UNIVERSITÀ
DI PARMA

Dipartimento di Ingegneria e Architettura

Corso di Laurea in Ingegneria dei Sistemi Informativi

Refactoring di un Software per la Prenotazione di Servizi Sanitari

Refactoring of a Software for Booking Healthcare Services

Relatore:
Prof. Michele Amoretti

Tesi di Laurea di:
Daniele Pellegrini

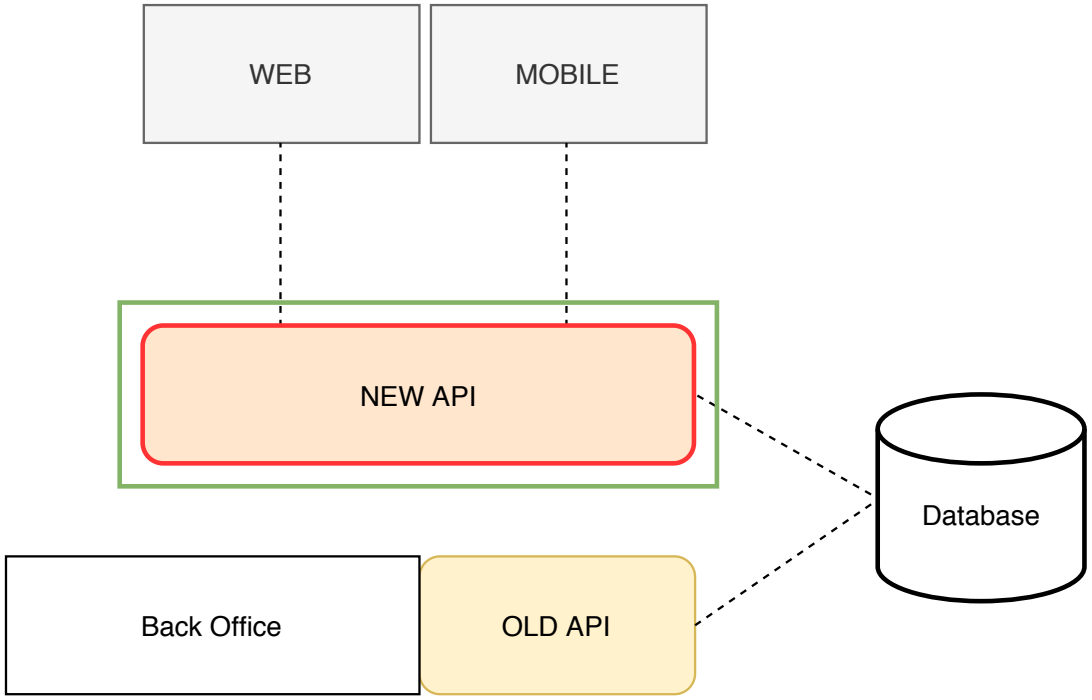
Correlatore:
Prof. Andrea Prati
Dr. Ing. Fabio Strozzi

ANNO ACCADEMICO 2019/2020

Il progetto di Tesi descritto in questo elaborato ha avuto come obiettivo la reingegnerizzazione di un software utilizzato per la prenotazione di servizi sanitari: *Zerocoda*. L'elaborato presentato è frutto dell'attività di tirocinio svolta presso l'azienda *Maps Group S.p.A* totalmente in loco, presso la sede centrale. L'azienda si occupa principamente si sviluppo software, vantando diversi clienti riconosciuti a livello nazionale.

L'applicativo permette, a seconda della struttura ospedaliera presso cui si effettuano prenotazioni, di usufruire di diverse tipologie di servizi. Il primo scopo di Zerocoda è quello di velocizzare l'accesso ai servizi, controllandone l'affluenza. I benefici derivanti dalla reingegnerizzazione che è stata effettuata non sono limitati al cliente, che in questo modo può continuare a prenotarsi nell'orario che desidera, ma riguardano anche la struttura. Essa infatti, attraverso una corretta implementazione del software, migliora il proprio sistema logistico. Avere a disposizione con largo anticipo i dati dell'utente prenotato, significa anche avere la possibilità di preparare l'occorrente della visita medica per tempo. Oltre alla riduzione del contatto interpersonale, l'ottimizzazione di software come questo può ridurre il rischio di assembramento e una migliore profilazione dell'utente con cui il personale e gli altri clienti entrano in contatto. Il COVID19 ha portato questa esigenza, che inizialmente era una comodità, ad essere a tutti gli effetti una necessità e un requisito fondamentale per le aziende che richiedono questo tipo di servizi: per la loro gestione interna e per garantire la sicurezza dei loro clienti, a maggior ragione per quelle nel settore sanitario.

Nel primo capitolo dell'elaborato sono state presentate le tecnologie utilizzate per la risoluzione del problema, giustificando le scelte fatte. Si è proceduto, nel secondo capitolo, con un'azione di reverse engineering del sistema di partenza, dove sono stati analizzati i suoi punti critici. Questo stesso capitolo ha riguardato inoltre la progettazione del nuovo sistema, in particolare qui è stata studiata una soluzione sulla base dei dati raccolti. Il terzo capitolo della tesi ha riguardato l'implementazione del nuovo sistema: qui sono stati presentati i design pattern e le tecnologie utilizzati, con riferimenti al codice per facilitarne la comprensione. Nell'ultimo capitolo della Tesi sono stati presentati degli *stress test* sul sistema di partenza e quello implementato, mettendo a confronto i risultati ottenuti. In queste fasi non si è agito direttamente sul frontend, la parte visibile dall'utente finale e con la quale interagisce, ma sul backend, la parte del sistema che elabora i dati e che ha il compito di interfacciarsi con il database, dove questi vengono salvati. Il fine ultimo di quest'opera di reingegnerizzazione è stata la *costruzione di*



un nuovo layer di REST APIs dietro un reverse proxy. Il sistema di partenza non presenta un layer di API, pertanto le chiamate dal frontend al backend vengono trasmesse in modo contorto e poco efficiente: la banalità del sistema monolitico può essere causa di un attacco informatico o malfunzionamenti, per questo si è optato per un'architettura di microservizi. Questo nuovo strato è stato pensato per essere affiancato al sistema esistente, fino a quando un successivo sviluppo ne permetterà una completa migrazione. In Figura viene mostrato come i due sistemi sono disposti a livello architetturale una volta che le nuove API sostituiranno il relativo codice sul branch di produzione. Le vecchie API rimarranno in supporto al backoffice, che sarà oggetto di una fase successiva di refactoring. Per la riscrittura del backend, precedentemente scritto in PHP, è stato utilizzato il linguaggio Java, facendo uso della JDK 15 e del framework Spring. Con la reingegnerizzazione il sistema ottiene nuove funzionalità richieste dalle strutture affiliate e migliora quelle già implementate, il tutto rimanendo coerente con il suo funzionamento: il nuovo strato inserito tra frontend e backend deve infatti garantire il corretto funzionamento di tutte le operazioni che già erano presenti.

Sulla base di *Zerocoda*, si è deciso di creare un'applicazionea analoga per i servizi commerciali: *Royalty Zerocoda*. Si tratta di un'esigenza diversa, ma vicina alla precedente. L'idea è già stata sviluppata partendo da un branch di questo software. Il layer di API realizzato costituisce quindi il primo passo sulla strada che porterà alla fusione delle due. Al completamento dell'implementazione delle nuove REST APIs seguono le successive fasi del refactoring. Queste interessano prima l'*Authentication Server*, il cui sviluppo è cominciato in parallelo a questo progetto, e, in seguito, il frontend, che subirà un leggero restyling finalizzato a migliorare il supporto alle nuove chiamate REST. Nella fase ancora successiva si procederà allo sviluppo del *Communication Server*. Zerocoda offre la possibilità ai propri utenti di essere notificati attraverso email e sms. Questi servizi sono offerti da terzi e permettono all'utente di ricevere aggiornamenti riguardo lo stato della propria prenotazione, o per operazioni di autenticazione. I server sopra citati rappresentano i nuovi microservizi introdotti dal sistema progettato. Il precedente sistema adotta un'architettura monolitica, pertanto è necessario scomporre le funzionalità che mette a disposizione e implementarle una per volta. L'ultima parte del refactoring riguarderà il backoffice, e con esso il database, dove il sistema di gestione dei dati verrà completamente rivoluzionato rispetto lo stato attuale. A questa modifica seguiranno le correzioni agli componenti architetturali a cui si è appena fatto riferimento, che sono stati implementati prima di esso.