ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA

DIPARTIMENTO DI INFORMATICA – SCIENZA E INGEGNERIA

CORSO DI LAUREA IN INGEGNERIA E SCIENZE INFORMATICHE

TITOLO DELL'ELABORATO

Booking engine online riminese - l'applicazione web moderna riprogettata da zero

Elaborato in
Systems Integration

Relatore:
Prof.re Vittorio Ghini

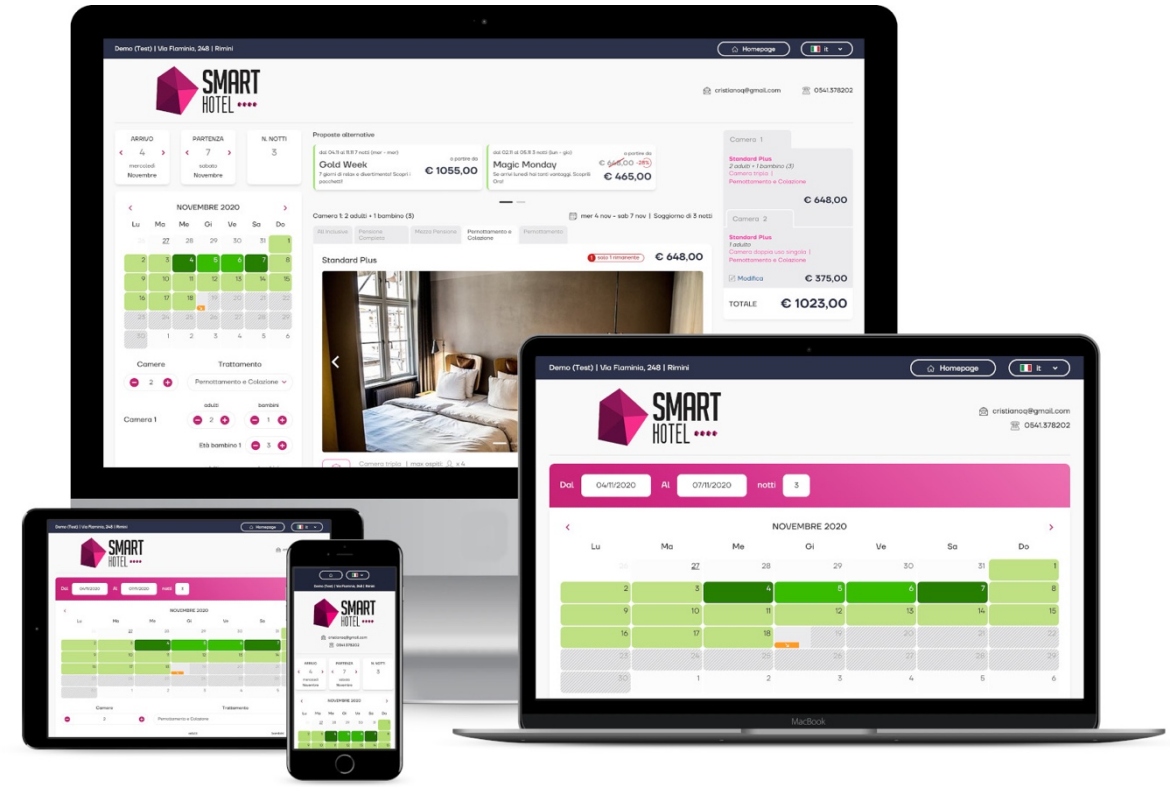
Presentata da:
Stanislav Crivoseenco

Anno Accademico 2019/2020

Alla mia famiglia

Indice

Introduzione



Booking engine online di Ipernet S.R.L.

Questo lavoro di tesi mira ad analizzare e spiegare in dettaglio tutto il percorso di riscrittura di un'applicazione web Booking engine online - motore di prenotazioni di soggiorno online. Si vuole mostrare il punto di vista di uno sviluppatore che si è occupato di una grande porzione del lavoro, svolto nell'azienda riminese Ipernet S.R.L, che dal 1994 si occupa dello sviluppo di software per l'ospitalità.

Verranno presentate le funzionalità e le caratteristiche del nuovo prodotto – già realizzato e commercializzato durante la scrittura di questo lavoro di tesi – un Booking engine competitivo che risponde alle richieste del mercato di software d'ospitalità. Verranno elencati i punti deboli della precedente versione del Booking engine e i motivi decisivi per la riscrittura dell'applicazione.

Seguiranno approfondimenti sulle scelte riguardanti l'architettura dell'applicazione e i motivi di tali scelte, basandosi principalmente sul principio, noto in campo informatico, "divide et impera", ovvero la suddivisione di un problema complesso in tanti problemi più semplici.

Infine saranno affrontati i nuovi aspetti organizzativi all'interno del team di sviluppo che si sono mostrati efficaci per la prima volta nella storia dell'attività di Ipernet.

Capitolo 1

Servizi dell'applicazione

Nel paese con il maggior numero di siti UNESCO del patrimonio mondiale, secoli di conquiste hanno lasciato in eredità un'ampia varietà linguistica, architettonica e culinaria, non deve quindi sorprendere che il mercato turistico del Belpaese contribuisca per circa il 12% del PIL nazionale con oltre 120 milioni di turisti annuali [1].

Prospetto 19.2 **Arrivi, presenze e permanenza media negli esercizi ricettivi**
Anni 2014-2018, valori assoluti in migliaia

ANNI	Arrivi		Presenze		Permanenza media
	Valori assoluti	Variazioni % sull'anno precedente	Valori assoluti	Variazioni % sull'anno precedente	
2014	106.552	2,6	377.771	0,3	3,55
2015	113.392	6,4	392.874	4,0	3,46
2016	116.944	3,2	402.962	2,6	3,45
2017	123.196	5,3	420.629	4,4	3,41
2018	128.101	4,0	428.845	2,0	3,35

Fonte: Istat, Indagine sul movimento dei clienti negli esercizi ricettivi (R)

Fonte https://www.istat.it/it/files//2019/12/C19.pdf

Il settore turistico italiano è uno dei settori industriali a più rapida crescita e a maggiore redditività. Per gestire l'elevato flusso turistico nei tempi sempre più ristretti imposti dalla digitalizzazione, gli operatori del settore hanno dovuto dotarsi di strumenti di gestione moderni, veloci ed efficaci.

Uno strumento fondamentale per gli operatori del settore turistico si chiama *Booking engine online*, ovvero un servizio di gestione computerizzata della prenotazione del soggiorno, composta da due marco componenti, *backoffice* – il gestionale riservato agli operatori della struttura ricettiva e *frontend* – oggetto dell'analisi di questa tesi - il sito web accessibile da tutti che consente a suoi visitatori di eseguire le prenotazioni.

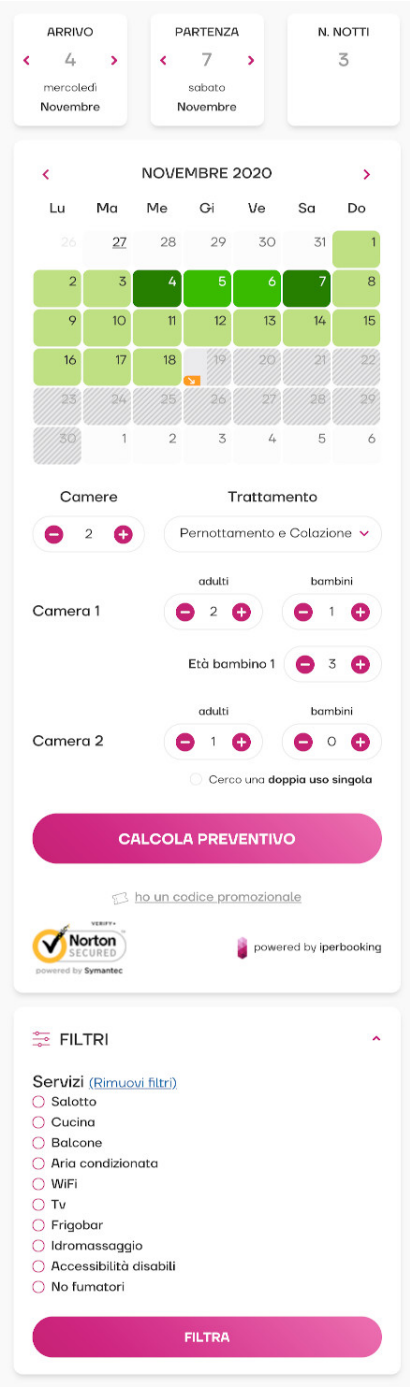
1.1 Booking engine online

L'idea del booking engine non è nuova, essa è l'evoluzione del primo servizio usato dalle compagnie aeree, nel 1946, per automatizzare la verifica di disponibilità sui voli ma il cui utilizzo era limitato agli operatori delle compagnie. Col passare del tempo i sistemi di prenotazione si sono evoluti drasticamente grazie agli sviluppi tecnologici del settore informatico, fino a permettere ai clienti di fare prenotazioni senza interventi da parte degli operatori.
Come capita spesso, il successo di un settore attrae l'interesse di altri settori con simili esigenze, tra i quali il settore di ospitalità. Questo settore ha investito molte risorse nello

sviluppo degli strumenti sempre più intuitivi, facili da usare e capaci di ridurre il costo della gestione della prenotazione.
L'azienda riminese Ipernet, operante nel mercato dei booking engine italiani, fu una delle prime nel settore a sviluppare un servizio di prenotazione esclusivamente online agli albori del web che conosciamo oggi.

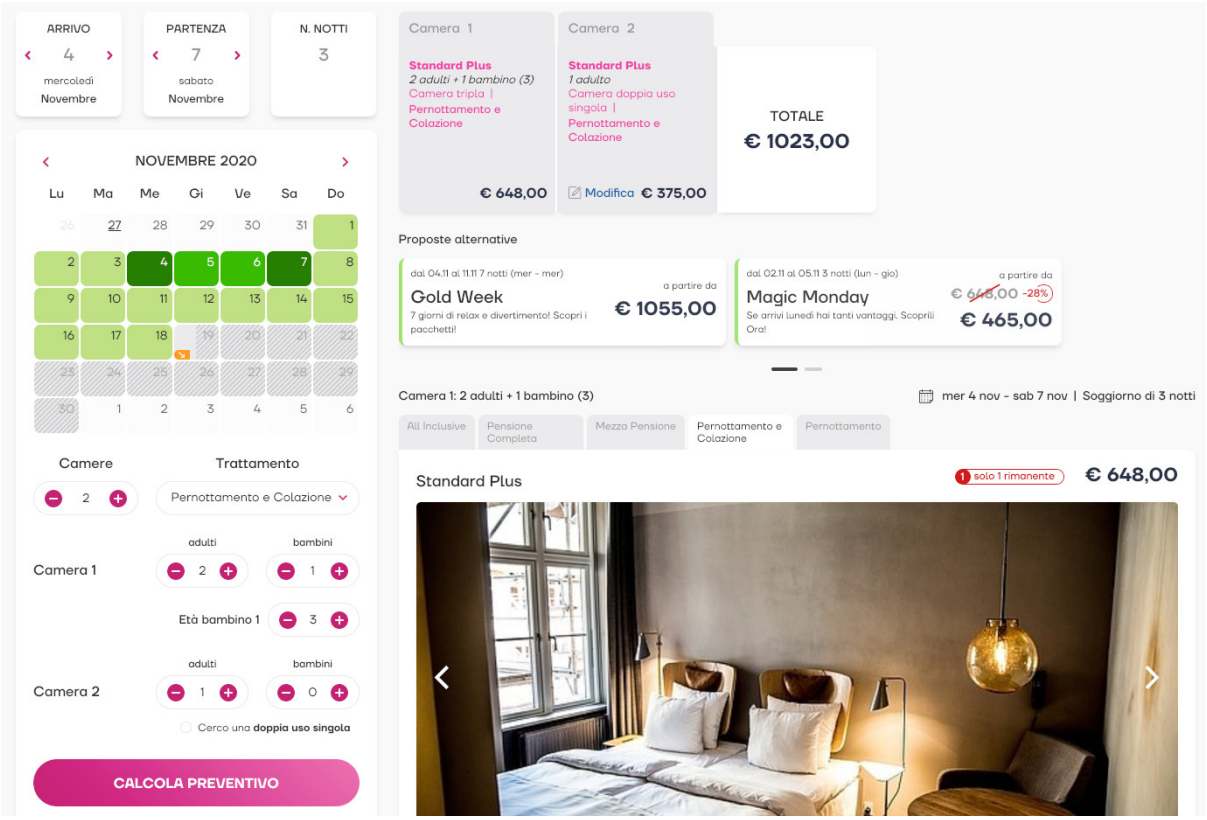
1.2 Sistema dei preventivi

Tra le funzionalità presenti nel booking engine analizzato, troviamo anche quella comune a tutti i prodotti di questo tipo: fornire risposte alle ricerche dei clienti in forma di preventivi, calcolati a partire dai dati inseriti dalla struttura ricettiva con variazioni determinate dalla specifica ricerca dal cliente. I parametri di tale ricerca dovranno essere impostati dal cliente stesso attraverso appositi componenti dell'applicativo come si vede nell'immagine sottostante.



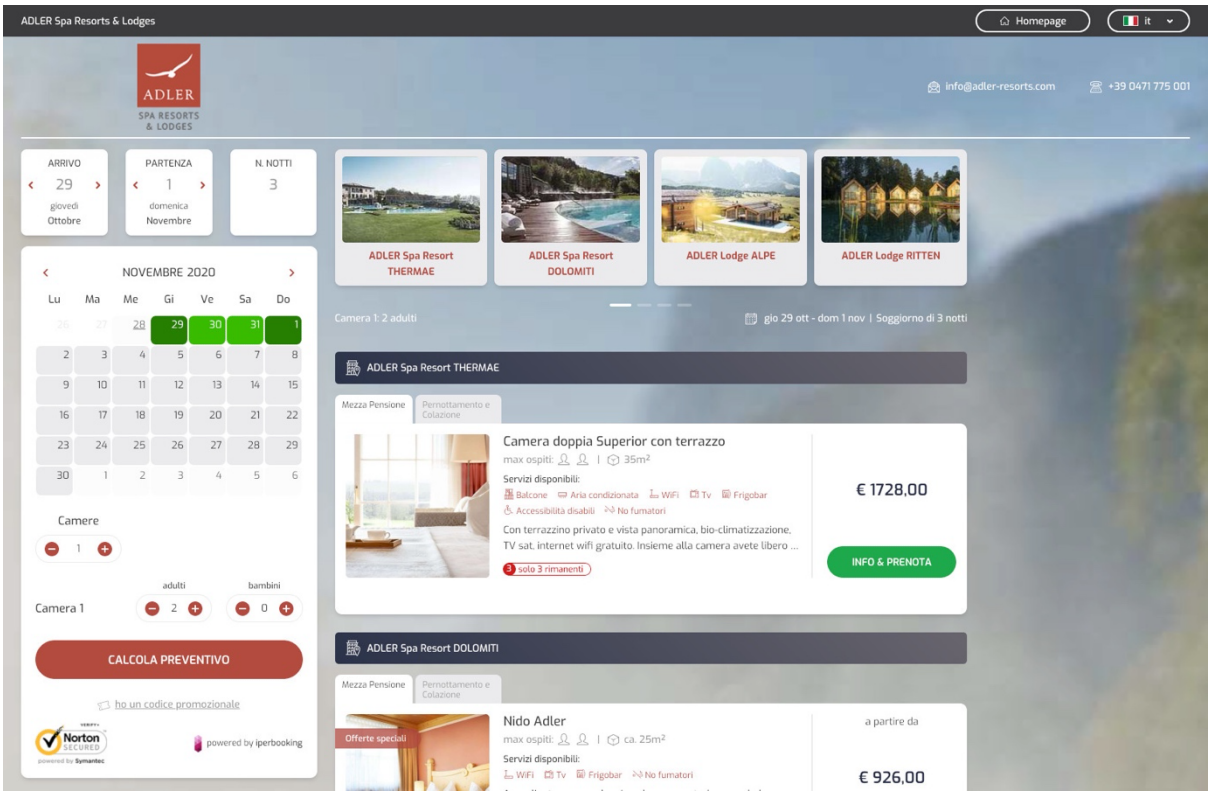
Componente di configurazione della ricerca

L’applicativo ha due interfacce web, una pubblica (frontend) per consentire ai potenziali clienti di calcolare preventivi ed eseguire prenotazioni, una privata (backoffice), accessibile solo alle strutture, per l’inserimento di tutte le informazioni utilizzate dal frontend per pubblicizzare le camere. Qualunque variazione eseguita dal backoffice sarà immediatamente presente anche nei risultati del frontend.
Per le strutture ricettive interessate, lo strumento fornisce la possibilità di attivare i *preventivi multi camera*, consente cioè ai clienti di selezionare più camere in una singola prenotazione, come succede spesso con i gruppi di turisti organizzati per le visite collettive.



Vista di un preventivo multicamera

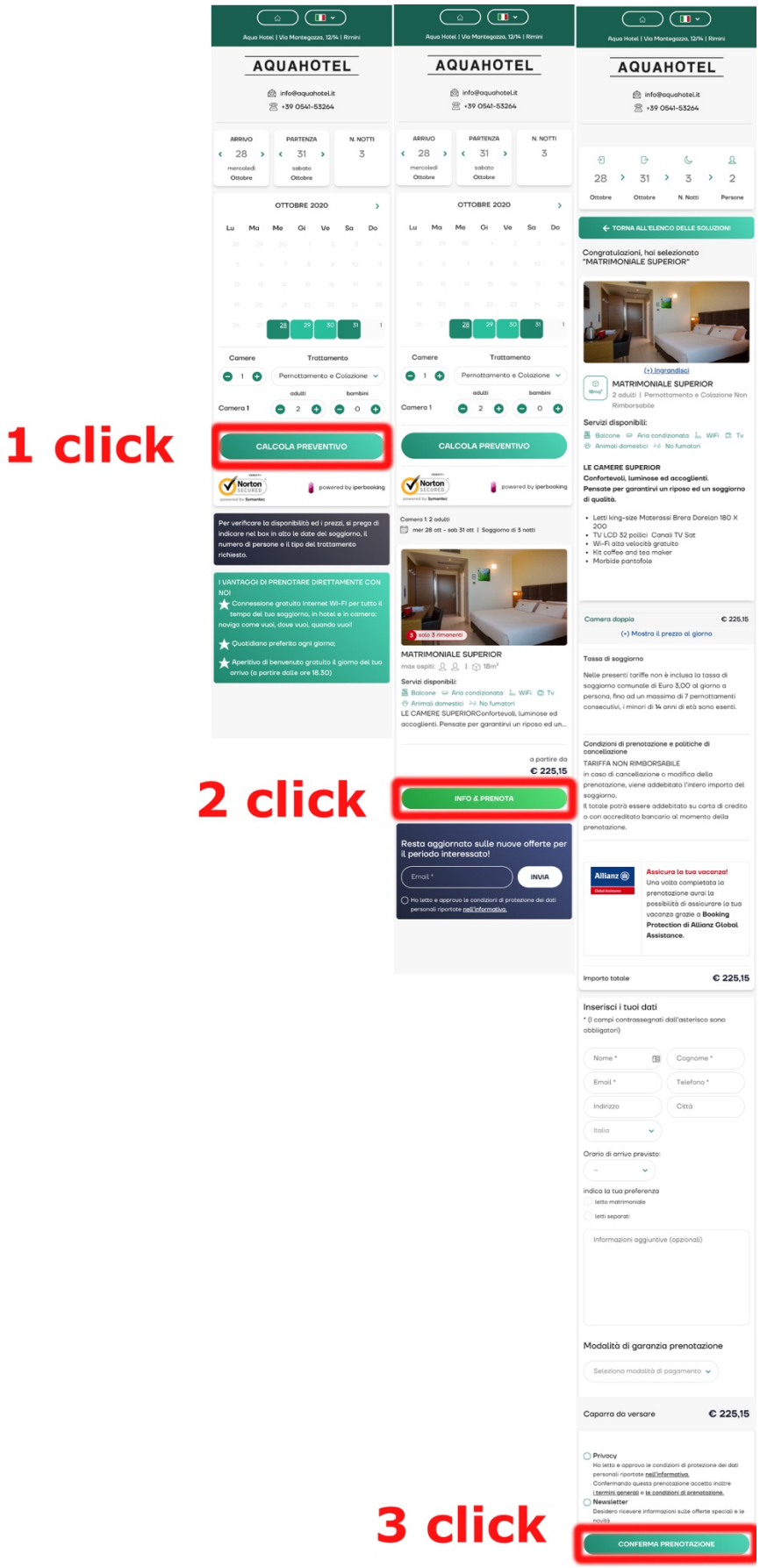
L’applicativo permette peraltro di fornire preventivi per le cosiddette *catene di strutture* che si presentano come un gruppo di strutture ricettive indipendenti. Pur rappresentando una realtà minore nel mercato turistico italiano, che coinvolge meno del 10% degli alberghi, la tendenza ad affiliarsi a catene è stata in costante crescita negli ultimi anni.[2].
Per questo motivo, nella fase di progettazione, si decise di analizzare i particolari requisiti delle catene di strutture, cercando di sviluppare un prodotto in grado di rispondere alle crescenti richieste del mercato.



Vista di una pagina dei risultati di una “catena di strutture”

1.3 Prenotazioni "3-click"

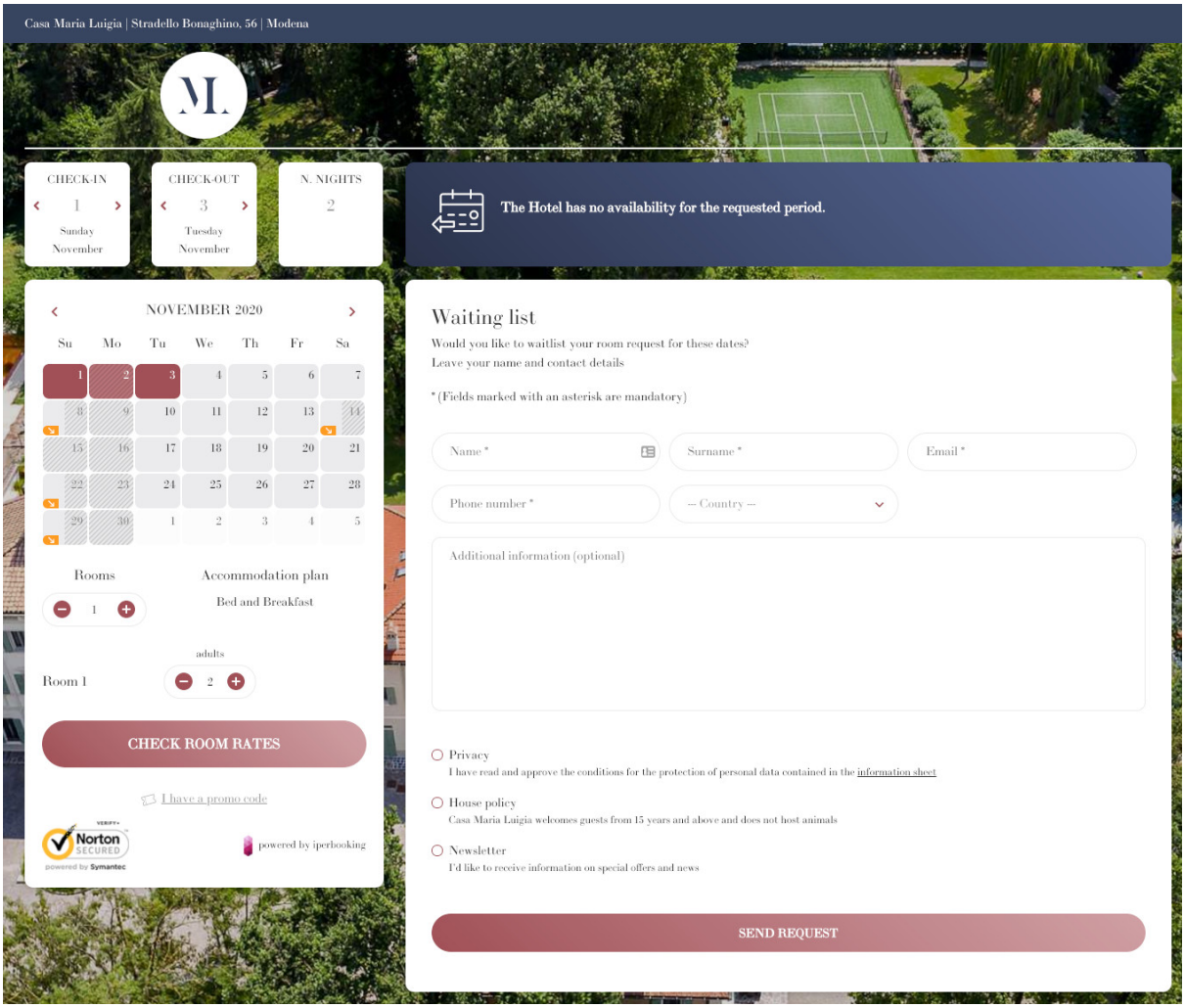
Punti di forza dell’applicativo in analisi sono la semplicità e la velocità con le quali un cliente interessato può concludere la prenotazione. Nel 2020 si dà per scontato che i sistemi d’acquisto online siano intuitivi e piacevoli da usare per fare acquisti virtuali. Di seguito si può vedere uno scenario molto comune di un cliente che desidera prenotare una camera per un periodo autunnale di 3 notti in “3 click”.



Dimostrazione di una prenotazione in 3 click

1.4 Richieste di disponibilità

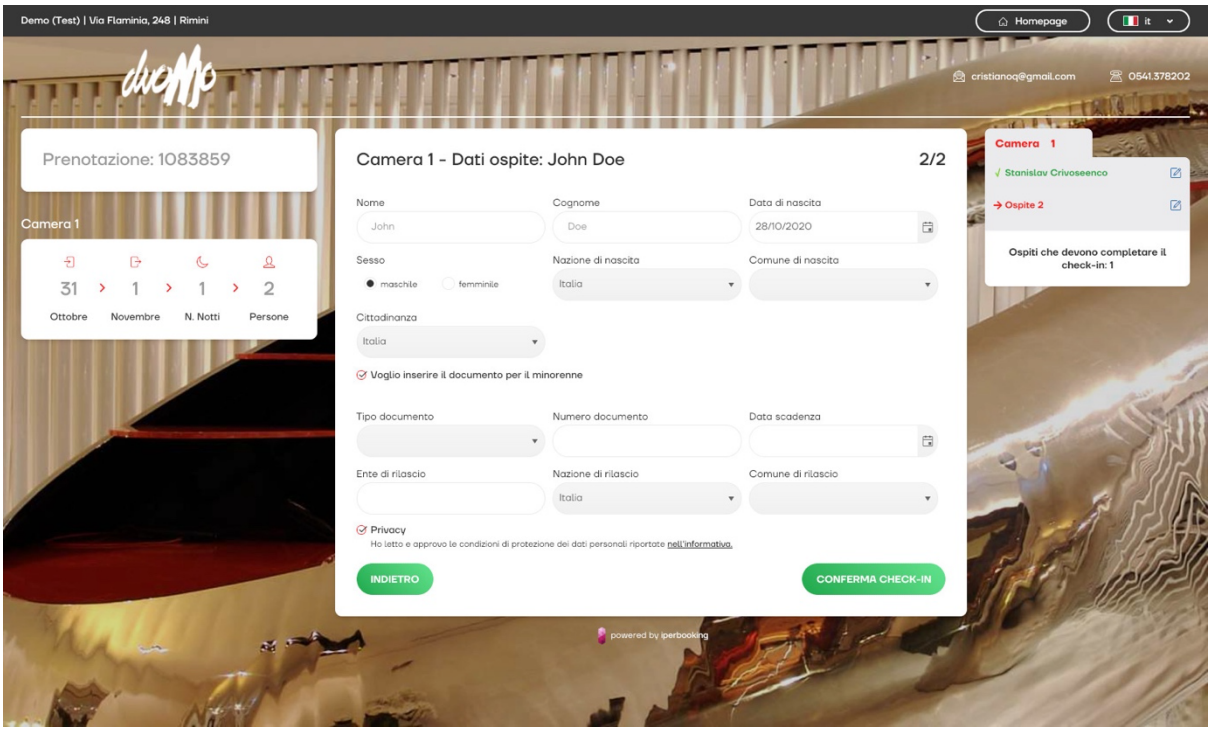
Un'altra funzionalità di grande valore per le strutture ricettive ad elevata occupazione, è dare all'eventuale cliente la possibilità di iscriversi ad una lista di attesa (waiting list) qualora la camera richiesta non fosse disponibile. Iscrivendosi il cliente informa la struttura del proprio interesse ad essere ricontattato, senza impegno di acquisto, se la camera dovesse tornare disponibile per il periodo richiesto.



Vista della pagina con il modulo della richiesta di disponibilità

1.5 Web check-in

Come già accaduto nei sistemi di prenotazione online delle compagnie aeree, anche le strutture di soggiorno hanno incontrato simili difficoltà nel trovare una soluzione per ridurre i tempi d'attesa burocratici nella fase di ricevimento e registrazione dei nuovi ospiti (check-in).
Per soddisfare questa esigenza è stato sviluppato il web check-in, o check-in online, che consente alle strutture di delegare ai clienti l'inserimento dei dati personali di ogni occupante, comodamente da casa e in qualunque momento tra la prenotazione e l'arrivo in struttura. In questo modo la struttura può ulteriormente ottimizzare le attività del personale e i clienti abbreviare le operazioni di check-in.



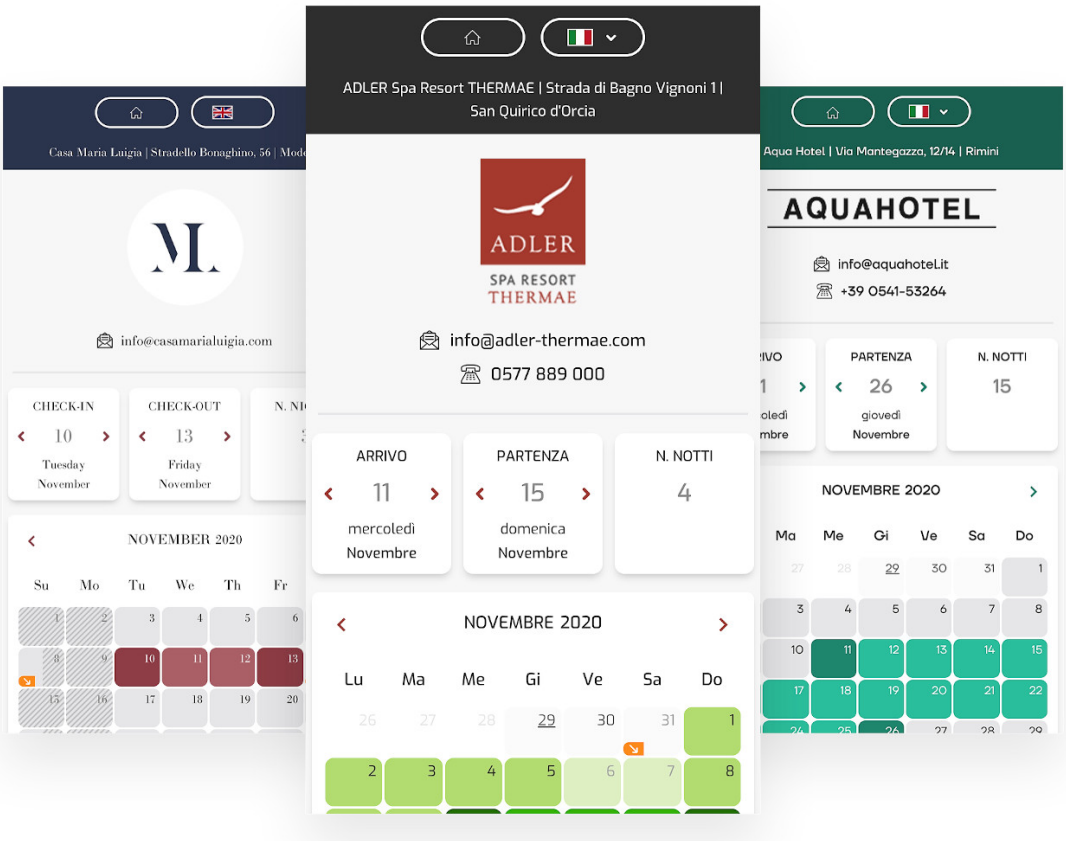
Vista della pagina con la modulistica per eseguire il check-in online

1.6 Personalizzazione grafica

Accade spesso che le strutture ricettive chiedano di eseguire adattamenti grafici all'interfaccia del booking engine, la richiesta più comune è che la grafica del frontend sia quanto più simile a quella del sito ufficiale della struttura, in modo che il cliente possa sentirsi *guidato* in un ambiente coerente privo di soluzioni di continuità. Questo aiuta a mantenere il contatto con il cliente e a creare un ambiente stimolante per concludere la prenotazione.

Il backoffice del nuovo Booking engine offre una serie di controlli grafici per dare alle strutture la possibilità di impostare i colori dei pulsanti e dei blocchi dei componenti, per scegliere se visualizzare o nascondere qualunque elemento presente nelle pagine, nonché per caricare le varie tipologie di immagini utilizzate nel frontend: sfondi, loghi, foto di camere e paesaggio, e, nel caso di catene di strutture, anche le foto delle strutture stesse.

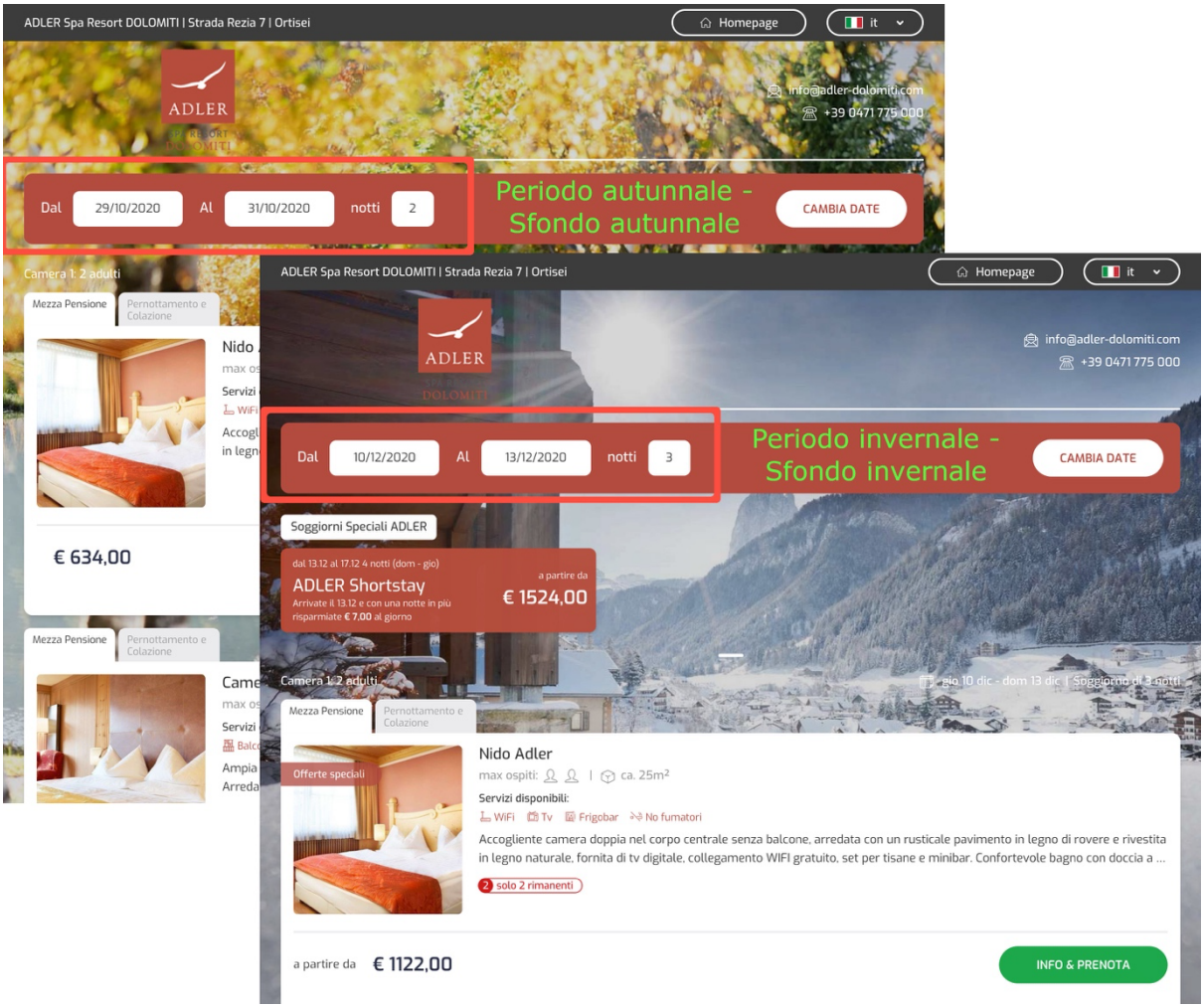
Una volta impostata la grafica, il backend cambia in automatico la versione del file dello stile CSS della struttura per il frontend in formato *"style.css?v=[numero della versione]"*. Questa operazione, chiamata in gergo tecnico *cache busting*, serve per comunicare ai browser di tutti i visitatori di scaricare la nuova versione dello stile css invece che usare la versione precedente rimasta nella memoria interna del browser, cioè la memoria cache.



Dimostrazione della grafica personalizzata per ogni cliente. Versione mobile

1.7 Personalizzazioni grafiche in base al periodo cercato

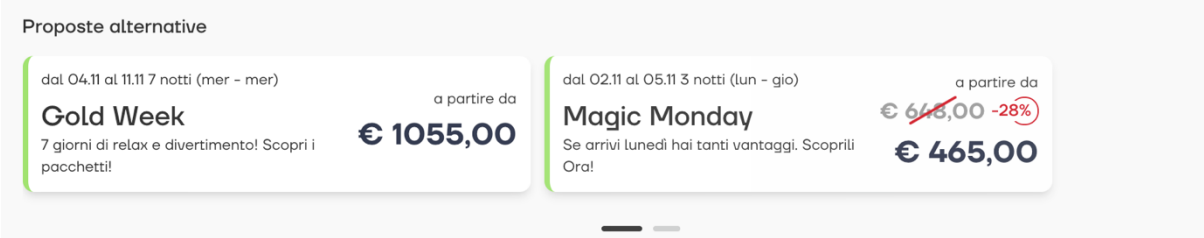
Rimanendo nell'ambito delle personalizzazioni grafiche dell'applicativo e con la volontà di migliorare ulteriormente il prodotto, si decise di voler dare la possibilità di cambiare lo sfondo della pagina web in base al periodo di soggiorno richiesto dal cliente. In questo modo, se la struttura utilizza tale funzionalità, il cliente vedrà gradualmente cambiare l'immagine di sfondo del frontend a seconda del periodo di soggiorno cercato. Per esempio, un cliente che in autunno volesse calcolare un preventivo per il periodo natalizio, vedrebbe gradualmente cambiare lo sfondo della pagina con l'immagine che la struttura ha scelto per pubblicizzare il periodo natalizio.



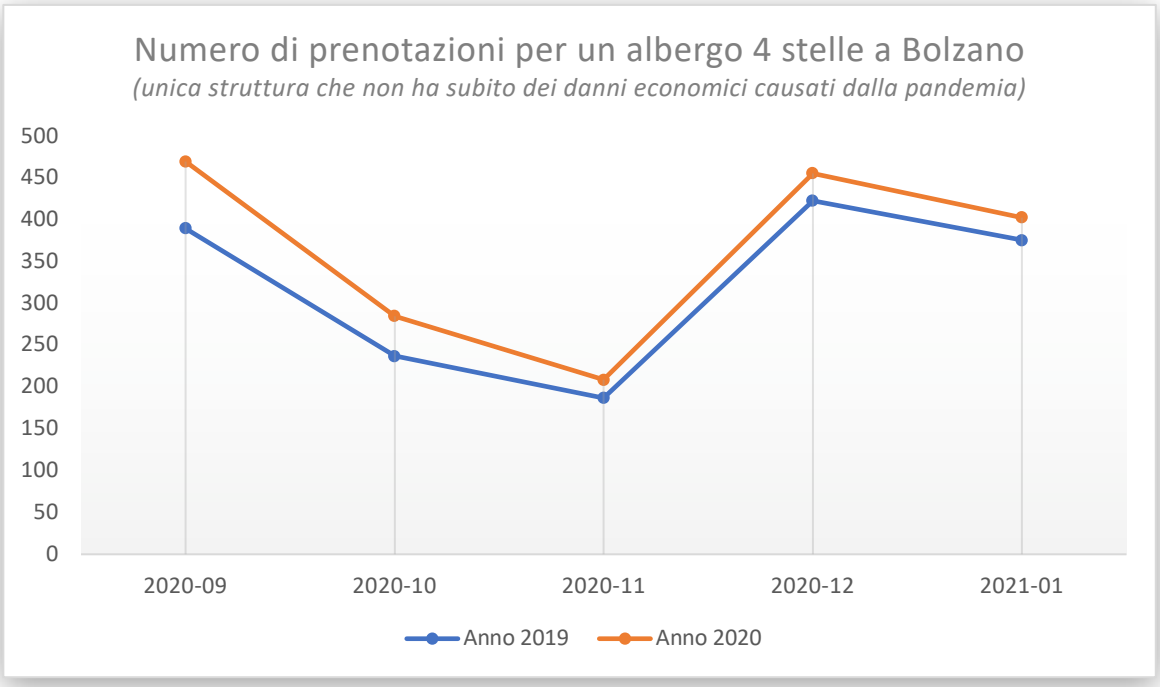
Dimostrazione del cambio sfondo in base al periodo cercato

1.8 Strategie di vendita

Negli ultimi anni, con il miglioramento delle analisi statistiche nel settore dell'ospitalià, è aumentata la possibilità, e quindi l'interesse, di proporre ai clienti prodotti simili a quelli cercati e con condizioni di vendita vantaggiose. Sono nati così i concetti di *upselling* e *cross selling*, ovvero le due strategie di marketing più note e più utilizzate per indurre i clienti ad acquistare dei prodotti complementari o simili a quelli a cui sono già interessati.
Affinché il cliente segua una delle strategie è stato sviluppato un componente dell'applicativo denominato *Proposte alternative*, che ha lo scopo di attirare l'attenzione del cliente con dettagli di soggiorni economicamente vantaggiosi oppure comprensivi di servizi non disponibili nel periodo cercato. Questo strumento consente alle strutture ricettive di adottare differenti strategie per gestire più efficientemente l'occupazione delle camere proponendo, per esempio, soggiorni a costo inferiore in periodi di bassa occupazione.



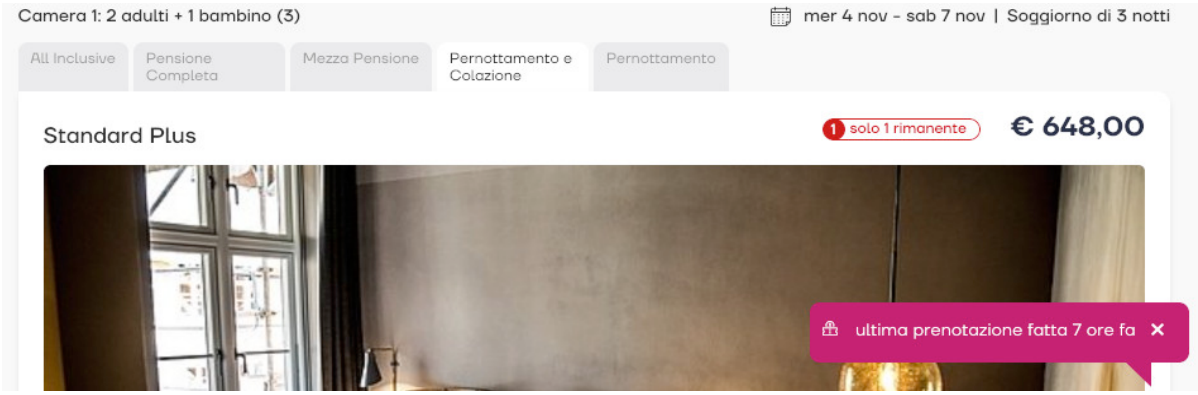
Componente "Proposte alternative"



L'aumento delle prenotazioni del 13% dalla data del rilascio delle "proposte alternative"

Un altro componente sviluppato è stato un contenitore di piccole dimensioni con messaggi come: "Ultima prenotazione fatta 7 ore fa", "Solo 1 camera rimanente" oppure "Ci sono 10

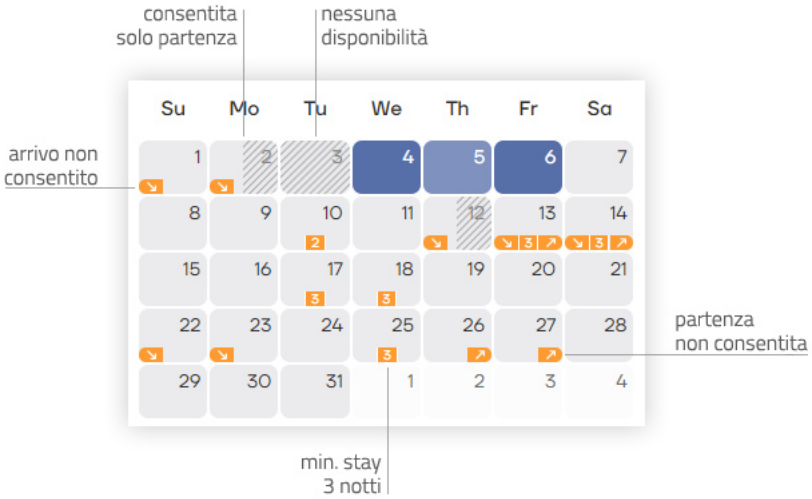
persone che guardano questa camera". È un esempio che sfrutta le debolezze psicologiche umani.



Presenza dei messaggi persuasivi

1.9 Panoramica della disponibilità istantanea della struttura ricettiva

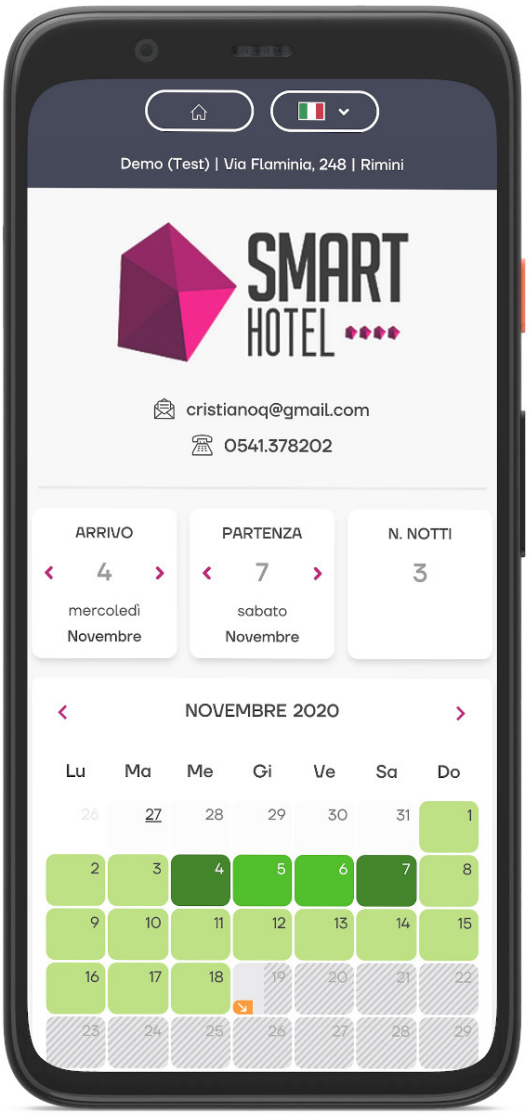
Uno degli strumenti progettati per velocizzare e facilitare la ricerca al cliente è il calendario con elementi interattivi, che comprende suggerimenti grafici e testuali caricati istantaneamente per visualizzare la disponibilità della struttura e gli eventuali vincoli per ogni giorno del mese. Questo fa sì che il cliente, una volta selezionato il mese di soggiorno desiderato, veda subito se ci sono e quanti sono gli alloggi che la struttura ricettiva è disposta a vendere.



Vista del calendario con extra informazioni

1.10 Mobile friendly

In seguito alla grande diffusione dei dispositivi mobile e al loro utilizzo per la navigazione web al posto dei personal computers, trend che è in continua crescita, Ipernet ha previsto che il booking engine potesse essere utilizzato in schermi di piccole e medie dimensioni. Essendo innegabile la convenienza che tali dispositivi offrono agli applicativi come i Booking engine, consentendo ai visitatori di controllare ed eventualmente di completare le prenotazioni in qualsiasi momento della loro vita quotidiana.
È stato dunque deciso di sviluppare l'aspetto grafico e l'interfaccia del nuovo Booking engine con una mentalità "*mobile first*", cioè con un'interfaccia progettata dapprima per i dispositivi mobili, tenendo ben presenti i nuovi limiti che essi impongono. Solo al raggiungimento di un'interfaccia mobile di base si sarebbe passati allo sviluppo dell'interfaccia per pc con le tradizionali caratteristiche di prestazioni e di dimensioni dello schermo.



Vista dell'interfaccia "Mobile friendly"

Capitolo 2

Punti di debolezza

La riscrittura di un software è sempre una decisione presa dopo un'analisi approfondita, perciò è opportuno elencare i problemi che hanno spinto alla decisione di abbandonare la manutenzione della precedente versione dell'applicativo e di riscriverlo dall'inizio in un'ottica progettuale completamente diversa.

2.1 La personalizzazione grafica rigida

Molti clienti hanno espresso critiche e malcontento per la presenza, nel frontend, di un'elevata quantità di elementi grafici non personalizzabili. Considerata la numerosità dei clienti di Ipernet e l'eterogeneità delle loro esigenze, dalla richiesta di nascondere un pulsante alla necessità di replicare la grafica del brand della struttura, si decise di investire risorse nella creazione di uno strumento flessibile e robusto che consentisse, sia agli operatori della struttura che a quelli del reparto di supporto tecnico di Ipernet, di applicare le modifiche in modo semplice a tutti gli elementi grafici.

L'idea alla base del nuovo frontend fu la seguente: "Ipernet fornisce l'applicativo con il layout di base e la grafica standard a ciascun operatore. L'operatore deve poter cambiare a proprio piacere, in qualsiasi momento ed in piena autonomia, i colori dei testi e dei componenti che lo contengono, nonché i bottoni e gli elementi di ombreggiatura". Il vantaggio più immediato ed evidente derivato da tale ragionamento fu la drastica riduzione delle richieste di supporto tecnico e, di conseguenza, la possibilità di ottimizzare le risorse aziendali indirizzandole al miglioramento costante del prodotto.

2.2 L'impossibilità di proporre le soluzioni in valute diverse

Altro punto debole, sempre più rilevante nel contesto di globalizzazione delle economie e del flusso turistico in costante crescita, era l'incapacità di mostrare ai visitatori del Booking engine i prezzi nelle valute diverse dall'euro. Tale limite riduceva il tasso di conversione da visitatore a cliente della struttura visitante, poiché molti visitatori stranieri desideravano sapere quanto avrebbero speso per le vacanze, senza perdere tempo nella ricerca di servizi di conversione della valuta dall'euro a quella di interesse.

2.3 Una fragile gestione per i testi multilingua

Restando nel contesto della globalizzazione, non si può ignorare il fatto che l'Italia sia una destinazione turistica per viaggiatori da tutto il mondo e, per far sì che le persone possano conoscere il mercato turistico italiano è indispensabile poter tradurre tutti i contenuti testuali del Booking engine in maniera semplice e rapida.
Le strutture ricettive desideravano poter gestire ogni riga di testo presente sulle pagine del Booking engine ma la vecchia versione disponeva soltanto dei testi predefiniti e soltanto in lingua italiana.
Le origini di questi limiti risalgono al fatto che la prima versione del Booking engine fu progettata pensando solo alle strutture italiane che lavorano principalmente con clientela italiana, assegnando una minore importanza alla gestione dei testi multilingua. Con l'abolizione delle frontiere interne europee, il crescere del numero di visitatori esteri e l'aumento della popolarità del Booking engine di Ipernet, si palesò l'esigenza di tradurre ogni testo in ogni lingua possibile.

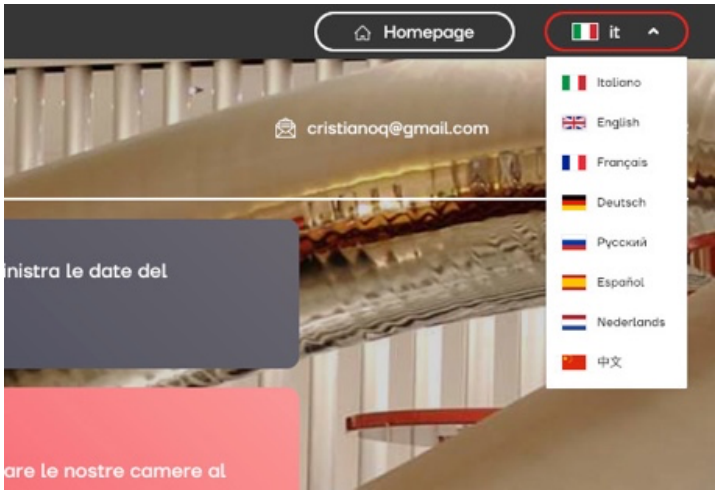
Si decise quindi di avere all'interno della pagina del frontend un oggetto *singleton* di Javascript che è una collezione di coppie di chiave-valore, come mostrato di seguito.

```
ib.dati.traduzioni
```

```
1. {Appartamento_9: "Apartment 9 beds", Appartamento_8: "Apartment 8 beds",
   comune_di_rilascio: "Issuing city",
   testoBreveAccettazionePrivacyRichiestaDisponiblita: "I have read and approve
   the conditions for the pro…="click: showModalePrivacy">information
   sheet</a>", Appartamento_7: "Apartment 7 beds", …}
    1. 1_ordinale: "1st"
    2. 2_ordinale: "2nd"
    3. 3_ordinale: "3rd"
    4. 3° Letto: "3rd Bed"
    5. 4_ordinale: "4th"
    6. 4° Letto: "4th bed"
    7. 5 Posti: "5 beds"
    8. 5_ordinale: "5th"
    9. 5° Letto: "5th bed"
    10. 6 Posti: "Sleeps 6"
    11. 6_ordinale: "6th"
    12. 6° Letto: "6th bed"
```

Oggetto di JavaScript con le prime 12 chiavi con le rispettive traduzioni in inglese

La chiave rappresenta una stringa che identifica univocamente (token) un testo traducibile, il valore associato rappresenta invece il testo tradotto nella lingua impostata dal visitatore del Booking engine, come mostrato nella sezione vista di seguito.



Il micro componente responsabile della configurazione della lingua del frontend

Proprio per la sua definizione, la struttura dati mappa di chiavi-valore è molto efficiente nell'operazione di ricerca di un elemento con il costo computazionale O(1), a differenza degli array che hanno il costo O(n) nel caso non si conosca l'esatta posizione dell'elemento cercato.

Name	Insert	Access	Search	Delete	Comments
Array	O(n)	O(1)	O(n)	O(n)	Insertion to the end is `O(1)`. Details here.
HashMap	O(1)	O(1)	O(1)	O(1)	Rehashing might affect insertion time. Details here.
Map (using Binary Search Tree)	O(log(n))	-	O(log(n))	O(log(n))	Implemented using Binary Search Tree
Set (using HashMap)	O(1)	-	O(1)	O(1)	Set using a HashMap implementation. Details here.

Confronto sul costo di operazione "ricerca" tra struttura dati del tipo Array e HashMap
Fonte https://adrianmejia.com/data-structures-time-complexity-for-beginners-arrays-hashmaps-linked-lists-stacks-queues-tutorial/ - Summary

All’interno di ogni componente dell’applicazione sarebbe stata chiamata la funzione di supporto accessibile a tutti i componenti, che accetta in ingresso la chiave da tradurre e restituisce la traduzione corrispondente.

```
// Funzione di supporto
ib.i18n.traduci = function (chiave) {
  return ib.dati.traduzioni.hasOwnProperty(chiave)
    ? ib.dati.traduzioni[ chiave ]
    : "";
};
```

Esempio della funzione di traduzione

```
// Client code
var mvvm = {
  getLabelPromozioneNonAttiva: function (e) {
    return ib.i18n.traduci("Promozione non è attiva");
  }
};
```

Esempio di come è utilizzata la funzione di traduzione

2.4 La struttura del codice in forma di "spaghetti"

Dal punto di vista degli sviluppatori di Ipernet l'implementazione dell'applicativo aveva una struttura del codice rigida e poco flessibile, era inoltre difficile comprendere quali moduli del sistema dipendessero dalla porzione di codice ispezionato. Capitava spesso che l'aggiornamento del codice dell'applicativo, sia per inserire nuove funzionalità che per correggere errori, generasse nuovi malfunzionamenti non sempre facilmente rilevabili; tipicamente a causa di un'architettura del software inconsistente e poco strutturata con la presenza di codice duplicato, ridondante, eccezioni nei flussi logici per gestire situazioni non previste nel progetto iniziale e altri ostacoli alla manutenibilità accumulatisi in anni di sviluppo.

2.5 Software poco estendibile

La versione precedente dell'applicativo aveva violato uno dei principi fondamentali nell'ingegneria del software, ovvero di prevedere la crescita ed espansione del software nel futuro. In un settore in rapida evoluzione come quello in cui opera l'azienda Ipernet, è prevedibile che sarà costante l'esigenza di aggiungere nuove funzionalità in grandi quantità e, essendo l'applicativo la principale fonte di reddito dell'azienda, questo dovrà essere progettato per poter crescere velocemente e senza compromettere le funzioni già presenti.

2.6 Errori difficilmente ritrovabili

Ai problemi prima menzionati si aggiungeva l'impegno di eseguire una delle operazioni più impegnative nello sviluppo di un software, cioè il *debugging*. Descriviamo di seguito due tra i problemi più frequenti presenti nella precedente versione del frontend:

- Memory leaks: JavaScript è uno dei linguaggi con gestione della memoria *garbage collected* (GC), questa caratteristica semplifica notevolmente il lavoro al programmatore ma non lo solleva da un utilizzo attento delle risorse. GC gestisce la memoria compiendo analisi periodiche per determinare quali porzioni siano state allocate e quali siano ancora utilizzate dall'applicazione; la memoria viene resa nuovamente disponibile solo quando non siano più presenti riferimenti ad essa. Di conseguenza, se il programmatore non è attento nella gestione delle strutture dati che occupano la memoria, si verifica il cosiddetto memory leak (letteralmente, perdita di memoria) a causa di riferimenti indesiderati alle strutture non più necessarie. Nella recedente versione del frontend questo problema è stato riscontrato molte volte, solitamente causato dall'uso scorretto degli event handlers nelle funzioni di binding e dall'allocazione di variabili senza l'utilizzo della parola chiave *"var"*, rendendole in questo modo globali invece che locali.
- L'uso scorretto delle funzioni setTimeout() e setInterval(): pur rientrando nella fattispecie del punto precedente (memory leaks), abbiamo voluto evidenziare questo caso specifico perché è stato uno dei difetti più gravi del precedente frontend.

Analizziamo il seguente frammento di codice per comprendere la gravità del problema. Dato il seguente codice:

```
// Codice dell'applicazione
// ...
// ...

var rispostaAPI = getAPIData({ idUtente: 1 }); // restituisce grande quantita' di dati
setInterval(function() {
    var node = document.getElementById('Node');
    if(node) {
      // Fare qualcosa con il nodo e la rispostaAPI
      node.innerHTML = JSON.stringify(rispostaAPI));
    }
}, 1000);

// ...
// ...

rispostaAPI = getAPIData({ idUtente: 2 }); // restituisce grande quantita' di dati

// ...
// ...

delete(rispostaAPI);
// Il programmatore si e' dimenticato di liberare l'interval

// ...
// ...
```

- *Esempio di un codice con l'effetto indesiderato*

Stiamo osservando due problemi:

1) l'oggetto HTML *"node"* potrebbe essere rimosso in futuro rendendo l'intera funzione dentro *"setInterval"* inutile. Il GC avrebbe potuto liberare la memoria, ma non riesce a farlo può fare perché *"setInterval"* non è stato deallocato dal programmatore
2) Il corollario del primo problema: nonostante il programmatore abbia liberato esplicitamente le risorse con il *"delete(rispostaAPI)"* il GC non può ancora liberare presumibilmente grandi quantità di dati salvati nella variabile *"rispostaAPI"* perché è presente un riferimento a suddetta variabile all'interno della funzione *"setInterval"*.

Grazie a questo esempio possiamo concludere che le astrazioni come GC possono essere utili strumenti d'aiuto per i programmatori ma, allo stesso tempo, cause di cali prestazionali tali da compromettere la stabilità dell'inteno software, proprio come era accaduto nel frontend precedente.

2.7 Desktop only

La grafica del Booking engine fu progettata su computer fissi per computer fissi, in un periodo storico in cui gli smartphone si stavano da poco affacciando sul mercato ma non godevano ancora di ampia diffusione; per diversi anni la postazione fissa fu l'unica presa in considerazione durante lo sviluppo e aggiornamento del frontend. Col tempo tuttavia si è creata un'ampia varietà di dispositivi portatili, capaci di navigare il web e con schermi dalle dimensioni assai variabili; tendenza questa che non sembra variata negli ultimi anni.

2.8 L'aspetto grafico obsoleto

Nell'ultimo decennio i linguaggi utilizzati per strutturare e formattare il contenuto delle pagine web, ed in particolare i CSS, hanno subito innumerevoli aggiornamenti, che abbinati ad un miglior supporto dai principali browser hanno consentito di creare e gestire elementi grafici di tale qualità e complessità, da essere paragonabili a quelli generati da software di progettazione di immagini digitali come *Adobe Photoshop*.
La grafica del precedente Booking engine non rifletteva la situazione del presente bensì rimaneva con gli elementi grafici obsoleti.
Gli strumenti utilizzati per gestire l'aspetto grafico del precedente Booking engine non erano stati aggiornati, impedendo in questo modo che l'interfaccia potesse adeguarsi alle nuove esigenze del mercato, presentandosi con elementi grafici obsoleti e inadatti ad essere utilizzati sui nuovi dispositivi.

Capitolo 3

Motivi decisivi per la riscrittura dell'applicazione

Per anni Ipernet è stata in grado di evitare la riscrittura dell'applicativo nonostante i problemi descritti nel capitolo precedente, tuttavia l'accelerazione dell'evoluzione degli strumenti adottati nel mercato dell'ospitalità ha reso tali limiti evidenti e non più economicamente tollerabili. L'aggiunta di nuove funzionalità risultava eccessivamente complessa e poco scalabile. Ora focalizzeremo la nostra attenzione sui fattori che hanno contribuito ad intraprendere la riprogettazione ex novo del Booking engine.

3.1 Esigenze del mercato

La dimensione del mercato dell'ospitalità è in costante crescita e ciò comporta una vasta offerta di booking engine alternativi. Per essere competitivi in questo mercato è necessario supportare il collegamento con diverse piattaforme di vendita, come i motori di ricerca per alloggi tra cui Booking.com, Expedia Group e HRS Group, offrendo un servizio fruibile sia sui dispositivi desktop che su tablet e mobile. Non bisogna inoltre sottovalutare la crescente richiesta, da parte delle strutture ricettive, di ottenere una presenza diretta sui metamotori di ricerca tra cui Google Hotel Ads, Trivago e TripAdvisor; in questo modo si cerca di raggiungere la massima visibilità per avere il massimo potenziale di contattare e conquistare l'attenzione dei nuovi clienti e anche nuove fonti di guadagno.

3.2 Il mondo dell'ospitalità cambia molto velocemente

I gusti e le possibilità delle persone sul come e dove soggiornare durante il viaggio sono cambiati grazie all'effetto di nuove start-up di successo, *Airbnb* è stata la prima società ad offrire una piattaforma di vendita per alloggi non convenzionali, con un'ampia varietà di scelta dal materasso in una cameretta economica al castello. La forte concorrenza dovuta al numero elevato di strutture ricettive, alla concorrenza estera e alla diversa disponibilità economica dei clienti ha portato l'intero settore ad una rivoluzione tecnologica. È sempre più necessario utilizzare strumenti evoluti per mettere al centro l'esperienza dell'ospite rispetto al solo servizio di vitto e alloggio.

3.3 Lo stile di vendita si è evoluto

Mentre fino a qualche anno fa il cliente reperiva le informazioni sulla struttura ricettiva da cataloghi cartacei e prenotava attraverso le agenzie di viaggio o al telefono, ora l'accesso alle informazioni è radicalmente cambiato. Il cliente è abituato ad ottenere facilmente e velocemente tutte le informazioni necessarie per decidere in quale struttura soggiornare, non accontentandosi più di inviare un'email e attendere ore o giorni per ricevere una risposta. Le risposte, così come le informazioni, devono essere ottenibili in tempo reale usando le chat, ci si aspetta anche di ottenere proposte personalizzate in base alle proprie esigenze. Per queste ragioni era necessario avere a disposizione un frontend con il quale ogni struttura ricettiva avesse la possibilità di applicare strategie di vendita personalizzate, di qualsiasi natura e complessità.

3.4 Difficoltà nel fare una buona impressione e guadagnare l'attenzione del cliente

È stato misurato in 3 secondi il tempo medio impiegato da una persona per decidere se abbandonare una pagina web lenta o poco reattiva per cercare altrove. L'impazienza è una delle caratteristiche più comuni tra i visitatori del Booking engine, per fare in modo che i potenziali clienti utilizzino il servizio è indispensabile mettere in opera diverse tecnologie, pratiche di ottimizzazione del software e un attento studio estetico ed ergonomico di tutte le pagine. È necessario valutare attentamente tutto il processo di vendita per fare in modo che l'ospite si senta a proprio agio e non siano presenti elementi di disturbo che possano indurre ad utilizzare un prodotto concorrente, bensì tutti gli elementi devono guidare il visitatore fino alla conclusione della prenotazione, con la volontà di ripetere in futuro l'esperienza vissuta grazie all'utilizzo del nuovo frontend.

3.5 Richieste per servizi innovativi spinte dalla pandemia di COVID19

È importante non perdere mai di vista le mutazioni del mondo reale e sapervisi adattare tempestivamente; il permanente stato d'emergenza sanitaria mondiale ha spinto le strutture ricettive ad adottare pratiche volte ad evitare il contatto fisico con gli addetti della portineria e a velocizzare le procedure burocratiche. Queste necessità hanno accresciuto l'offerta di servizi come il web check-in e l'invio di chiavi virtuali per l'apertura di camere e appartamenti. Le strutture ricettive hanno inoltre aggiunto varie sezioni informative per rassicurare l'ospite sul corretto rispetto della normativa necessaria per contenere la pandemia. Sono esigenze imprevedibili come queste che spingono a rendere il nuovo frontend sempre più modulare e resiliente alle modifiche straordinarie, modularità che purtroppo non era presente nella versione precedente del frontend.

3.6 Pressione della concorrenza

Il mercato dei Booking engine rimane in costante crescita e tale è anche la pressione della concorrenza. Il mercato italiano del software gestionale alberghiero attualmente si concentra attorno ad un grande attore del settore che, attraverso un'intensa attività di acquisizione, contribuisce sempre più alla creazione di un mercato monopolista. Un'azienda di software come Ipernet può resistere alla concorrenza grazie alla professionalità del proprio personale. Essendo un'impresa di dimensione ridotte, Ipernet cerca di offrire un servizio di qualità mantenendo un rapporto di fiducia con le realtà ricettive, fornendo un servizio di supporto preciso ed estremamente competente. Si riescono a realizzare in tempi rapidi nuove funzionalità per soddisfare le esigenze dei clienti e mantenere con essi un buon rapporto. Tuttavia, senza la possibilità di arricchire di nuove funzionalità il frontend precedente, si rischiava di perdere tale rapporto conquistato con anni di attento servizio.

3.7 I punti di forza dell'applicativo che ci rendono competitivi

I punti di forza di Ipernet consistono nell'ampio supporto dei clienti in tutti gli aspetti del servizio fornito, nell'ascoltare le esigenze particolari e nell'avviare lo sviluppo di nuove funzionalità nel Booking engine in tempi relativamente brevi, nonché nel garantire l'affidabilità e velocità del servizio. Il cliente ha un rapporto personale con chi lo segue e sa che le sue richieste vengono sempre valutate in maniera approfondita. La percentuale di gradimento del prodotto e del supporto è estremamente elevata – a dimostrazione che il cliente si senta affiancato da un partner e non abbia solo acquistato un servizio. Per questi motivi, con la riprogettazione del nuovo frontend ci si auspicava di realizzare un prodotto moderno, sia sotto l'aspetto grafico che di progettazione, capace di consolidare un alto gradimento del prodotto, di aumentare la base clienti e, di conseguenza, generare maggiori profitti.

Capitolo 4

Passaggio dall'infrastruttura on-site alle soluzioni cloud

Una strategia che permetta di astrarre il concetto di infrastruttura, la strategia cloud nota ai dipartimenti IT di innumerevoli aziende nel mondo, negli anni ha raggiunto una tale popolarità e diffusione da diventare la nuova strategia tipica del calcolo computazionale. I grandi fornitori come Amazon Web Services, Azure e Google cloud computing offrono oggi un'ampia scelta di servizi per il calcolo computazionale cloud, i clienti sono aziende di qualunque dimensione che ottengono precise garanzie sulla stabilità del servizio a prezzi modesti e proporzionali alle proprie esigenze.
Si mettono in evidenza alcune delle problematiche incontrate da Ipernet e come queste siano state risolte o mitigate adottando il cloud computing.
Sono stati migrati alcuni servizi critici su Microsoft Azure per guadagnare ridondanza e stabilità anche nel caso di eventuali problemi su uno dei server. Questo ha permesso di ottenere l'efficienza e la solidità del sistema che non sarebbe stato possibile raggiungere con l'infrastruttura on-site. Inoltre, accedendo alla piattaforma cloud, abbiamo colto l'opportunità di utilizzare alcuni strumenti critici come load balancer, database replicati e backup geo ridondati che si sono resi indispensabili per raggiungere un'elevata affidabilità del servizio; il passaggio al cloud ha permesso anche di variare dinamicamente le caratteristiche dei server in base al traffico reale durante l'anno. Non ultimo abbiamo ereditato livelli di sicurezza molto elevati garantiti dal controllo degli accessi al datacenter e possibilità di essere in conformità con le normative più stringenti come quelle del settore delle carte di pagamento PCI DSS.

Si considerano i problemi rilevati nell'infrastruttura on-site sulla quale operava il Booking engine precedente:

- Potenza di calcolo inferiore
- Costi di manutenzione
- Problemi di sicurezza del software
- Spazio fisico occupato dall'infrastruttura
- Misure da applicare in casi d'emergenza
- Connessione internet mediocre

Col passaggio del nuovo Booking engine alle soluzioni cloud, oltre a risolvere i problemi sopracitati, sono stati trovati i seguenti vantaggi:

- Scalabilità potenzialmente infinita
- Possibilità di ridimensionare le risorse in base alla reale necessità
- Accesso a strumenti evoluti (AI, monitoring)
- Soluzioni di backup e i servizi di ripristino geo ridondanti
- Pagamento delle risorse in base all'utilizzo

4.1 Gestione delle carte di credito e dei dati sensibili.

Con l'entrata in vigore della nuova normativa europea riguardante il trattamento dei dati personali e della privacy (GDPR) nel 2019, Ipernet ha dovuto affrontare i problemi derivanti dalla corretta gestione e conservazione di vari dati sensibili, come le informazioni di identificazione personali e le carte di credito. Per rispettare le principali normative, l'azienda avrebbe dovuto garantire livelli di sicurezza molto più elevati di quanto mai fatto in precedenza, venne quindi avviato un nuovo progetto denominato Gringott; utilizzare la piattaforma cloud Azure per ospitare un servizio di gestione delle carte di credito e dei dati d'identificazione. In questo modo la sicurezza dell'infrastruttura sarebbe stata garantita da Microsoft.
Tale scelta ha garantito ad Ipernet il rispetto delle normative relative alla gestione dei dati delle carte di credito secondo le specifiche dello standard PCI DSS.

4.2 Load balancer

Una nuova adozione che ha permesso di migliorare le prestazioni e la tolleranza ai guasti è il *load balancer*, è il primo elemento dell'infrastruttura a ricevere richieste HTTP dall'esterno e, in base a politiche programmabili, a decidere quale server dovrà gestire la richiesta. Grazie al potenziale offerto dalle soluzioni cloud, il load balancer può in tempi minimi incrementare la potenza di calcolo dei server Ipernet in base ai parametri impostati da essa, come nel caso di picchi di traffico alla vigilia delle festività o di significativi eventi locali, oppure ridurre le prestazioni durante la bassa stagione. Il booking engine ha cominciato ad offrire una continuità di servizio del 99,99% del tempo subito dopo la messa in opera della nuova strategia di gestione dell'infrastruttura. Anche il rilascio di nuove funzionalità ha beneficiato dell'aggiornamento, per esempio potendo limitare l'utilizzo del nuovo codice ad un numero ridotto di richieste HTTP prima di procedere con il rilascio sull'intera piattaforma.
In futuro verrà utilizzato per effettuare degli A-B Test in modo da valutare l'efficacia dell'implementazione di alcune nuove funzionalità.

4.3 Ripristino e backup più veloci ed efficaci

Altro aspetto spesso preso in considerazione solo dopo la perdita di dati cruciali per il funzionamento del servizio di booking, è il servizio di backup e ripristino di dati in casi d'emergenza, come un guasto o un errore umano. Il servizio, fornito da un altro fornitore di servizi cloud *Rackspace*, offre un ripristino dell'intero database in 30 minuti con la garanzia di aver salvato degli snapshot del DB ogni ora negli ultimi 30 giorni. Questo servizio si è dimostrato utile in diverse occasioni: dati eliminati dalle strutture ricettive per errore; errori

di programmazione o in rari casi dal supporto tecnico. Grazie alla frequenza di backup si è riscontrata un ottima capacità di recupero delle informazioni.

4.4 Analisi dei costi

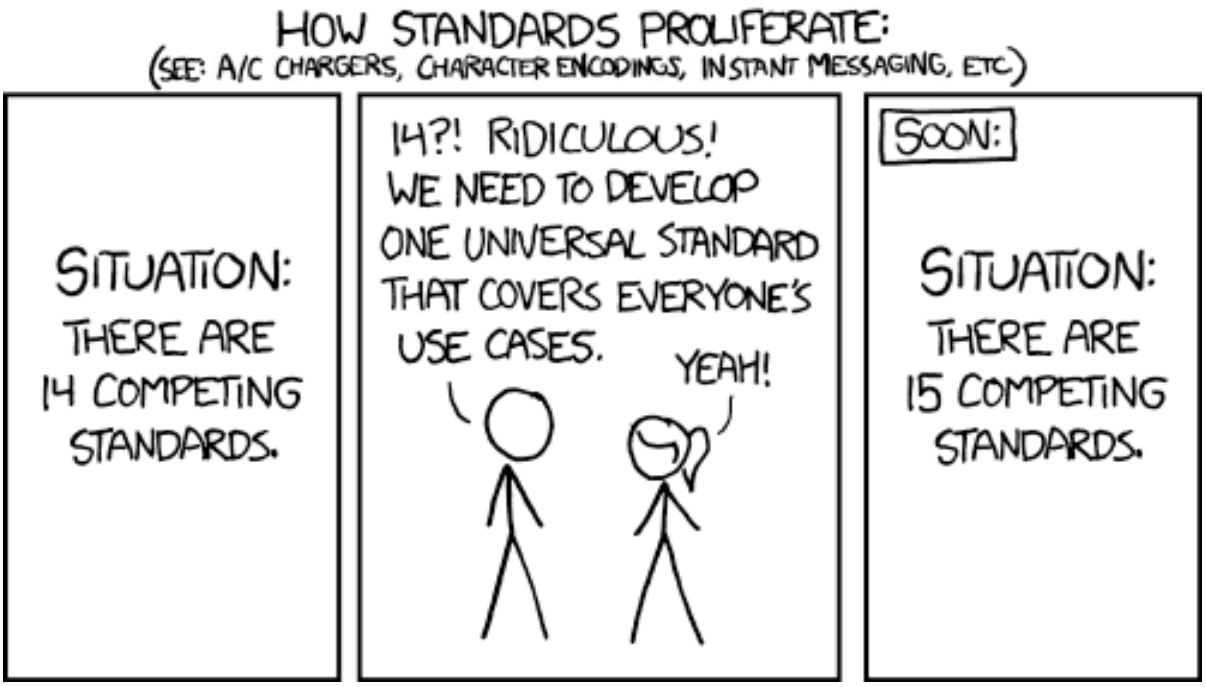
Per fornire gli stessi livelli di servizio on-site che offre Azure cloud, Ipernet avrebbe dovuto spendere il doppio rispetto alle tariffe del cloud per la sola infrastruttura e gli accertamenti legali. Oltre a questi Ipernet avrebbe speso una cifra difficilmente valutabile nella ricerca e assunzione di personale specializzato. Quello che si è capito effettuando queste valutazioni è che sia insensato spostare l'intero Booking engine dall'infrastruttura on-site al cloud senza aver prima apportato importanti cambiamenti architetturali. Perché un'applicazione possa essere performante, economica e sostenibile sul cloud è necessario utilizzare tutti gli strumenti specifici del fornitore cloud scelto.
Nel caso dell'applicazione web in analisi, un esempio di strumento cloud è l'immagazzinamento (cloud storage) dei contenuti statici come immagini, file CSS o JavaScript. Per la stessa necessità, nell'infrastruttura on-site occorre avere uno storage di grandi dimensioni, tipicamente con dischi di enorme capacità d'archiviazione per le esigenze future; occuparsi del backup con gli appositi software, costosi sia in fase di acquisto che in termini di ore di formazione e lavoro aggiuntivo per il personale; gestire una sincronizzazione tra diversi server, spesso assumendo una persona competente e ricorrendo a un necessario spegnimento del servizio per tutta la durata della sincronizzazione.
Al contrario, nel caso del cloud è sufficiente utilizzare un repository progettato appositamente per gestire questo tipo di informazioni - tra i più noti del mercato citiamo Amazon S3 e Azure File Storage - con costi estremamente contenuti, centralizzando le informazioni e avendo la possibilità di geo-replicarle per ridurre i tempi di accesso. Questo consente una notevole riduzione dei costi operativi.

Capitolo 5

Frameworks

Prendendo in considerazione il progetto della nuova versione dell'applicativo, con il termine *software framework* ci si riferisce alla struttura di base attorno alla quale avverrà il futuro sviluppo del software. Progettati a partire dall'analisi di più o meno specifiche esigenze del mercato, i framework forniscono procedure per la gestione delle attività più comuni ed una struttura del codice facilmente estensibile; grazie a queste caratteristiche lo sviluppo del software avviene più velocemente e con un minor numero di errori, permettendo di dedicare la maggior parte del tempo all'implementazione della business logic e non alla soluzione di problemi noti e per i quali vengono già fornite efficienti soluzioni. In questo capitolo si andranno ad analizzare i framework con i quali è stato realizzato il nuovo Booking engine.

5.1 Idee comuni ed errate sui framework



https://xkcd.com/927/

Nonostante numerosi dei framework esistenti si pubblicizzino come "framework universali all-in-one" e capaci di contribuire allo sviluppo di software di alta qualità, si deve evidenziare il fatto che sia difficile trovare, per ogni azienda di software, un framework robusto ed efficace nel rispondere a tutte le esigenze della loro attività.

Per non ripetere gli errori del passato anche durante lo sviluppo del nuovo Booking engine è opportuno elencare alcune idee sbagliate che idealizzano i sostenitori dei framework:

- *Aggiornamenti e manutenzioni garantiti* – il mercato del software è diventato saturo e, come accade spesso in tali condizioni, i framework servono per attrarre più sviluppatori ad adottare l'ecosistema proposto dei loro produttori. Nonostante la popolarità dei produttori, l'evidenza empirica mostra che i framework non garantiscono una struttura che possa definirsi stabile per più di qualche anno. Oltre a ciò è possibile che l'azienda sviluppatrice fallisca, lasciando il framework in uno stato di abbandono, senza alcuna manutenzione né aggiornamenti. Per un'azienda come Ipernet, l'instabilità della struttura di base venne considerato un importante fattore di rischio nel corso della progettazione.

- *I framework sono indispensabili in ogni software moderno* – parlando in termini generali è vero, tuttavia i framework sono "semplicemente" degli strumenti per risolvere certi problemi, il loro uso eccessivo, o il loro abuso, si dimostra controproducente. L'adozione di un framework genera un forte legame tra il codice dell'azienda produttrice e quello del prodotto sviluppato, alle volte costringendo a costosi interventi di manutenzione su di un prodotto perfettamente funzionante per poter utilizzare la versione aggiornata del framework.

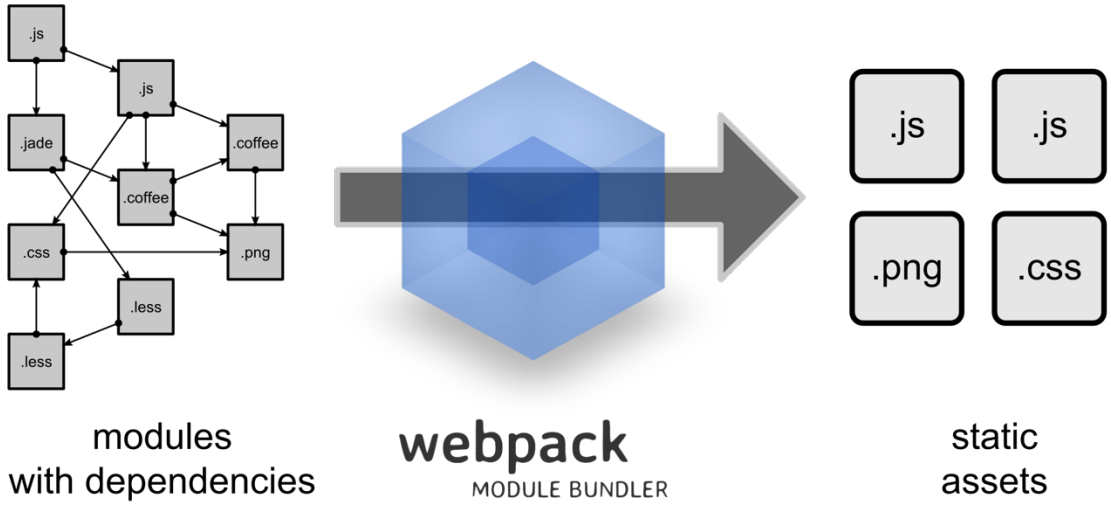
- *Sono facili da imparare e da usare* – a differenza dei framework a pagamento, la maggioranza dei framework ad uso gratuito costringono l'ingegnere del software ad utilizzare una documentazione difficilmente comprensibile, non aggiornata, poco approfondita, con esempi di codice banali e inutili per aiutare nella soluzione dei problemi complessi. Il tempo e l'impegno necessari per raggiungere la competenza nell'utilizzo di un framework sono elevati, purtroppo solo una minima parte di questa conoscenza potrà essere riutilizzata qualora si decidesse di utilizzare un altro framework.

5.2 Il motivo della scelta del tech stack

Allontanando l'attenzione dalla tecnologia usata nello sviluppo del nuovo Booking engine, si evidenzia come l'interfaccia backend non abbia avuto nel tempo la necessità di effettuare rilevanti aggiornamenti agli strumenti di sviluppo, come per esempio l'uso di un nuovo linguaggio di programmazione o di DBMS di un produttore differente. Il motivo principale per cui non si sia verificato questo cambiamento è stata la valutazione dei tempi uomo necessari per la riscrittura di tutte le funzionalità, attualmente sviluppate in Adobe ColdFusion, in un differente linguaggio. Comparando l'ambiente di sviluppo in uso con le sue possibili alternative, non sono emersi svantaggi rilevanti per quanto riguardasse le prestazioni o la sicurezza tali da giustificare un così impegnativo e costoso cambio tecnologico. Negli anni Adobe ha dimostrato di essere un partner affidabile, manutenendo ed aggiornando il prodotto, ad esempio con il rilascio tempestivo di aggiornamenti di sicurezza o di nuove release ogni anno.

La situazione è invece molto diversa nell'interfaccia frontend, in quanto in pochi anni si è assistito ad una radicale evoluzione degli strumenti e delle pratiche raccomandate per lo sviluppo sulla piattaforma web.
Dove prima l'unico strumento di sviluppo era il linguaggio di programmazione *JavaScript*, ora esiste una ampia disponibilità di strumenti complementari che estendono tale linguaggio in termini di funzionalità aggiuntive e consentono di ridurre i tempi di sviluppo. Questi, per citarne alcuni, sono: Gulp, Webpack, TypeScript, ESLinter, ES6 Modules.

Prendiamo in esame il bundler di moduli Webpack.



Scopo del Webpack

È uno strumento dall'uso consolidato nella comunità degli sviluppatori web che ha dato un particolare contributo nello sviluppo del nuovo frontend. Tra le varie funzionalità offerte da questo strumento, una di particolare importanza è stata la possibilità di strutturare l'applicazione web in moduli, definiti dallo stesso linguaggio JavaScript e, in fase di rilascio nell'ambiente di produzione, *impacchettare* tutti i moduli in unico modulo. Questa operazione ha i seguenti vantaggi:

- riduzione della complessità del codice sorgente
- chiara distinzione dei compiti svolti da ogni modulo JavaScript
- riduzione del codice duplicato grazie alla possibilità di condividere i moduli e usarli come dipendenze in altri moduli
- semplificazione dei test del codice sorgente
- riduzione delle dimensioni del codice nell'ambiente di produzione attraverso una fase di compressione e minimizzazione del codice sorgente

5.3 Frontend framework "Kendo UI"

È opportuno fare alcune osservazioni sul framework usato nel nuovo Booking engine; il team di sviluppo ha deciso all'unanimità di sviluppare il frontend basandosi su un framework, ponendo tuttavia diverse condizioni:

- deve garantire una retro compatibilità minima del 90% tra le versioni vecchie e quelle nuove
- deve offrire una *suite di widget* - piccoli componenti pronti all'uso per le esigenze più comuni (es. Dropdown list, Grid, Image uploader) - che serva a soddisfare la maggioranza delle esigenze prevedibili, deve inoltre permettere di crearne di nuovi senza compromettere la funzionalità del framework
- in caso di malfunzionamento, il supporto tecnico del produttore deve garantire assistenza entro 24 ore dalla segnalazione

Nella vasta offerta di framework presenti nel mercato come React (sviluppato da Facebook) o Angular (sviluppato da Google) è stata individuata una sola azienda attiva globalmente il cui framework soddisfi tutti i requisiti sopracitati, ovvero Progress Software Corporation Telerik. Il loro framework "Kendo UI" viene concesso in licenza a pagamento, ha una documentazione di altissimo livello e viene sviluppato con la filosofia di "consentire alle organizzazioni di creare le proprie applicazioni mission-critical pur rimanendo saldamente all'avanguardia della tecnologia per creare le applicazioni di domani" [3].

Capitolo 6

Separation of concerns and design patterns

In questo capitolo verranno analizzati alcuni concetti fondamentali dal punto di vista filosofico e architetturale che hanno permesso di produrre un ottimo risultato durante lo sviluppo del nuovo Booking engine.

6.1 “Scrivete programmi che facciano una cosa e che la facciano bene” [4]

Prendere come riferimento la citazione di Peter H. Salus e perseguire tale filosofia è stato particolarmente d’aiuto per lo sviluppo di entità autosufficienti, che siano componibili in entità più grandi e che possano comunicare tra loro indipendentemente dalla propria complessità interna. Come sostiene Peter H. Salus “scrivete programmi che gestiscano flussi di testo, perché quella è un’interfaccia universale”.
Grazie a questa osservazione si introducono alcune idee strutturali che hanno seguito tale filosofia.

6.2 Dallo “spaghetti code” al “lasagna” e “ravioli” code

Secondo la definizione di Wikipedia lo “*spaghetti code* è un termine dispregiativo per il codice sorgente di quei programmi per computer che hanno una struttura di controllo del flusso complessa e/o incomprensibile, con uso esagerato ed errato di costrutti di branching (diramazione del controllo) non strutturati ed è un esempio di anti-pattern nello sviluppo del software” [5].
In altre parole, il termine “spaghetti code” si riferisce al sorgente di un software in cui sono presenti blocchi di codice duplicato (anti pattern del DRY “dont repeat yourself”); codice strettamente accoppiato; funzioni che assolvono troppi compiti; codice difficilmente leggibile che rende il programmatore timoroso perché non riesce a valutare tutte le conseguenze della modifica; infine codice senza alcuni tipo di test automatizzato.
Di seguito possiamo vedere un tipico frammento di codice spaghetti confrontato con un frammento di codice strutturato, ovvero il codice ravioli.

```
// Spaghetti code
// ./main.js
$("#btn-login").on("click", function (element) {
  var email = $("input [name='email']").val();
  var password = $("input [name='password']").val();
  var loginForm = $("#loginForm");

  if (typeof email == "string" && email.trim() == "") {
    alert("Email è richiesto");
    return false;
  }

  if (typeof password == "string" && password.trim() == "") { // codice ripetuto
    alert("Password è richiesto");
    return false;
  }

  if (
    !(typeof email == "string" && email.trim() == "") && // codice ripetuto
    !(typeof password == "string" && password.trim() == "") // codice ripetuto
  ) {
    if (email == "emailValido@mail.it" && password == "passwordValida") {
      loginForm.submit();
    }
  }

});

```

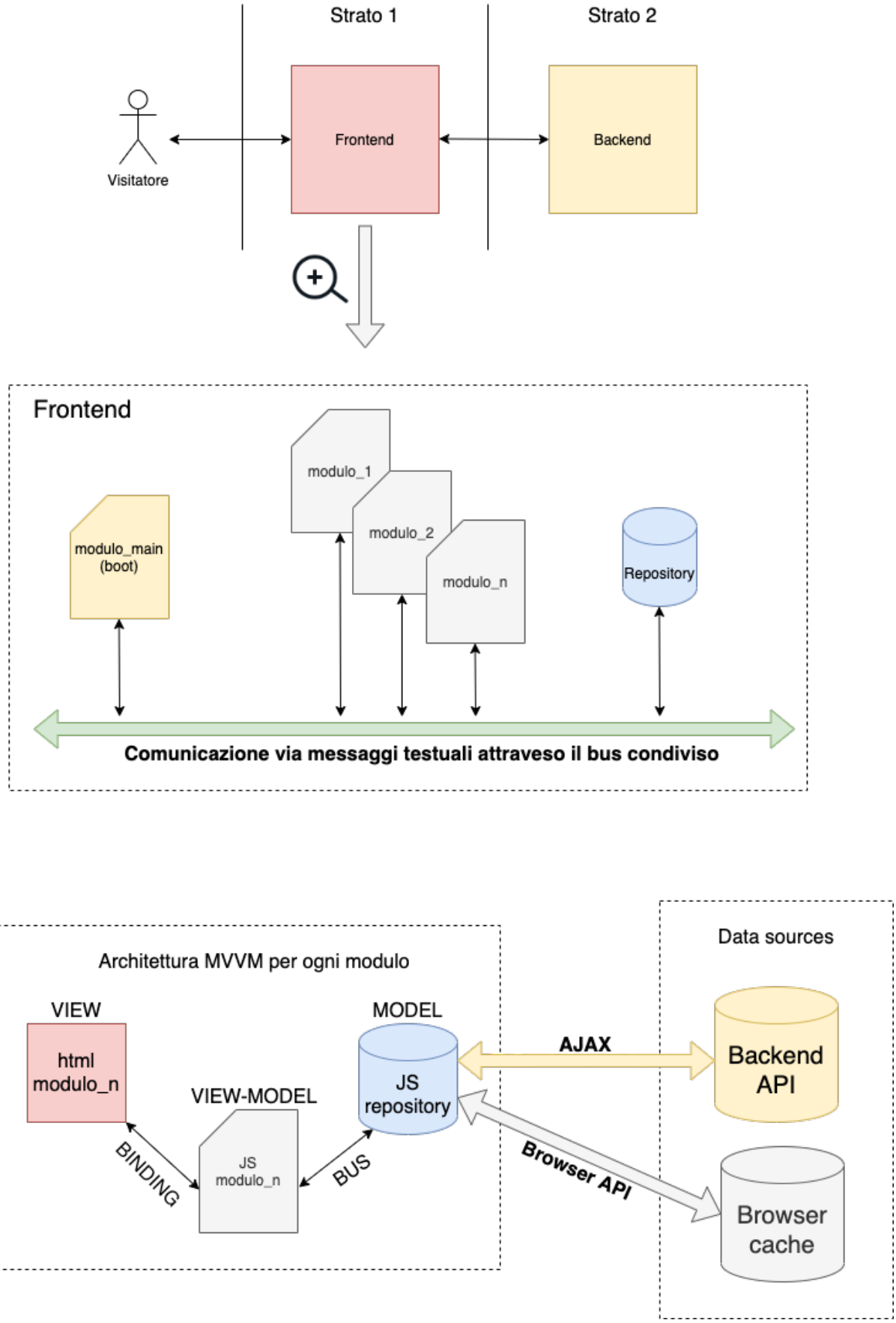
Esempio di codice spaghetti

```
// Modulare
// modulo ./isEmpty.js
function isEmpty(value) {
  return typeof value == "string" && value.trim() == "";
}
// modulo ./emailValidator.js
function emailValidator(value) {
  return !isEmpty(value) && value == "indirizzoValido@email.it";
}

// modulo ./passwordValidator.js
function passwordValidator(value) {
  return !isEmpty(value) && value == "passwordValido";
}

// ./main.js
$("#btn-login").on("click", function (element) {
  var emailValue = $("input [name='email']").val();
  var passwordValue = $("input [name='password']").val();
  var loginForm = $("#loginForm");

  if(emailValidator(emailValue)) { alert("Email non valida"); return; }
  if(passwordValidator(passwordValue)) { alert("Password non valida"); return; }

  loginForm.submit();
});
```

Esempio di codice ravioli

La versione precedente del Booking engine aveva una vasta copertura di codice spaghetti che rendeva difficoltoso eseguire molte delle modifiche necessarie per continuare lo sviluppo, portando spesso alla scrittura di codice duplicato.

Per contrastare tale situazione, nella nuova versione si è deciso innanzitutto di definire i requisiti di funzionamento del nuovo Booking engine e, conseguentemente, di separare il codice relativo agli specifici compiti in moduli di dimensioni minime, che svolgessero un solo compito e che lo svolgessero bene. Tale separazione prende il termine “lasagna code” in quanto è presente una forte distinzione tra i livelli superiori e sottostanti, metaforicamente descritti come appunto le sfoglie di pasta che compongono il piatto.
All’interno di ogni strato, ben separato dal resto del software, si potrebbe continuare ad applicare ricorsivamente lo stesso principio delle “lasagne”, ma nel caso in esame la strategia è stata applicata una sola volta. La tecnica applicata successiva ha aumentato la robustezza e la compressione del codice e prende il nome di “ravioli code”, essa adotta i principi fondamentali della programmazione ad oggetti tra i quali i più noti: incapsulamento, ereditarietà e polimorfismo.



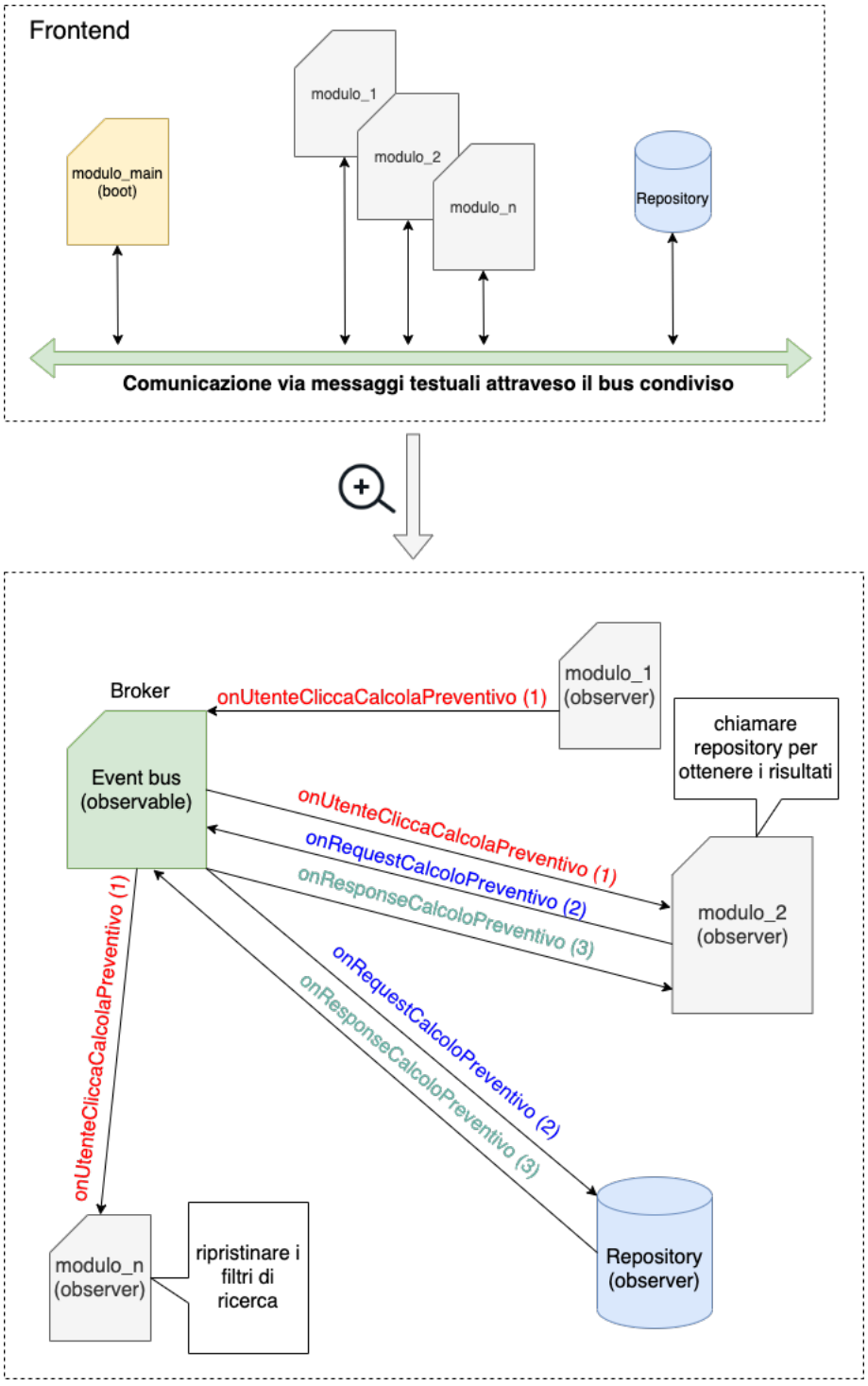
L'architettura del frontend del booking engine (vista top-down)

Questi due approcci hanno dato un notevole contributo nel corso dello sviluppo del nuovo Booking engine, permettendo al team di sviluppo di scrivere codice leggibile, con una struttura documentata. La definizione di ruoli e responsabilità per ogni membro del gruppo di lavoro, abbinata alla suddivisione del progetto in moduli, ha consentito di migliorare ulteriormente il coordinamento degli gli sviluppatori.

6.3 I design patterns utilizzati

Il primo pattern adottato è stato *observer pattern*, perché permette di instaurare un rapporto tra due o più entità definendo i ruoli di osservatore e soggetto; le entità osservatore attendono un messaggio che notifichi il completamento di un'operazione all'interno di un'entità soggetto. Tale pattern ha incentivato la scrittura di codice debolmente accoppiato, permettendo di perseguire la tesi che sostiene la separazione del codice e la riduzione delle dipendenze.

Il secondo pattern adottato è stato *event bus*, utile in abbinamento con *observer pattern*, perché permette di definire un mezzo di trasporto per i messaggi tra i vari moduli del frontend. Il flusso di un'applicazione web è intrinsecamente determinato dagli eventi generati dall'utente che ne usufruisce, la struttura dell'applicazione deve quindi permettere di generare ed elaborare gli eventi in modo asincrono; anche la separazione in moduli dei componenti richiede che essi possano comunicare tra loro in qualche modo. Esistono varie possibili architetture ed implementazioni dell'event bus, si è deciso di iniziare lo sviluppo con un modello semplice pensando alla possibilità di un'ottimizzazione futura che non compromettesse il funzionamento dei componenti già sviluppati, un event bus con topologia *broker*. In questa topologia il componente invia un messaggio testuale ad un intermediario centrale, il broker, il quale provvede ad inoltrarne una copia a tutti gli ascoltatori. In questo modo il componente sorgente ha segnalato il verificarsi di un particolare evento e tutti i componenti interessati a tale evento potranno eseguire le proprie operazioni in modalità asincrona.



Il principio di funzionamento dell'event bus nell'esempio di tre eventi e quattro osservatori che si comportano in base della propria logica programmata.

L'ultimo pattern applicato nello sviluppo del Booking engine è stato la macchina a stati finiti. Il concetto di un automa non è nuovo ed è molto diffuso nel campo dell'informatica, tuttavia solo recentemente è stato associato allo sviluppo web. Uno strumento che permetta di definire i finiti stati in cui un componente può esistere e di descrivere il comportamento deterministico degli eventi associati ad ogni transizione valida tra coppie di stati, rappresenta un'ottima soluzione per la gestione delle animazioni e del comportamento dei componenti all'interno di un Booking engine.

6.4 Programmazione orientata agli oggetti e programmazione funzionale

JavaScript è un linguaggio di programmazione multi paradigma, il paradigma di programmazione orientata agli oggetti (OOP) e il paradigma di programmazione funzionale (FP) sono i due predominanti ma non si sostituiscono uno all'altro, bensi si completano a vicenda.

Molte volte, nei progetti di medie e grandi dimensioni scritti in OOP, il programmatore lavora con oggetti appartenenti ad una gerarchia di classi profonda e complicata. Il maggior inconveniente si verifica quando è necessario riutilizzare una porzione di codice da classi esistenti, ma per farlo occorre rispettare tutte le dipendenze; questo problema prende il nome di *Banana Gorilla Jungle problem*, come definito dallo scienziato informatico inglese Joe Armstrong. Armstrong che sosteneva che "il problema con i linguaggi OOP è che hanno tutto questo ambiente implicito che si portano dietro. Volevi una banana, ma quello che hai ottenuto è stato un gorilla con in mano la banana e l'intera giungla" [6].
Un altro problema molto comune è la condivisione dello stato dell'oggetto tra i metodi dell'oggetto stesso che, durante l'esecuzione parallela di piú metodi, può portare a risultati non deterministici (ci si riferisce a questa situazione con il termine inglese *race condition*).

La FP fornisce a sua volta i seguenti concetti:

- facile composizione di piccole e semplici funzioni in funzioni di ordine superiore
- i dati in ingresso alle funzioni sono immutabili e il loro risultato deterministico
- l'impossibilità di modificare dati esterni al contesto della funzione (side effect) e l'assenza di uno stato condiviso tra qualunque componente del software

Questi concetti rimuovono alcune possibilità di errore, aumentando contemporaneamente la riutilizzabilità del codice già implementato senza rischiare di ricadere nel problema citato da Joe Armstrong.
Nonostante ciò, la FP non è la soluzione universale per tutti i problemi della programmazione; il motivo principale risiede nella rigidità della teoria sulla quale essa si basa che, a volte, è assai restrittiva, come nei casi in cui i side effects o lo stato condiviso siano desiderati o necessari. Per questo motivo, pur incoraggiando l'approccio funzionale, nel nuovo frontend si è deciso di procedere analizzando caso per caso come integrare al meglio entrambi i paradigmi.

6.5 Design delle API

Abbiamo approfittato dell'opportunità di riprogettare sia il backend che il frontend del nuovo backoffice per migliorare la struttura delle comunicazioni tra queste interfaccie. Si consideri il seguente diagramma:

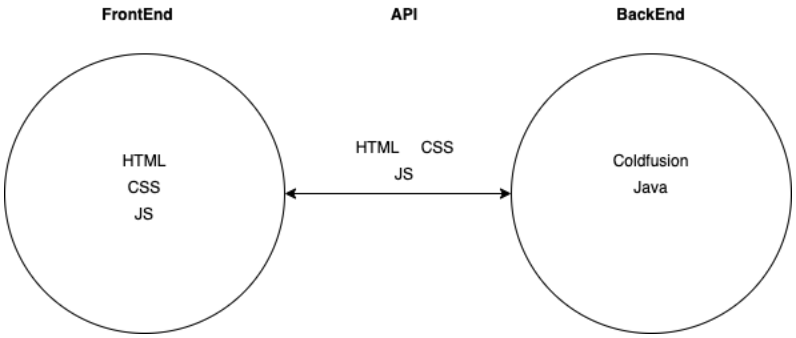


Diagramma d'interazione tra frontend e backend (versione precedente)

Il diagramma rappresenta il formato delle comunicazioni tra frontend e backend adottato nella precedente versione del Booking engine. Si noti che la fase di rendering era stata parzialmente delegata al server backend, in particolare quando questi generava e restituiva codice HTML pronto per essere utilizzato dal browser. Questa cattiva abitudine aveva instaurato una stretta dipendenza tra i dati restituiti dalle API e la logica specifica della piattaforma web, che ad oggi è implementata in HTML, CSS e JavaScript. Era stato violato il principio della separazione dei concetti cioè, nel caso d'analisi, si aveva un accoppiamento tra due livelli logici: il Model e la View dell'architettura Model-View-Controller. La principale conseguenza era l'impossibilità di lavorare su più piattaforme, la fragile gestione degli aggiornamenti e delle modifiche, nonché il forte legame agli specifici linguaggi di programmazione.

Ora si consideri il diagramma della situazione attuale:

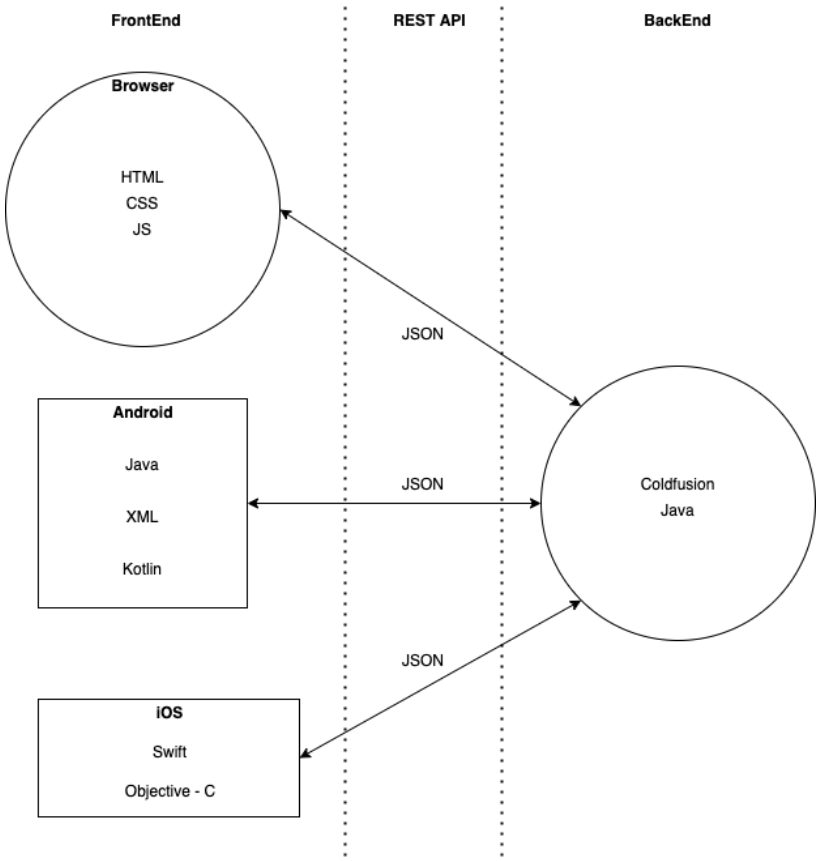


Diagramma d'interazione tra frontend e backend (versione nuova)

Osserviamo una chiara distinzione tra i livelli logici e l'interazione attraverso un'interfaccia uniforme, che permetta alle applicazioni frontend di comunicare con il backend usando un unico formato di scambio (json), con strutture dati interoperabili come numeri, stringhe di caratteri e valori booleani. Il livello API è stato riprogettato secondo lo stile REST API senza che fossero necessarie ulteriori librerie o software particolare. Le REST API non vincolano la risposta ad essere in un solo formato, ma è possibile usare alternative a JSON come XML, YAML o qualunque altro formato richiesto per una specifica piattaforma frontend.

La scelta del REST API è stata motivata dall'esigenza di poter effettuare modifiche nel frontend senza compromettere il funzionamento del backend o viceversa, nonché poter raggiungere altre piattaforme cruciali per la crescita dell'azienda come le applicazioni mobile native.
In aggiunta, visto che le REST API sono stateless, ciascuna chiamata API contiene tutti i dati necessari per l'autorizzazione al server, liberando perciò il server dall'esigenza di salvarli nelle sessioni.
Infine, REST API incoraggia a sfruttare le tecniche di caching di ogni piattaforma client, permettendo di ridurre il numero di chiamate al backend e di ottimizzare notevolmente i tempi di risposta e l'uso di risorse.

Capitolo 7

Strategie per la migrazione dal vecchio al nuovo sistema

Una delle fasi più critiche riguardanti il percorso di sviluppo di un software è il momento della migrazione dei dati tra la vecchia e la nuova versione. In questo capitolo ci concentreremo sulle problematiche riscontrate durante lo sviluppo del nuovo Booking engine.

7.1 Quando devi "sederti su due sedie contemporaneamente"

Ogni software ha un proprio ciclo di vita che include una fase di dismissione; al termine dello sviluppo il nuovo Booking engine aveva introdotto un rilevante numero di funzionalità prima inesistenti cercando di preservare quante più possibili di quelle già presenti. Tuttavia non importa quanto sia moderno o ricco di funzionalità un software, se la politica di migrazione non è ben progettata si rischia di, come minimo, perdere i clienti e danneggiare la loro fedeltà nel prodotto e, nel caso peggiore, arrivare al punto di non ritorno e dichiarare il fallimento. Il team di sviluppo perciò decise di attivare contemporaneamente i due sistemi, in modo che ogni cliente di Ipernet potesse continuare la propria attività nel vecchio sistema finché non avesse avuto il tempo necessario a migrare i dati su quello nuovo.

7.2 I compromessi per una migrazione al nuovo sistema più flessibile possibile

Pur di supportare i due sistemi in parallelo, durante il periodo della migrazione sono stati accettati alcuni oneri aggiuntivi sia di tipo tecnico che economico; il principale onere tecnico è stata la duplicazione del codice di alcune sezioni responsabili del completamento delle prenotazioni, l'onere economico è derivato dall'ampliamento dello spazio d'archiviazione necessario per le esigenze di due sistemi attivi in parallelo.

7.3 Come gestire lo spegnimento del vecchio sistema

Nonostante fossero stati adottati tutti gli accorgimenti necessari per mantenere attive sia la vecchia che la nuova versione del booking engine, era anche stato preparato il piano per la gestione dello spegnimento graduale del vecchio applicativo, facendolo finalmente diventare parte della storia di iper.net. La prima cosa da decidere fu il quando, vista la natura stagionale delle attività dei clienti esistevano sicuramente dei periodi di minor carico sul sistema in cui sarebbe stato possibile operare per minimizzare il disservizio. Oltre alla preparazione tecnica era altrettanto importante aggiornare il team di supporto, istruendolo ad operare anche sul nuovo sistema che a breve sarebbe stato utilizzato dai clienti.
Serviva inoltre un piano d'emergenza nel caso fossero state individuate delle strutture con una configurazione non migrabile in maniera automatica alla nuova versione del booking engin, doveva cioè essere possibile mantenere attiva la vecchia versione del booking engine solo per queste strutture; il team di supporto avrebbe provveduto a contattarle tempestivamente per guidarle e consigliarle durante l'aggiornamento.

Capitolo 8

Il lavoro in team

Il mutamento e l'evoluzione dei metodi di sviluppo del software ha sicuramente indotto un'evoluzione nell'organizzazione dei gruppi di sviluppo. È fondamentale che ogni membro segua il percorso di crescita dell'azienda in maniera consistente e in armonia con i colleghi. Nel mercato di oggi, per una sana crescita aziendale è necessario che:

1. i leader sappiano guidare il gruppo
2. la fiducia sia incoraggiata
3. le aspettative siano chiare a tutti
4. il riconoscimento e le ricompense siano condivisi sinceramente

Questo capitolo tratterà l'evoluzione del team di sviluppo di Ipernet durante lo sviluppo del nuovo Booking engine in riferimento ai quattro pilastri appena elencati.

8.1 Assegnazione e gestione dei compiti

Il lavoro di gruppo richiede una suddivisione dei compiti tra i membri e, nel caso dello sviluppo in un ambiente multidisciplinare e altamente complesso come quello in cui opera Ipernet, tale suddivisione è soggetta a costanti miglioramenti.

Uno dei recenti miglioramenti che ha riguardato l'organizzazione del team di sviluppo è stato l'introduzione della metodologia di lavoro agile, con un framework ibrido tra i più utilizzati, composto da *Scrum* e *Kanban*; il primo è stato sviluppato da Ken Schwaber e Jeff Sutherland, mentre il secondo è stato sviluppato all'interno della società automobilistica Toyota.
Di seguito vengono elencati i principi seguiti dagli sviluppatori durante lo sviluppo del nuovo Booking engine:

- Trasparenza – ogni membro mostra i propri compiti a tutti i collaboratori per mantenere un ottimo livello di fiducia nel gruppo
- Ritmo di lavoro coordinato – grazie alla trasparenza, il responsabile del progetto è in grado di impostare una tabella (roadmap) dei compiti da completare in un periodo di tempo preciso ed è altresì in grado di stimare la data di compimento del lavoro richiesto con alta precisione
- Adattabilità – nel caso si verifichi un riordino delle priorità e degli obiettivi, il framework di lavoro agile permette di alterare il processo di produzione in corso d'opera, sia per quanto riguarda i ruoli che per quanto riguarda i compiti
- Quadro generale – permette a tutti i collaboratori di conoscere l'avanzamento del progetto nel suo complesso
- Focus sulle consegne continue – è meglio compiere i lavori in sequenza che in parallelo

- Un numero ridotto di compiti – ogni sviluppatore è incoraggiato a ridurre il numero di attività intraprese purché egli concentri tutta la sua attenzione e competenza nel completamento dei lavori iniziati
- Time-frame ben definiti – si dà una stima dei tempi necessari e si cerca di consegnare il lavoro entro e non oltre i tempi prestabiliti, salvo i casi imprevisti

8.2 Limiti e poteri di ogni membro del team

Il secondo obiettivo nello sviluppo di una grande squadra è individuare le peculiarità, cioè i punti forti e i punti deboli di ciascun membro, infine ognuno desidera conoscere lo scopo del proprio lavoro e quale sia il proprio contributo nel quadro generale. Ogni membro ha delle competenze che gli permettono di affiancare un collega quando questi si trovi difficolta. L'ingresso di una nuova persona all'interno del team è forse il momento più complesso e delicato, perché è necessario condividere la conoscenza appresa da tutto il team negli anni precedenti ma è anche necessario lasciare all'ultimo arrivato il tempo di comprendere gli equilibri presenti; se il team è aperto e reattivo, come quello di Ipernet, una persona si adatta in tempi brevi e di conseguenza si possono riassegnare i compiti per diventare una squadra ancora più efficiente ed efficace nel lavoro collettivo.

8.3 Condivisione di scoperte e novità

Quando si lavora con l'obiettivo di crescere professionalmente è importante saper condividere le conoscenze già acquisite ed avere una mentalità aperta, curiosi di analizzare e comprendere le novità. È fondamentale per la crescita del team, e per il raggiungimento dei risultati, che tutti condividano i propri successi e ancora di più i propri fallimenti; questo aiuta a conoscere e ad ammettere i propri limiti, spingendo al miglioramento. La condivisione accelera la risoluzione dei problemi perché evita che tutti ripetano lo stesso errore, non è importante sapere chi abbia commesso un errore, ma è fondamentale comprendere come risolverlo e saperlo prevenire in futuro. Questo aiuta a superare i punti deboli di ogni membro del team in modo che tutti conoscano le proprie capacità e non venga lasciato nulla al caso.

8.4 Discussioni

La comunicazione tra tutti i membri del team è fondamentale; non è consigliabile limitare le comunicazioni ai soli membri interessati, si consiglia invece di coinvolgere tutta la squadra nelle sessioni di aggiornamento. Questo aiuta a sentirsi parte di qualcosa di più grande, aiuta le persone a chiedere aiuto quando si trovano in difficoltà e ad offrire il proprio sostegno quando altri si trovino in un punto di stallo. Discutere con tutto il team aiuta ad avere un feedback da diversi punti di vista, in modo che tutti gli ambiti del problema vengano analizzati e pianificati prima di iniziare a realizzare le soluzioni. È molto importante che le discussioni siano regolamentate e che non finiscano in polemica o nel vuoto. Grazie all'evidenza empirica, una discussione, anche per breve tempo, che riguarda gli obbiettivi di oggi e le problematiche

riscontrate nei giorni recenti, facilità il raggiungimento dell'obiettivo, che deve essere l'unico scopo del team.

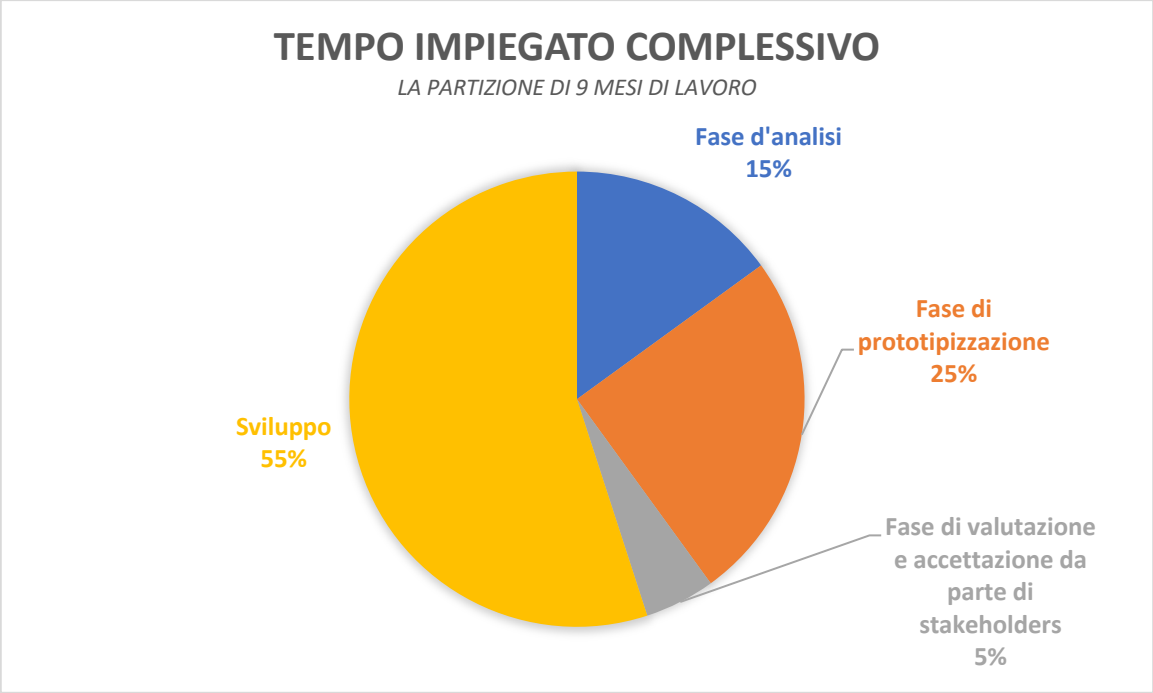
8.5 Deadlines

L'obiettivo del team è uno solo: produrre risultati in tempi accettabili e con la più alta qualità possibile. Darsi delle scadenze è fondamentale per non farsi travolgere dagli eventi, dagli imprevisti o da ambizioni facilmente giustificabili. È importante dividere un grande progetto in piccole deadline in modo che ogni settimana venga rilasciata una nuova parte di codice e/o vengano corretti i bug riscontrati. Questo aiuta il team a rendersi rapidamente conto se stia procedendo sulla giusta strada o se abbia accumulati troppi ritardi. In alcuni casi, per recuperare il ritardo è sufficiente focalizzarsi di più sulle funzionalità importanti rimandando quelle di minor importanza all'iterazione di rilascio successiva, altre volte invece è necessario che il team si compatti dedicando tutte le risorse alla soluzione di un problema che sta impedendo il progresso del progetto.
I ritardi nello sviluppo del software sono purtroppo all'ordine del giorno, eppure è importante cercare di avere la situazione sotto controllo, cosa veramente difficile se si osserva il progetto intero senza analizzare le piccole parti che lo compongono. Nel caso dello sviluppo del nuovo frontend, essendo un'applicazione web, è presente il notevole vantaggio di potere effettuare rilasci continui senza che il cliente finale se ne accorga, permettendo di aggiornare il software anche diverse volte al giorno. Le deadline, se definite con criteri realistici, non sono nemiche del team ma piuttosto lo aiutano ad essere più produttivo e a riprogrammare gli interventi non necessari.

Conclusioni

Il presente lavoro di tesi si era posto l'obiettivo di analizzare e descrivere lo sviluppo del software nel contesto di una piccola impresa di software italiana; mettere in evidenza lo spirito con il quale si affrontavano le sfide poste dal mercato e infine presentare il risultato finale.
Per fini statistici, il progetto di riscrittura ha coinvolto il lavoro di quattro ingegneri di software full-time, suddividendo il carico di lavoro tra due ingegneri frontend e due ingegneri backend. Comprendendo tutte le fasi, il lavoro ebbe l'inizio nel febbraio 2020 e il rilascio del prodotto finale avvenne nel novembre 2020, quindi in un totale di nove mesi.
La fase d'analisi iniziò a febbraio 2020 e si concluse agli inizi di marzo 2020.
Successivamente, il prodotto rimase nella fase di prototipazione grafica per i due mesi successivi. Una volta coperti tutti i casi d'uso e ottenuta l'approvazione da parte dell'amministrazione, alla vigilia d'estate il progetto entrò nella fase successiva, cioé quella dello sviluppo vero e proprio.

Nel seguente grafico appare evidente come la sola fase di sviluppo, compresi i tempi d'implementazione, d'integrazione e di testing, ha impiegato più del 50% del tempo complessivo, nel periodo che va dagli inizi di febbraio e fino alla data di rilascio ufficiale del 1° novembre 2020.



Si precisa inoltre che l'attività di Ipernet non è stata rallentata in alcun modo dalla diffuzione del virus COVID 19 poiché, lavorando in un campo IT, si è riusciti a dare una rapida risposta organizzativa nel contesto della pandemia. Il personale ha proseguito il lavoro in modalità remota (smart working) per tutto il 2020, consentendo che il prodotto fosse consegnato nei tempi previsti.

Il periodo di transizione concesso alle strutture ricettive per aggiornare i propri siti web dalla precedente versione del frontend a quella aggiornata è stato dal 1° novembre 2020 al 31 gennaio 2021. Un numero ridotto di strutture collaborò nella verifica del prodotto a partire da ottobre 2020, aiutando ad individuare e correggere tempestivamente alcuni difetti prima della data di rilascio.

Si desidera mettere in evidenza la portata e la complessità del lavoro svolto, senza neppure affrontare in dettaglio i cambiamenti dal backend, per un'applicazione web nel 2020. Se in precedenza il web fu una piattaforma per la mera condivisione di informazioni statiche, considerando la sua evoluzione negli ultimi 10 anni, siamo testimoni del cambiamento di un paradigma tecnologico in cui tante imprese e startup preferiscono affidare le proprie operazioni ad applicazioni web piuttosto che a quelle native.

La piattaforma di sviluppo web è ora ricca di strumenti indispensabili e a volte molto complessi nel loro utilizzo. Le competenze necessarie per far fronte ai problemi architetturali di un'applicazione web sono allo stesso livello di quelle necessarie per un'applicazione tradizionalmente eseguita sul calcolatore dell'utente finale. La nuova versione del Booking engine di Ipernet ne è la dimostrazione. Il prodotto continua a crescere e ad arricchirsi di nuove funzionalità, diventando sempre più complesso sia per gli sviluppatori che per gli utenti finali; nonostante ciò il prodotto gode di grande popolarità tra gli agenti del settore dell'ospitalità. La riscrittura dell'applicazione ha permesso di fornire risposte solide a fabbisogni quotidiani del settore di oggi, creando stabili fondamenta per affrontare i fabbisogni del mercato di domani.

Ringraziamenti

Si ringraziano tutti i professori del corso di “Scienze e ingegneria informatica” del campus di Cesena per il loro impegno professionale di altissimo livello. Un particolare ringraziamento al professor Vittorio Ghini per i saggi consigli e l’incredibile disponibilità.

Un ampio ringraziamento va a tutti i colleghi di Ipernet per il supporto fornito durante la composizione di questo lavoro, per le competenze trasferite nel campo pratico dell’ingegneria del software che sono state le basi indispensabili durante lo sviluppo del nuovo Booking engine.

Infine, un richiamo speciale alla mia famiglia per il supporto e i sacrifici senza i quali non sarei riuscito a integrarmi nel tessuto sociale e a concludere i miei studi.

Sitografia

[1] - http://www.ontit.it/opencms/opencms/ont/it/stampa/in_evidenza/Panoramica_Turismo_in_Italia.html
[2] - https://www.mycomp.it/blog/trends-e-statistiche-turismo/catene-alberghiere-in-italia-scenario/
[3] - https://www.progress.com/company
[4] - https://it.wikipedia.org/wiki/Filosofia_Unix - cite_note-3
[5] - https://it.wikipedia.org/wiki/Spaghetti_code
[6] - http://www.codersatwork.com/