare un sistema progettato in itinere con lo sviluppo delle varie funzionalità, realizzando un prodotto di successo utilizzato da una delle più grandi cliniche odontoiatriche italiane. Intendo pertanto concludere questo elaborato sottolineando l'importanza della formazione, prerequisito fondamentale per raggiungere gli obiettivi prefissati e che fornisce gli strumenti per acquisire capacità di problem solving, trovando una soluzione ad ogni problema dovesse insorgere in corso d'opera.

11 BIBLIOGRAFIA

- https://www.crmpartners.it/cosecrm/
- https://www.cwi.it/applicazioni-enterprise/enterprise-resource-planning-erp
- http://bias.csr.unibo.it/golfarelli/SISPEC/dispense/Studio%20di%20Fattibilit%C3%A0.pdf
- https://it.wikipedia.org/wiki/Modello_a_cascata
- https://medium.com/geekandjob-blog/scrum-cos%C3%A8-e-come-funziona-metodologia-agile-7c8988feec01
- https://hubstrat.it/metodo-agile-scrum-vantaggi-azienda/
- https://en.wikipedia.org/wiki/Scrum_(software_development)
- https://medium.com/geekandjob-blog/scrum-cos%C3%A8-e-come-funziona-metodologia-agile-7c8988feec01
- https://www.sintraconsulting.it/cosa-e-il-test-driven-development-e-come-puo-aiutarci-nei-processi-di-sviluppo/#:~:text=Il%20test%2Ddriven%20development%20(abbreviato,che%20lo%20sviluppo%20del%20software
- https://en.wikipedia.org/wiki/Test_driven_development
- https://it.wikipedia.org/wiki/Unit_testing
- https://it.wikipedia.org/wiki/Repository#:~:text=Un%20repository%20(letteralmente%20deposito%20o,prende%20il%20nome%20di%20metabase.
- https://community.pega.com/knowledgebase/articles/migrating-pega-cloud-services/stage-2-deploying-and-testing-staging-environment
- https://www.cantierecreativo.net/blog/2014-04-10-unit-e-integration-test/
- https://www.html.it/articoli/phpunit-test-di-codice-php/
- https://www.ictsecuritymagazine.com/articoli/sviluppo-sicuro-delle-applicazioni-processo/
- https://blog.netsons.com/git-software-guida-facile/#:~:text=Git%20(che%20nello%20slang%20americano,Version%20Control%20Systems%20o%20DVCS).
- https://www.perforce.com/blog/vcs/what-is-version-control
- https://www.targetweb.it/differenze-git-svn-quale-scegliere-per-collaborare-al-meglio/
- https://devdev.it/guida-gitflow/come-funziona-gitflow-branch-develop-e-master/
- https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow
- https://www.my-personaltrainer.it/salute-benessere/panoramica-dentale.html
- https://www.nginx.com/resources/glossary/load-balancing/
- https://help.skebby.it/2.0/it/topic/cos-e-skebby