ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA

Dipartimento di Informatica – Scienza e Ingegneria
Corso di Laurea in Ingegneria e Scienze Informatiche

PROGETTAZIONE E REALIZZAZIONE DELLA FUNZIONALITÀ DI PRENOTAZIONE SLOT PER CARICO E SCARICO MERCE IN UN MAGAZZINO

Elaborato in
PROGRAMMAZIONE AD OGGETTI

Relatore
Prof. MIRKO VIROLI

Presentata da
FRANCESCA TONETTI

Correlatore
VALERIO BORIONI

Anno Accademico 2018 – 2019

Alla mia famiglia e a coloro che
mi sono stati vicini in questo percorso.

Sommario

Lo scopo di questa tesi è quello di descrivere e analizzare la realizzazione di una nuova feature all'interno di un'applicazione Client-Server inerente all'ambito industriale, precisamente legata ai sistemi di trasporto, per automatizzare il processo di carico e scarico dei mezzi nei piazzali delle aziende produttrici. Questo progetto nasce grazie a una proposta da parte dell'ente Onit Group SRL che da anni collabora con aziende come Orogel, Unieuro, Ondulkart, le quali usufruiscono del software gestionale "On.TMS" per pianificare la consegna della merce ai clienti destinatari. L'estensione software richiesta ha come volontà quella di permettere sia all'azienda sia ai trasportatori, di migliorare e organizzare in maniera efficiente l'accesso alle bocche di carico/scarico. Per lo sviluppo del sistema sono state adottate tecnologie note, come C#, che permettono agli sviluppatori una facile modifica ed estensione. Il lavoro di sviluppo è stato preceduto da una serie di incontri con le aziende di interesse che hanno consentito di comprende a pieno le loro esigenze e i punti sui quali focalizzarsi. Nel testo sono analizzati i requisiti che il sistema deve soddisfare, le scelte progettuali e quelle implementative, gli strumenti di sviluppo adottati e la sperimentazione del software attraverso test specifici. Il risultato ottenuto è un sistema che fornisce tutte le funzionalità di base per la prenotazione di slot orari a seconda di caratteristiche definite a partire dalle aziende. Inoltre, le interfacce realizzate, il cui aspetto si ispira al software "On.TMS" già in uso, consentono agli utenti finali un utilizzo semplice e intuitivo.

Indice

Introduzione

Il progetto proposto come tesi sperimentale dall'ente Onit Group SRL consiste nella realizzazione di una nuova funzionalità all'interno di un'applicazione software Client-Server inerente all'ambito industriale, precisamente legata ai sistemi di trasporto.

L'idea nasce a partire da precise richieste proposte da aziende come Orogel, Unieuro e Ondulkart che da anni utilizzano il software gestionale "On.TMS" per pianificare il trasporto della merce.

Attualmente, quando una azienda di trasporto è chiamata a prendere in carico un viaggio pianificato, lo fa in base alle tempistiche relative allo scarico dei prodotti presso i destinatari senza considerare la disponibilità dei magazzini o bocche di carico/scarico libere presso le aziende produttrici. Tale organizzazione comporta spesso un eccessivo spreco di tempo nelle piazzole di sosta delle aziende, una disorganizzazione interna ai magazzini per la preparazione della merce, insoddisfazione nell'organizzazione lavorativa sia dei trasportatori che delle aziende con le quali collaborano.

Per questo motivo nasce la necessità di aggiungere al software in uso, uno strumento condiviso che permetta a tutti i fornitori di un ente di prenotare slot temporali liberi inseriti in un calendario settimanale, questo per evitare sovrapposizioni all'arrivo di più fornitori diversi. Il calendario a sua volta potrà essere configurato dalle aziende stesse in base a scelte organizzative interne, a seconda del giorno della settimana o degli orari in cui il flusso del lavoro può essere variabile.

L'obiettivo del progetto è quello di ottenere un software, con una interfaccia grafica user-friendly, che permetta la fruizione in modo semplice e intuitivo della prenotazione di slot temporali da parte dei trasportatori. La volontà è quella di migliorare, mediante funzionalità complete, la gestione piazzale di ogni azienda per il carico e scarico della merce con l'obiettivo di

monitorare e diminuire i tempi di attesa, organizzare l'accesso alle bocche facendo sì che tutti possano ricavarne profitto e vantaggio nella propria organizzazione lavorativa.

Per lo sviluppo del progetto sono stati adottati strumenti e tecnologie che rendono il sistema facilmente estensibile e modificabile nonché Visual Studio e C# come linguaggi di sviluppo, il Pattern MVC per la parte di programmazione, reactive extensions all'interno del codice e Vue.js come supporto allo sviluppo dell'interfaccia grafica.

Il fine ultimo di questo processo di sviluppo è quello di inglobare la funzionalità inerente alla gestione piazzale all'interno del software "On.TMS" con l'obiettivo di portarla in produzione nei sistemi dei clienti che l'hanno commissionata.

L'elaborato di tesi riguarda il primo prototipo funzionante per la gestione piazzale. Al suo interno è stato suddiviso in sei capitoli, che permettono di ripercorrere tutte le fasi di realizzazione e sperimentazione svolta. A questi capitoli, seguono le conclusioni e gli sviluppi futuri.

- Il primo capitolo fornisce una panoramica sui sistemi TMS (Transport Management System) e sullo scenario applicativo specifico del software "On.TMS".

- Il secondo capitolo, a partire dal precedente, analizza le problematiche evidenziate e si sofferma sui requisiti del sistema richiesti dai committenti, definendone il comportamento e le qualità che deve possedere.

- Il terzo capitolo descrive le fasi di progettazione dell'architettura e dell'interfaccia grafica, approfondendo le motivazioni che hanno portato a tali scelte.

- Il quarto capitolo, gli strumenti e i metodi di sviluppo adottati durante il corso della realizzazione del progetto.

- Il quinto capitolo si sofferma sulla fase implementativa del sistema, andando ad approfondire le scelte implementative che hanno portato al completamento del software.

- Il sesto capitolo tratta la sperimentazione del software, analizzando le metodologie utilizzate e i risultati ottenuti.

Capitolo 1

Background

Il sistema TMS *[12]* con il quale ci si è interfacciati per la realizzazione della seguente tesi, consiste in un software di gestione trasporti che permette di pianificare il miglior routing possibile per la consegna della merce ai clienti. Precisamente "On.TMS" è il software web realizzato da Onit per la pianificazione e l'ottimizzazione della logistica dei trasporti.

Si tratta di una piattaforma modulare in grado di adattarsi alle esigenze esistenti tra le aziende manifatturiere e di autotrasporto, società di brokeraggio del trasporto e operatori intra logistici.

I principali vantaggi che un sistema come il TMS assicura ai propri clienti sono molteplici, tra i quali una riduzione dei costi operativi, una riduzione dei tempi di attività d'ufficio, completa tracciabilità dei carichi e un aumento della saturazione dei veicoli.

1.1 Logistica

Comunemente l'uso del termine logistica si riconduce a un processo inerente al trasporto merci *[7]*. Contrariamente a questa convinzione, la logistica è molto più di questo.

Il termine in questione indica tutto ciò che ha un senso logico, organizzativo e ad oggi la logistica industriale è una delle componenti più importanti del ciclo produttivo, la cui pianificazione spesso può determinare il successo o l'insuccesso imprenditoriale di un'azienda.

Il Council of Logistics Management scrive: "*La logistica è il processo di pianificazione e controllo dei prodotti e delle relative informazioni dal punto di origine al punto di consumo con lo scopo di soddisfare le esigenze dei clienti [..]*".

1.2 Logistica dei trasporti

Nel caso specifico della logistica dei trasporti *[8]*, questa si occupa della gestione della rete di distribuzione della merce secondo gli accordi intercorsi fra l'azienda e il cliente.

Nell'ambito dei costi della distribuzione fisica della merce da un punto di origine alla destinazione finale, mediamente i costi dei trasporti incidono quasi la metà del totale; naturalmente si possono evidenziare variazioni rispetto alla media in relazione al settore merceologico preso in esame.

Per questo motivo nasce la necessità di risolvere il problema della riduzione dei costi di trasporto andando a toccare gli aspetti organizzativi e di pianificazione delle attività stesse mediante lo studio di numerosi fattori come:

- i flussi di traffico e tratte di consegna,
- i livelli di saturazione degli automezzi in relazione alle migliori modalità di carico,
- i tempi tra il carico del mezzo nel magazzino dell'azienda produttrice e la consegna finale del prodotto.

È sempre importante ricordare che una migliore gestione della logistica dei trasporti, consiste in un potente strumento di vantaggio o detrimento per il successo delle aziende produttrici.

1.3 Scenario Applicativo specifico

"On.TMS" è il software realizzato da Onit Group *[13]* per le aziende interessate alla gestione logistica della spedizione dei prodotti di vendita ai clienti, dai

quali vengono contattati attraverso chiamate o mediante altre integrazioni a livello di database.

Una volta effettuato un nuovo ordine questo verrà inserito nel sistema dell'azienda, come Orogel o Unieuro, con tutti i dati riferenti all'attività commerciale che lo ha commissionato. Per ogni ordine importato nel software gestionale si ha la possibilità di visualizzarlo in uno spazio apposito nel quale è visibile l'elenco di prodotti ordinati con relativo peso e quantitativo, la data di partenza e di arrivo, il riferimento ad altri ordini passati già presi in carico in un viaggio e portati in consegna al destinatario.

Lo scopo di "On.TMS", una volta ricevuti gli ordini, è quello di consentire di pianificare i viaggi in modo da trasportare la merce usufruendo del miglior percorso di routing, ottimizzando viaggi e tappe o selezionando i veicoli con i quali giungere alle destinazioni. All'utente è permesso modificare manualmente tutte queste caratteristiche assegnando quindi autisti o modificando l'ordine dei viaggi in base a decisioni interne, per questo motivo molto spesso "On.TMS" viene utilizzato e gestito da uno o più utenti appartenenti all'azienda produttrice.

Alle aziende che usufruiscono del software è data la possibilità di richiedere l'aggiunta del portale trasportatori nonché un altro spazio del software stesso, che permette all'azienda di poter comunicare con i trasportatori con i quali collabora nel caso in cui sia necessario effettuare operazioni più controllate. L'accesso al portale è consentito sia all'azienda stessa mediante login con credenziali da amministratore, sia ai singoli trasportatori attraverso username differenti in modo tale che ogni trasportatore possa accedere solo al proprio spazio personale mentre l'azienda produttrice ha la possibilità di visualizzare tutti gli utenti.

Dal momento in cui un viaggio viene creato e assegnato a un trasportatore, nel portale per quel preciso trasportatore, apparirà un nuovo viaggio incaricato.

Genericamente ogni trasportatore nella pagina principale visualizza la lista di viaggi a lui assegnati e per ognuno dovrà indicare la data di carico o ritiro merce, confermare il veicolo con il quale raggiungerà il piazzale dell'azienda e altre caratteriste specifiche. Oltre ai controlli per ogni viaggio, è presente una gestione di certificazione gestista da politiche interne molto rigide secondo le quali sono assegnabili multe o sanzioni per coloro che risultano incongruenti al momento dell'accettazione viaggio.

La comunicazione tra “On.TMS” e il portale trasportatore risulta utile per quanto riguarda l’intera gestione dei viaggi compreso il momento di carico o scarico merce.

Capitolo 2

Requisiti

Per progettare il sistema richiesto, in modo che possa soddisfare i bisogni del committente, è necessario svolgere una fase preliminare durante la quale definire i requisiti di sistema. Tali requisiti indicano cosa il cliente richiede a un sistema, cioè lo scopo del software, senza però definire cosa andrà effettivamente costruito.

La raccolta dei requisiti è stata eseguita con lo svolgimento di una serie di colloqui e interviste, le quali sono state realizzate direttamente con responsabili e dipendenti delle aziende di interesse, nonché Orogel, Unieuro e Ondulkart.

L'obiettivo finale di questo elaborato è quello di ottenere un sistema efficiente tramite il quale, sia le aziende che i trasportatori, possano ricavarne vantaggi nella propria gestione lavorativa. Inoltre, attraverso questo strumento, alle aziende è permesso personalizzare il calendario settimanale di pianificazione degli arrivi per carico e scarico merce in modo da automatizzare le prenotazioni nel portale trasportatori.

2.1 Problematiche evidenziate

A partire dallo scenario applicativo specifico, analizzato nel paragrafo del capitolo precedente, il portale trasportatore consente quindi agli utenti di autenticarsi, visualizzare la lista di viaggi assegnati e presi in carico e modificare il dettaglio di richiesta di un viaggio pianificato.

Questo tipo di approccio ha messo in mostra alcuni aspetti poco efficienti per quanto riguarda l'organizzazione pratica del carico e scarico merce nei magazzini delle aziende: specificare unicamente la data nel dettaglio di un viaggio spesso ha comportato affollamenti nel piazzale delle ditte in orari di punta provocando lunghe attese e di conseguenza numerose lamentele da parte dei trasportatori, da i dipendenti dell'azienda produttrice incaricati a preparare la merce nei magazzini e soprattutto dai destinatari.

Per questo motivo è nata l'idea ma soprattutto la necessità di aggiungere una feature più specifica inerente alle prenotazioni piazzale attraverso slot temporali collocati in un calendario settimanale, in modo che il carico merce sia pronto in magazzino e disponibile per quel trasportatore al momento del suo arrivo. Allo stesso modo le prenotazioni da parte dei trasportatori saranno visibili in un calendario personale dell'azienda.

La volontà è quella di minimizzare i tempi di attesa, rendere fluidi i passaggi di carico o scarico del magazzino in relazione alle prenotazioni.

A partire da questo aspetto e anche grazie alle richieste delle stesse aziende di interesse, nasce l'idea di consentire una configurazione del calendario attraverso il quale i trasportatori effettueranno le prenotazioni. Tale configurazione potrà variare in base alla disponibilità delle bocche di carico, al numero di dipendenti nelle varie fasce orarie di lavoro, a scelte progettuali interne. Una volta impostato il calendario, i trasportatori con cui collaborano potranno selezionare unità di prenotazioni per il carico e lo scarico della merce nei magazzini.

2.2 Colloquio con gli stakeholder

Come osservato nei punti precedentemente analizzati, la fase riguardante il colloquio con i destinatari occupa una parte essenziale dello sviluppo di un software. Questa fase precede l'analisi dei requisiti e risulta essenziale per comprendere a fondo i problemi evidenziati, le caratteristiche sulle quali soffermarsi maggiormente, i vincoli da rispettare ed eventuali suggerimenti.

Comunemente per tutte le tre aziende che hanno commissionato il progetto per la prenotazione piazzale, nonché Orogel, Unieuro e Ondulkart, le difficoltà riscontrate e i punti sui quali costruire il progetto sono stati i medesimi.

Tutte le aziende produttrici sono costituite da più magazzini situati in un piazzale comune e ciascuno composto da numerose bocche addette al carico e scarico merce. L'arrivo dei trasportatori comporta spesso un accumulo di mezzi nel piazzale in orari di punta e questo comporta il rischio che il carico o lo scarico richieda alcune ore causando, oltre che un malcontento dei trasportatori stessi, una serie di problematiche inerenti alle politiche di viaggio per le ore di sosta obbligatorie o il numero di ore di guida. Si delineano quindi i vari obiettivi:

- organizzare ogni fascia oraria in base all'afflusso di mezzi di trasporto
- numero di dipendenti disponibili nei magazzini per il carico o scarico merce
- arrivo di mezzi con lunga gittata nei primi orari della giornata e al contrario quelli con corta gittata in fasce orarie differenti senza occupare posti necessari ad altri trasportatori.

Tutte queste caratteristiche rientrano nell'idea di definire un calendario configurabile di giorno in giorno ma ancor più di ora in ora, mediante slot temporali.

Con l'aiuto di più figure appartenenti a ogni azienda produttrice, è stato possibile stilare una serie di vincoli secondo i quali costruire l'idea per la configurazione manuale del calendario e gli obiettivi finali da raggiungere in termini di miglioramento dei tempi di attesa e gestione di carico/scarico merce in relazione all'arrivo dei traportatori.

Di seguito si è svolta l'analisi dei requisiti tenendo conto di tutte le nozioni raccolte in ogni colloquio.

2.3 Analisi

L'analisi dei requisiti rappresenta una attività preliminare per lo sviluppo di un sistema software, il cui scopo è quello di definire le funzionalità, i requisiti che devono essere soddisfatti. Questi ultimi descrivono ciò che i committenti si aspettano dal sistema e specificarli significa esprimerli in modo chiaro, univoco, consistente e completo.

Durante il processo di analisi è fondamentale usufruire dei colloqui con gli stakeholder in modo da capire a pieno le necessità sulle quali costruire il sistema.

2.3.1 Requisiti funzionali

I requisiti funzionali sono prerequisiti che specificano le interazioni tra il sistema e l'ambiente. Vanno perciò a definire le funzioni che il sistema possiederà.

Per prima cosa sono stati definiti gli attori, cioè gli utenti esterni che sistema deve supportare. In tal caso specifico saranno distinti due attori differenti riguardanti la figura dell'utente, nonché dipendente dell'azienda produttrice, che avrà il compito di configurare il calendario e la figura del trasportatore che potrà accedere al portale per effettuare modifiche a un viaggio ad egli assegnato, specificando uno slot temporale in relazione alla data di carico o scarico merce:

- L'*utente/amministratore/Il dipendente,* colui incaricato a configurare il calendario settimanale dell'azienda produttrice per la quale lavora, in base a caratteristiche variabili come ad esempio: il numero di magazzini di carico/scarico, il numero del personale disponibile nell'arco di una giornata, l'afflusso dei mezzi di trasporto nel piazzale e altri fattori ancora.

- il *trasportatore*, cioè un dipendente della ditta di trasporti con la quale l'azienda produttrice collabora, incaricato nell'accettazione di un viaggio pianificato e alla prenotazione di uno slot temporale collocato in un calendario settimanale per confermare l'arrivo in piazzola.

In seguito, sono stati identificati i casi d'uso, mostrati in Figura 2.1 e in Figura 2.2 per quanto riguarda "On.TMS" completo di portale trasportatore. Inoltre verranno descritti di seguito i requisiti di cui sono composti.

Configurazione calendario
Il sistema deve consentire agli utenti, nonché dipendenti dell'azienda produttrice, di selezionare la configurazione di un calendario che consenta a

sua volta di specificare i valori di ampiezza e capacità per ogni singolo spazio-temporale selezionato, definito come uno slot. Tali specifiche potranno variare in base a caratteristiche inerenti a scelte lavorative dell'azienda.

Vedi Figura 2.1

Prenotazione slot

Il sistema deve permettere all'utente di prenotare uno spazio temporale disponibile attraverso la visualizzazione di slot temporali in un calendario. Tali spazi prenotabili saranno visualizzati in base alle disponibilità reali e alle caratteristiche del mezzo di trasporto dell'utente stesso.

Vedi Figura 2.2

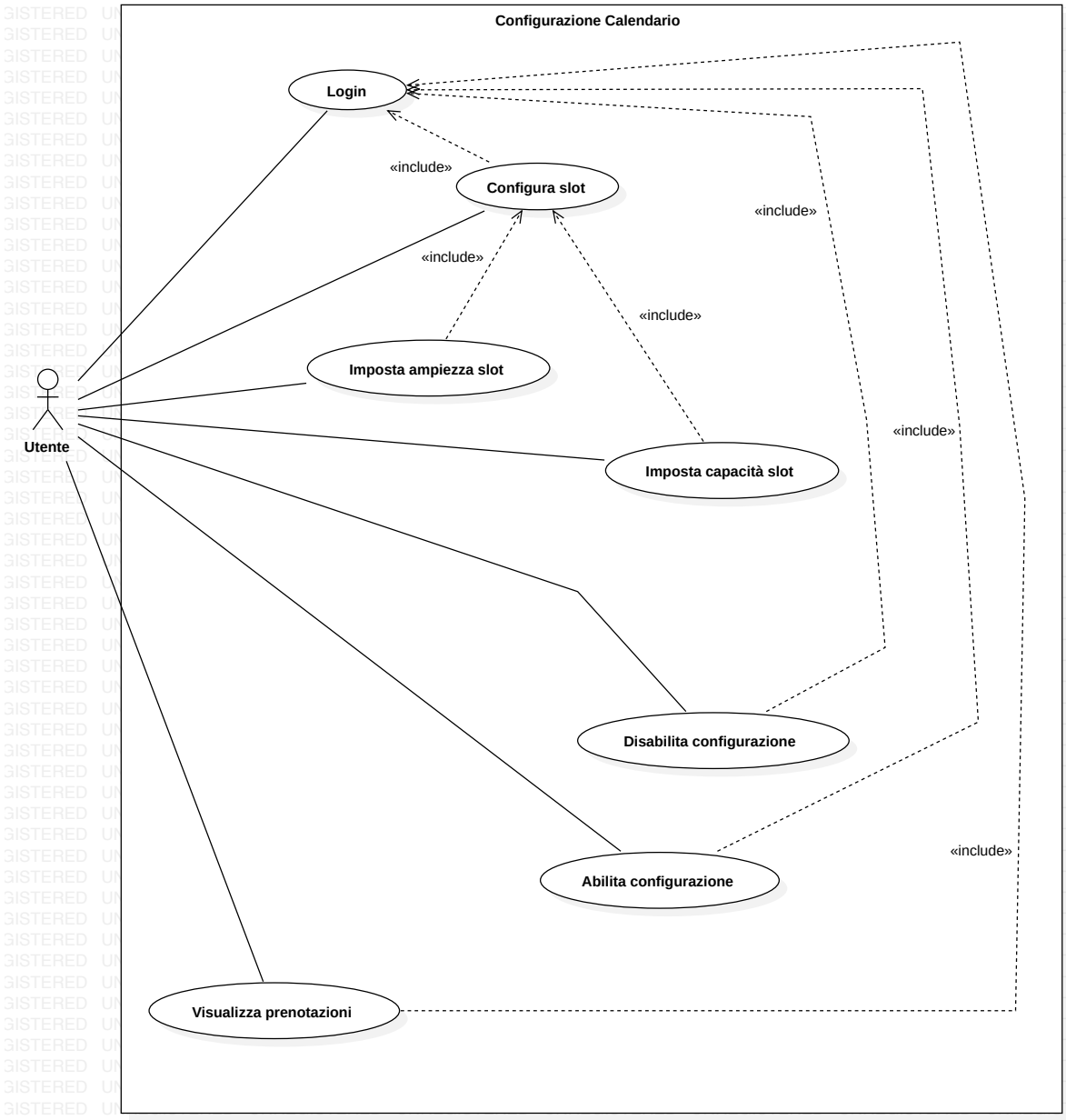


Figura 2.1: Diagramma dei casi d'uso della configurazione calendario – Portale Trasportatore.

Il sistema permetterà all'utente, nonché dipendente dell'aziende produttrice, le seguenti operazioni:

- effettuare il login da superutente nel sistema "On.TMS" portale trasportatori;

- configurare i singoli slot o unità di prenotazione in base a specifiche caratteristiche di ampiezza e capacità;

- impostare il valore di ampiezza inerente ad uno o più unità di slot selezionati, con ampiezza si intende il range orario "da" "a" per ogni unità di prenotazione selezionata;

- impostare il valore di capacità inerente ad uno o più unità di slot selezionati, con capacità si intende il valore massimo di mezzi di trasporto ammissibili per ogni unità di prenotazione selezionata;

- abilitare la versione della configurazione calendario manuale, rendendolo visibile per il trasportatore in fase di dettaglio viaggio;

- disabilitare la versione di configurazione calendario manuale;

- visualizzare la lista di prenotazioni prese in carico dall'azienda all'interno di un calendario configurato;

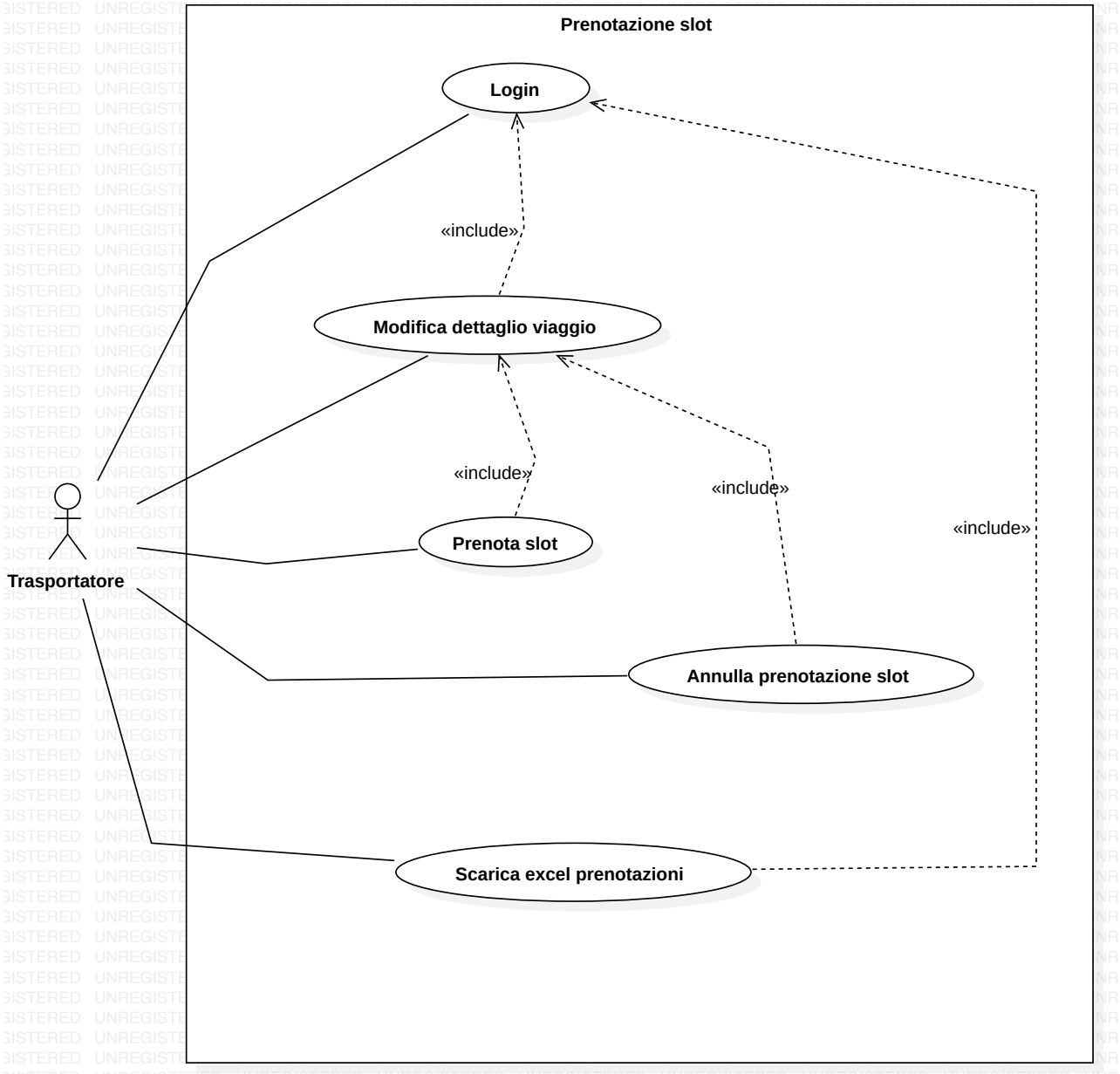


Figura 2.2: Diagramma dei casi d'uso della prenotazione Slot – Portale Trasportatore.

Il sistema permetterà all'utente, nonché dipendente dell'aziende produttrice, le seguenti operazioni:

- effettuare il login nel sistema "On.TMS" portale trasportatori;
- modificare il dettaglio di un viaggio assegnato all'attore;
- prenotare uno slot temporale collocato in un calendario settimanale raffigurante le disponibilità di accesso al piazzale dell'azienda per la quale ha in carico il viaggio;
- annullare la prenotazione di uno slot;
- scaricare il file Excel riguardante le prenotazioni effettuate sul portale;

2.3.2 Requisiti non funzionali

I requisiti non funzionali sono quelle proprietà misurabili dall'utente e non correlate al comportamento funzionale del sistema.

Per questo progetto, i requisiti funzionali individuati sono esposti qui di seguito.

Interfaccia user-friendly
L'interfaccia del software da realizzare deve essere facile da usare, e soprattutto in linea con quelle già presenti all'interno di "On.TMS". Questo per consentire agli utenti che ne faranno uso di avvalersi di tale fruizione in autonomia, senza bisogno di una specifica o approfondita formazione.

2.3.3 Requisiti implementativi

I requisiti implementativi comprendono quei tipi di requisiti inerenti agli aspetti interni del software, utili per i programmatori stessi del software.

Utilizzo di tecnologie note standard
Il sistema che si vuole ottenere deve essere estensibile per coloro che volessero

ampliarne il funzionamento, similmente al resto del software "On.TMS". Per questo sono state utilizzate tecnologie standard e note alla maggior parte degli sviluppatori.

Realizzazione di un sistema facilmente estensibile modificabile
Il sistema che si vuole realizzare deve essere progettato in modo che possa essere facilmente estensibile e modificabile da chiunque ne abbia la necessità all'interno del team aziendale, senza provocare delle modifiche a catena nel codice. In questo modo, nel futuro, sarà possibile aggiungere nuove estensioni in relazioni ai feedback di coloro di cui ne faranno uso.

Capitolo 3

Architettura e progettazione

Una buona pratica per costruire un modello che permetta di mostrare come il sistema dovrà essere strutturato e i componenti che ne andranno a far parte è definire l'architettura del software prima di implementare il codice.

L'architettura è l'organizzazione del sistema in termini di componenti e relazioni che intercorrono tra loro e l'ambiente. Attraverso l'architettura si definiscono anche i principi che guidano la progettazione e l'evoluzione del software, perché le scelte fatte sugli aspetti non visibili all'utente finale vanno a vincolare i singoli elementi del progetto. L'obiettivo è quindi quello di velocizzare la realizzazione delle singole funzionalità del sistema, cercando di soddisfare tutti i requisiti.

Le decisioni prese in questa fase del progetto hanno avuto un impatto decisivo sullo sviluppo del lavoro.

3.1 Struttura architetturale

L'architettura è stata definita nella prima fase di progettazione, con lo scopo di scomporre il sistema in sottosistemi.

Per rispettare questo obiettivo è stata scelta come tipologia architetturale quella Client-Server, poiché ritenuta la più idonea alle esigenze del progetto seguendo le orme del resto del software "On.TMS".

La progettazione del sistema ha portato a ottenere un'architettura

costituita da tre componenti, mostrati in Figura 3.1, che comunicano tra loro per scambiarsi i dati. Questi componenti sono i seguenti:

- un *Client*, nonché il componente che accede ai servizi offerti dal server e interagisce direttamente con l'utente finale, sia per quarto riguarda "On.TMS" sia per il portale trasportatori;

- un *Server*, che è il componente che si occupa della gestione logica del sistema, fornendo servizi al client e svolgendo operazioni sul database;

- un *Database*, che è il componente preposto alla memorizzazione dei dati. Vi può accedere solamente il server, al quale comunica le informazioni in base alle richieste ricevute.

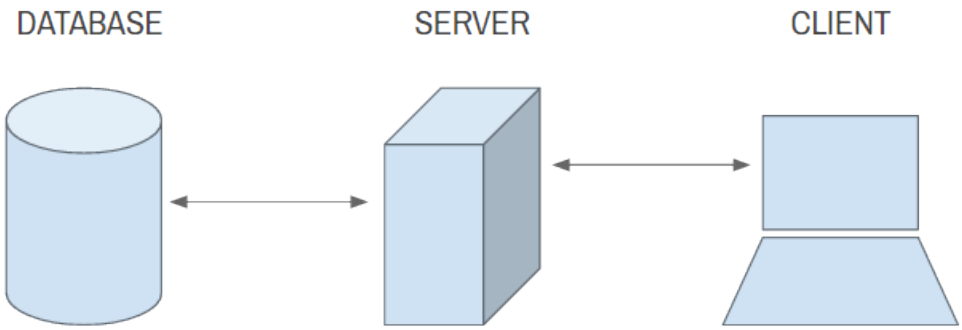


Figura 3.1: Visione grafica complessiva dell'architettura del sistema.

3.1.1 Scelta dei pattern architetturali

Una tra le ultime operazioni svolte per completare la progettazione della architettura del software è stata quella di individuare e scegliere i pattern architetturali da applicare al progetto.

Con pattern architetturale si descrive il modello organizzativo strutturale di un sistema software in termini di sottoinsiemi e relazioni che intercorrono tra essi. Consideriamo quindi una soluzione generale a un

problema che si verifica comunemente all'interno di un'architettura. La selezione dei pattern architetturali per il progetto è una decisione fondamentale per quanto riguarda la progettazione di un sistema software. Rispetto ai design pattern, il pattern architetturale opera a un livello più alto ed esprime schemi che permettono di impostare l'organizzazione strutturale di un sistema.

Il pattern architetturale scelto per il progetto, seguendo lo sviluppo del resto del software "On.TMS" è: Model-View-Controller (MVC).

Model-View-Controller

Il pattern Model-View-Controller (o MVC) *[9]* è un pattern che consente di separare la logica di presentazione dei dati dalla logica di business.

Il MVC, la cui struttura è mostrata in Figura 3.2, consente di suddividere l'applicazione in tre componenti:

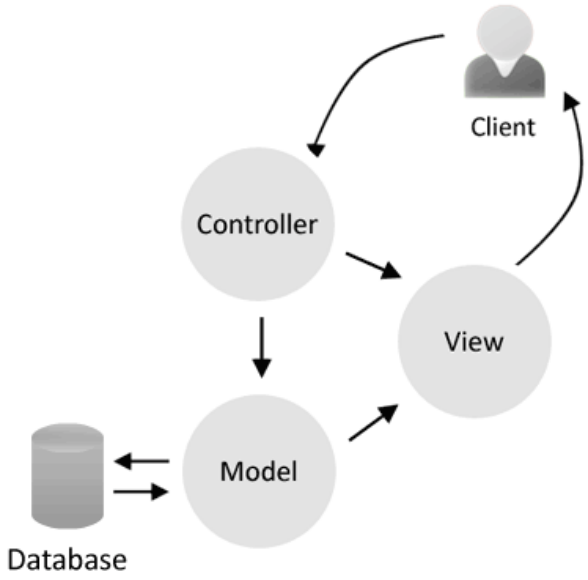


Figura 3.2: Divisione dei ruoli in MVC.

- il *model*, in cui sono contenuti e gestiti i dati e che descrive la logica del programma;

- la *view,* con cui si mostrano le informazioni all'utente attraverso delle interfacce grafiche;

- il *controller,* che è situato tra il model e la view, perciò gestisce gli input dell'utente e fornisce le informazioni necessarie al model.

Il vantaggio che porta la divisione in tre componenti dell'intera applicazione è che ogni singolo componente può essere riutilizzato, modificato e sostituito, senza che tali operazioni vadano a incidere sugli altri. Proprio per questo motivo il pattern MVC, come per il resto del software "On.TMS" in uso, è stato applicato al progetto corrente per trarne tutti i benefici.

3.2 Progettazione del database

I dati raccolti durante l'utilizzo del software devono rimanere persistenti per poter essere utilizzati o analizzati anche in momenti futuri. È necessario usufruire di una base dati in cui memorizzare le informazioni in modo strutturato.

Un database consiste in un archivio dati organizzati secondo una logica precisa e la conoscenza di tale struttura dati permette di inserirli o modificarli in modo rapido.

La tipologia di database scelta a supporto per il progetto corrisponde a quella già in uso per "On.TMS" nonché un database di tipo non relazionale ma documentale, ovvero che memorizza le informazioni in documenti in formato BSON (Binary JSON) e non necessita della definizione di uno schema, come invece avviene per i database SQL. Nello specifico il sistema scelto per la gestione del database non relazione è RavenDB.

A seguito della scelta del NRDBS, è stata definita la struttura dei dati e dei documenti necessari per rappresentare tutte le informazioni. Tra i principali definiti troviamo:

- Calendario: documento che racchiude le informazioni riguardanti il calendario configurato dall'azienda produttrice.

- SlotPrenotati: documento che comprende per ogni giorno la lista di slot prenotati in una precisa data in modo che sia più facile interrogare e modificare i dati.

- Prenotazioni in giorno: documento che definisce per ogni giorno del calendario la lista di slot prenotati in quella precisa data.

3.3 Progettazione dell'interfaccia grafica

La progettazione dell'interfaccia grafica nella progettazione della funzionalità di carico e scarico in un magazzino di un'azienda produttrice è stata uno dei punti centrali del progetto, in quanto rappresenta ciò con cui l'utente finale si rapporta per poter interagire con il programma.

Lo scopo di questa fase progettuale è stato quello di ottenere come risultato un'interfaccia che fosse semplice, intuitiva e soprattutto facile da usare ispirata alla grafica già realizzata per "On.TMS". Inoltre, è stato utile focalizzarsi sulla user experience perché senza una buona esperienza dell'utente, ci può essere un alto rischio di abbandono dell'uso del software e un ritorno ai metodi tradizionali per l'arrivo dei trasportatori nel piazzale di carico/scarico.

La progettazione si è composta di due fasi: la realizzazione iniziale di wireframe e successivamente i mockup.

Il wireframe è una bozza del lavoro che dovrà essere svolto, che mostra lo scheletro completo dell'interfaccia. Tale bozza è stata realizzata su carta è diventata la base per i mockup.

Il mockup invece consiste in una rappresentazione statica del prodotto completo, che contiene un alto livello di dettaglio e fedeltà. Serve per mostrare all'utente cosa si prevede di ottenere come risultato finale, senza però usufruire dell'interattività. I mockup realizzati rappresentano il punto di passaggio tra le fasi di analisi/progettazione e l'effettiva implementazione dell'interfaccia grafica del sistema.

La struttura della GUI è in portali differenti per quanto riguarda l'utente dipendente dell'azienda produttrice e il trasportatore loggato per confermare le pianificazioni di viaggi a esso assegnati. Nello specifico: la sezione per la configurazione del calendario, la vista delle prenotazioni effettuate dai trasportatori per una specifica azienda, la prenotazione di slot

temporali collocati in un calendario settimanale. Ognuna di queste sezioni verrà brevemente analizzata qui di seguito.

3.3.1 Sezione configurazione calendario

Nella sezione riguardante la configurazione del calendario si prevede di gestire la conformazione di quest'ultimo mediante la specifica di valori corrispondenti ad ampiezza e capacità per ogni slot temporale selezionato all'interno del calendario.

Nella pagina principale di questa sezione, visibile in Figura 3.3, sarà rappresentato un calendario vuoto di una generica settimana formata dai canonici giorni della settimana, che partono dal lunedì. Da questa schermata sarà possibile selezionare uno o più slot temporali di cui ogni giorno è composto e inserire i valori corrispondenti ad ampiezza e capacità con i quali verranno configurati. Lo spazio dedicato all'immissione dei valori verrà visualizzato mediante una modale.

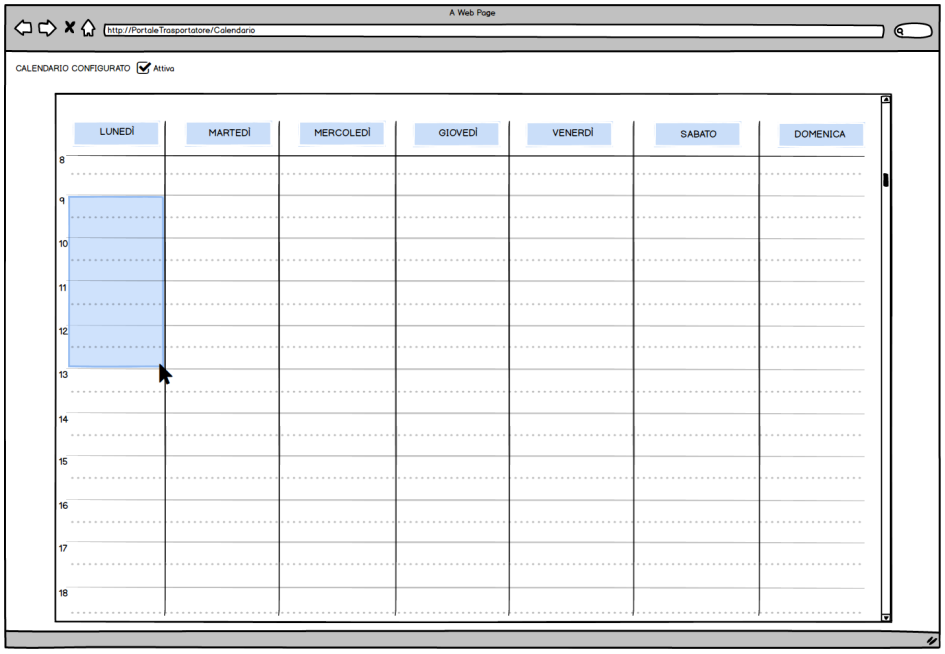


Figura 3.3: Mockup della pagina principale per la configurazione del calendario.

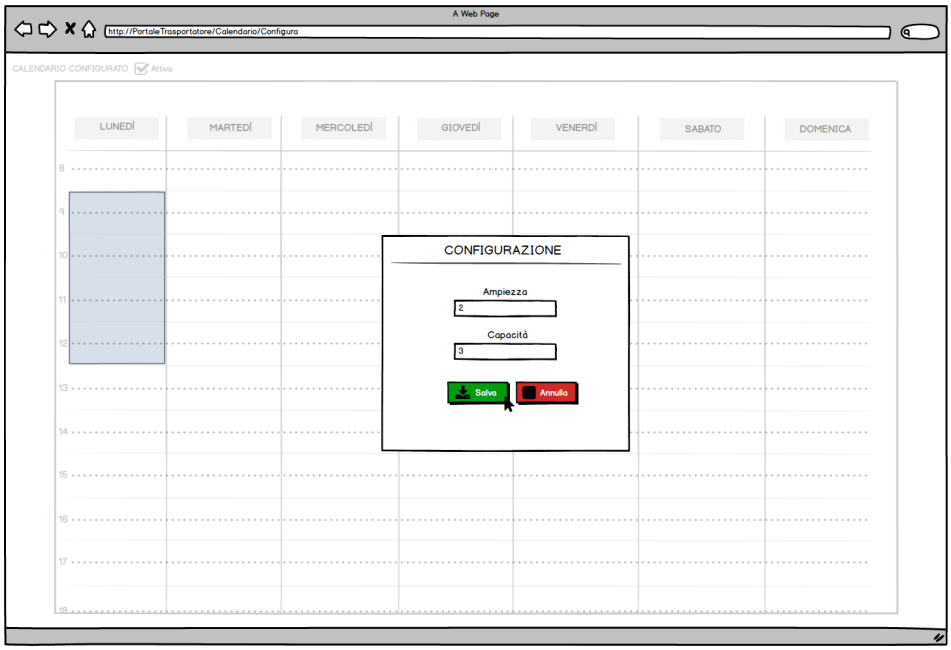


Figura 3.4: Mockup che mostra il passaggio di immissione dei valori per la configurazione degli slot selezionati nella pagina sottostante.

3.3.2 Sezione lista di prenotazioni

Tramite la sezione dedicata alla visualizzazione della lista delle prenotazioni, appartenente agli utenti dipendenti dell'azienda produttrice e mostrata in Figura 3.5, si vuole permettere all'azienda di visualizzare il proprio calendario configurato e di verificare gli arrivi per ogni giorno della settimana.

La pagina consiste in un calendario identificato dalla data della settimana corrente, che renderà visibile il nome dei traportatori prenotati per ogni giorno o meglio per ogni unità temporale di prenotazione.

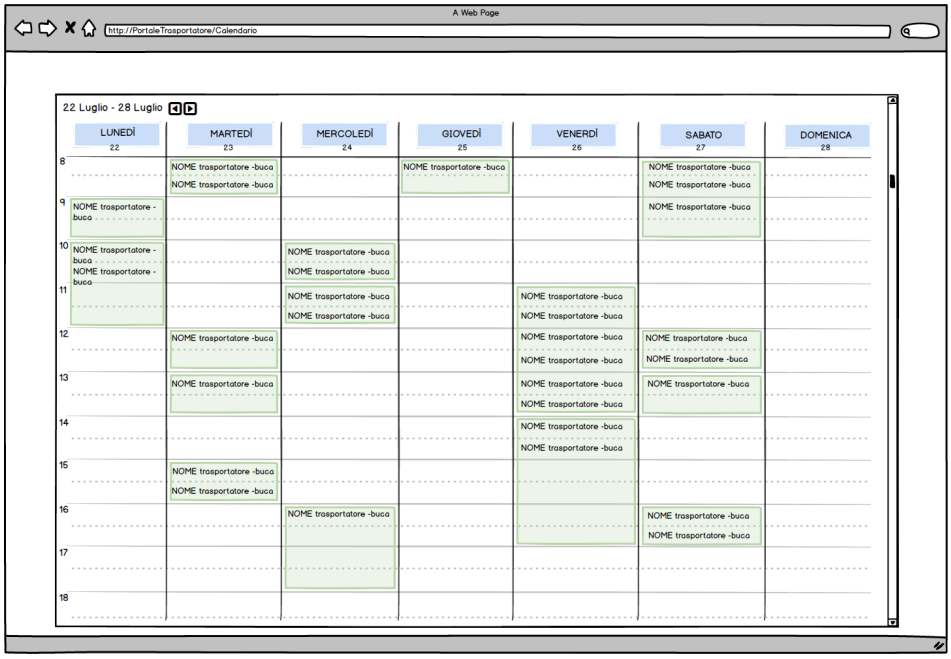


Figura 3.5: Mockup della pagina riguardante la visualizzazione delle prenotazioni.

3.3.3 Sezione prenotazione slot

La sezione della prenotazione slot è inserita all'interno della pagina di Dettaglio Viaggio già presente nel portale trasportatore; con l'inserimento di un calendario configurato, dalla selezione della data di arrivo si aprirà una modale che raffigura il calendario costituito da slot liberi e prenotabili identificati da un orario di inizio e di fine.

Una volta selezionato uno spazio disponibile con voce *prenota*, verrà richiesta una ulteriore conferma di prenotazione al termine della quale l'orario dello slot prenotato sarà visibile nel campo input denominato ora.

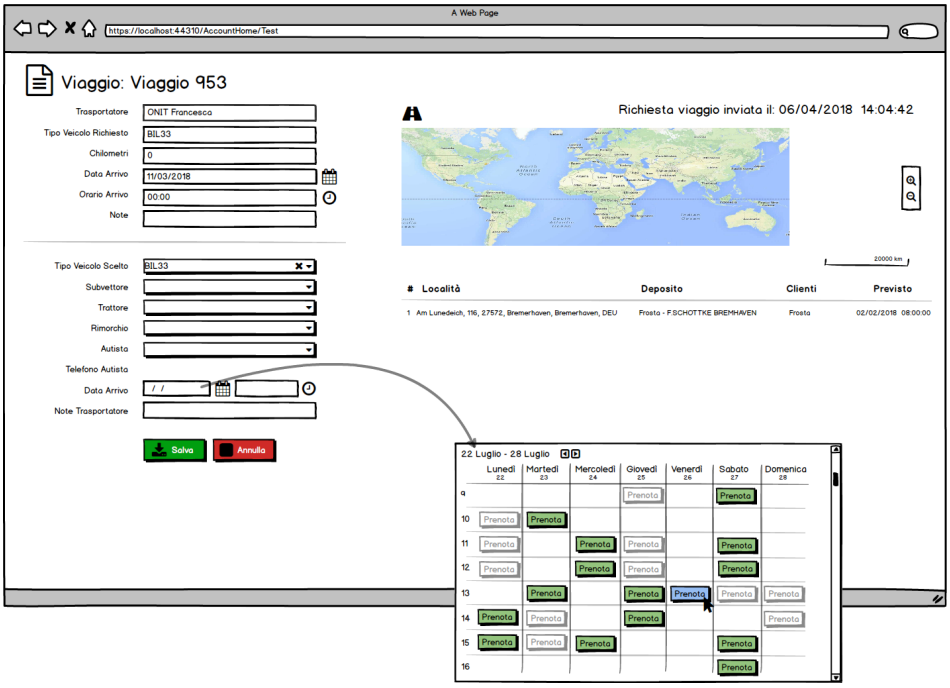


Figura 3.6: Mockup riguardante la prenotazione dello slot disponibile.

3.4 Design di dettaglio dell'applicazione

A seguito della progettazione dell'interfaccia grafica è stato possibile procedere con la progettazione dei componenti che devono comporre l'applicazione.

Il design di dettaglio ha richiesto due passi principali:

- la definizione della struttura del modello;
- la definizione delle viste e dei controller.

3.4.1 Modello

Il modello è stato progettato in modo che contenesse tutti metodi di accesso agli oggetti del sistema. Le classi che modellano i singoli oggetti rappresentano la vera e propria logica del programma.

Entrando più nello specifico, possiamo identificare gli oggetti del modello nel seguente modo: per poter gestire gli slot e la prenotazione di questi, come richiesto dai committenti, si ha la necessità di avere una struttura base nonché un oggetto Calendario che permetta di modellare i dati al suo interno. Quest'ultimo contiene una lista di oggetti di tipo Giorno che a sua volta contengono una lista di oggetti, denominati Slot, che rappresentano il punto focale per le azioni di prenotazione. All'interno del model si è scelto di suddividere a sua volta il concetto di slot in due classi separate a seconda che si trattasse di un SlotConfigurato o uno SlotPrenotato, questo per gestire in maniera migliore le azioni di visualizzazioni degli spazi prenotabili nel calendario del trasportatore.

Tra le altre classi ritroviamo una classe di tipo Booking che contiene la logica di prenotazione degli slot disponibili o annullamento. Una classe denominata Regole che permette di verificare quando ancora sia possibile o meno effettuare una prenotazione a seconda di specifici criteri di verifica.

3.4.2 Viste e controller

A seguito della progettazione del modello, sono stati progettati le viste e i corrispondenti controller, i quali ne gestiscono il funzionamento. Nella pratica, i controller definiti andranno ad aggiornale le viste a essi associate, in base alle interazioni dell'utente con la vista e si serviranno dei modelli, precedentemente definiti, per svolgere le operazioni sui dati.

Di seguito verranno mostrati i diagrammi delle classi corrispondenti alle fasi di sviluppo sia per quanto riguarda una prima implementazione a partire dai test riferiti ai casi d'uso principali, sia per quanto riguarda la successiva verifica della persistenza dati mediante l'uso di service e comandi.

Per entrambe i diagrammi sotto raffigurati, Figura 3.7 e Figura 3.8, le spiegazioni più esaustive e complete saranno analizzate nel capitolo 5 riguardante l'implementazione del progetto.

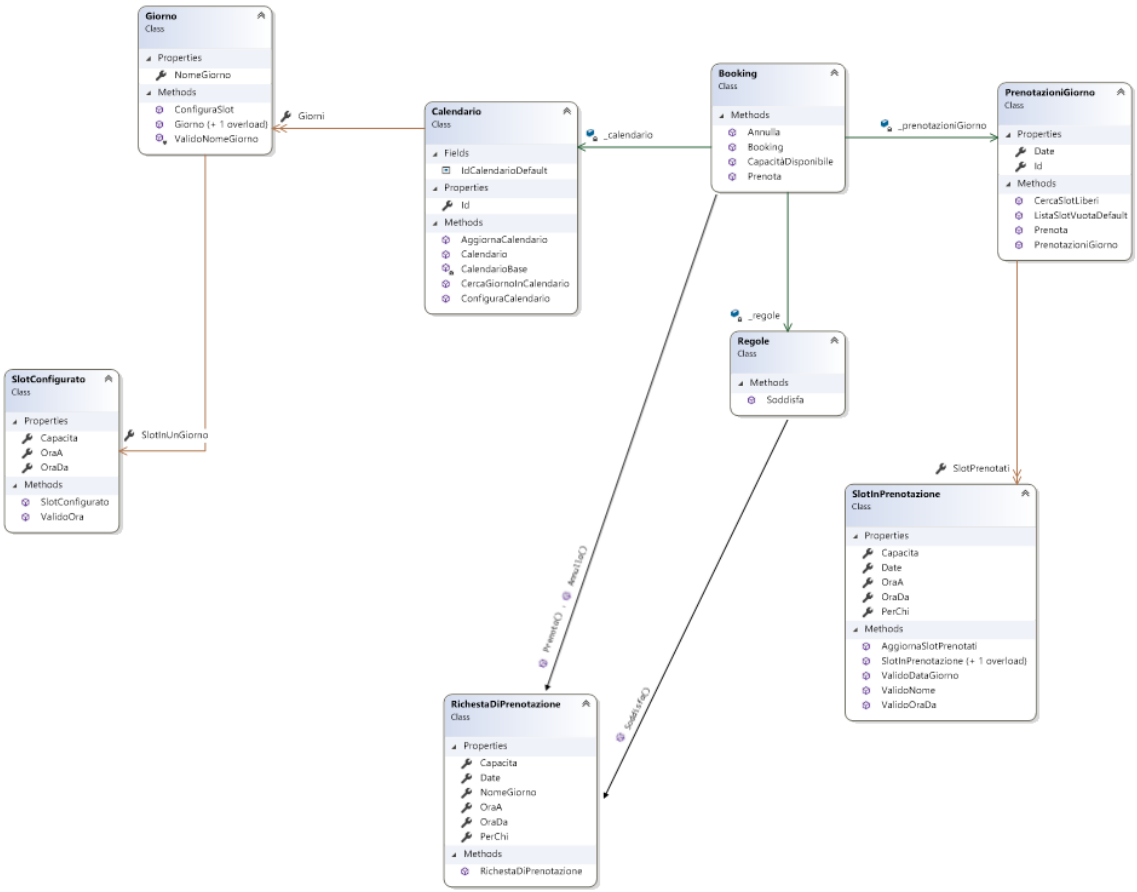


Figura 3.7: Diagramma delle classi riguardanti la fase di test sui casi d'uso principali.

Le classi sopra raffigurate e le relazioni che intercorrono fra esse rappresentano il modello del sistema definito a partire da una serie di test attraverso i quali sono state costruite tutte le classi principali.

Vedi Capitolo 6 per la trattazione dei test di cui ne hanno usufruito.

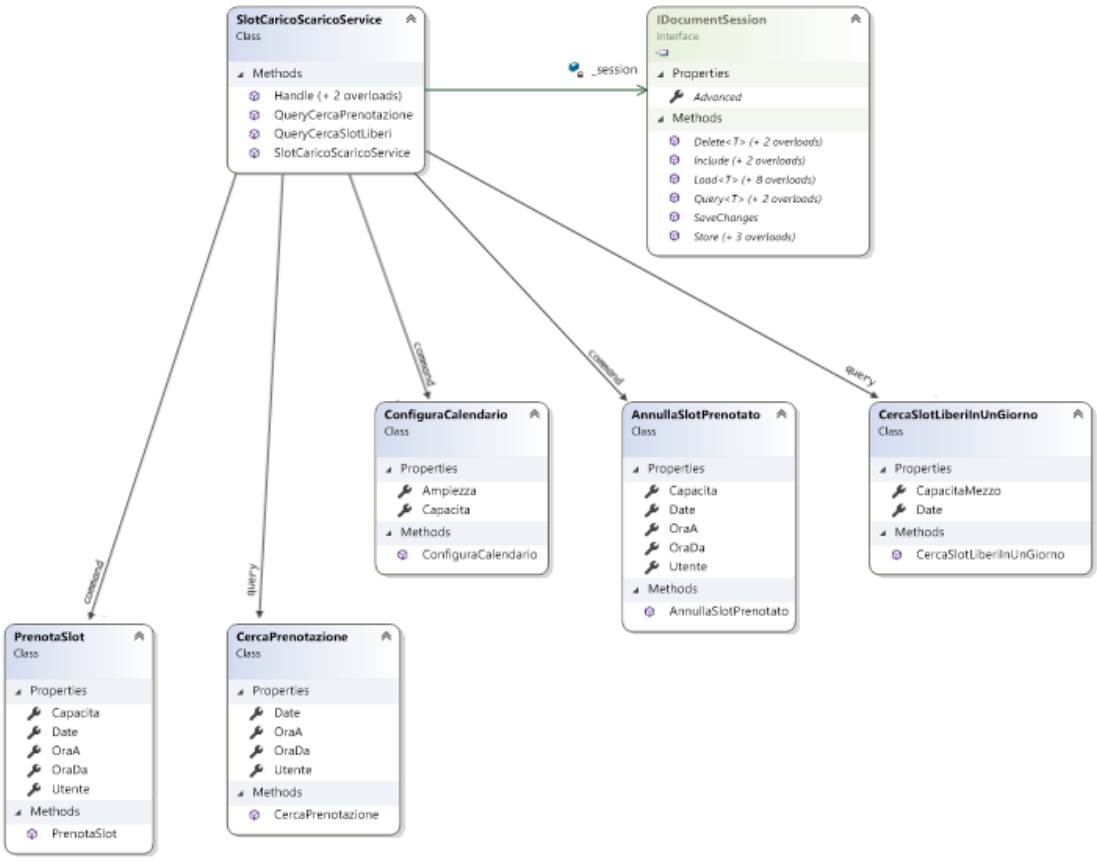


Figura 3.8: Diagramma delle classi riguardanti le classi di comandi e service del progetto.

Le classi sopra raffigurate corrispondono ai comandi ed alle query inserite nel service del sistema per testare la persistenza dei dati.

Vedi Capitolo 5 per la trattazione completa.

Capitolo 4

Strumenti e metodi di sviluppo

La scelta degli strumenti e dei metodi di sviluppo da adottare per implementare il progetto rappresenta una decisione molto importante perché ha ripercussioni sulla qualità del codice e sulla produttività degli sviluppatori.

Qui di seguito verranno illustrati i principali strumenti, metodi di sviluppo del codice e pratiche di test adottate in relazione con le metodologie impiegate per lo sviluppo precedente del software "On.TMS".

4.1 Agile software development

L'essenza del pensiero Agile viene in contrasto con la tradizionale ingegneria del software che prevede un piano predittivo precedente allo sviluppo *[5]*. Ciò su cui si basa lo sviluppo Agile è l'essere adattivo e non predittivo ed è orientato alle persone invece che ai processi.

L'ingegneria Agile vede lo sviluppo del software come un'attività principalmente umana in cui le persone coinvolte rappresentano il motore principale del successo mentre i processi e gli strumenti consentono di migliorare l'efficacia di una squadra, rimanendo sempre influenze di secondo ordine.

L'adozione di uno sviluppo software Agile non è un percorso facile o rapido ed è riassumibile in una serie di tappe che passano dalla focalizzazione sul valore, alla consegna e ancora all'ottimizzazione di tale valore.

Di seguito, in Figura 4.1, un modello di fluidità Agile che riassume i concetti precedentemente descritti.

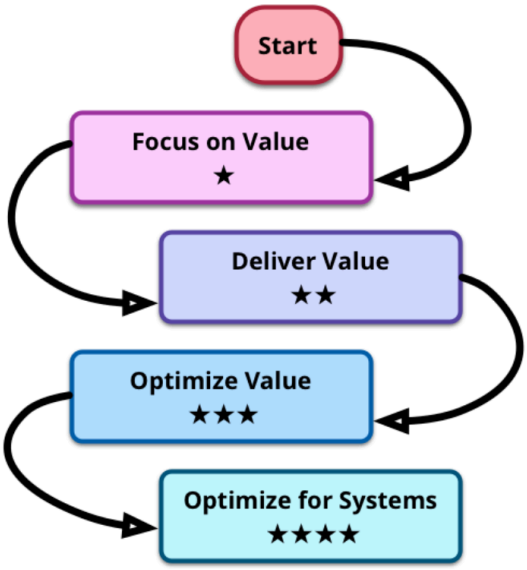


Figura 4.1: Modello di fluidità Agile.

Per portare a termine uno sviluppo software basato sui concetti di Agile e usufruire dei vantaggi di produttività e reattività di cui vanta, bisogna avere solide pratiche tecniche. La consegna continua rappresenta un vantaggio essenziale nell'approccio Agile che consente a sua volta di controllare in maniera efficiente il cambiamento del progetto.

Per far funzionare tutto questo è necessaria una pipeline di distribuzione che assicuri che il software sia costruito in piccoli incrementi pronti per la produzione. Di fatto molte organizzazioni Internet dominanti rilasciano questi incrementi eseguendo distribuzioni di produzione ogni giorno.

4.2 Practical Test

Il software pronto per la messa in produzione richiede numerosi test di verifica. Con il maturare della disciplina dello sviluppo del software, anche gli approcci

al testing del software sono maturati e per questo motivo i team di sviluppo si sono orientati verso l'automazione di gran parte di essi invece di avere numerosi software di test manuali. Questo per poter sapere in pochi secondi se il software presenta errori e rotture.

Il circuito di feedback drasticamente abbreviato alimentato da test automatizzati va di pari passo con pratiche di sviluppo agili e con consegna continua.

Il software è diventato una parte essenziale del mondo in cui viviamo e oggi le aziende cercano di trovare il modo di diventare aziende digitali di prima classe per velocità senza sacrificarne le caratteristiche relative alla qualità. Si parla di operazioni di *delivery*, nonché consegna di software pronto per essere messo in produzione in qualsiasi momento e l'utilizzo di una pipeline di *build* per testare automaticamente il software e distribuirlo negli ambienti di test e produzione.

Costruire, testare e distribuire una quantità sempre maggiore di software diventa presto impossibile, per questo l'automazione dell'intero processo diventa l'unica strada da percorrere.

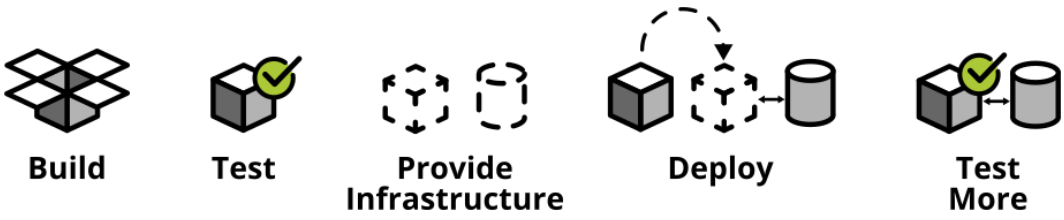


Figura 4.2: Pipeline di costruzione per avviare automaticamente e in modo affidabile il software in produzione.

Per poter prendere in considerazione l'automazione di test per un software in sviluppo è necessario conoscere un concetto chiave descritto in Figura 4.3 ovvero la *Piramide dei Test [4].*

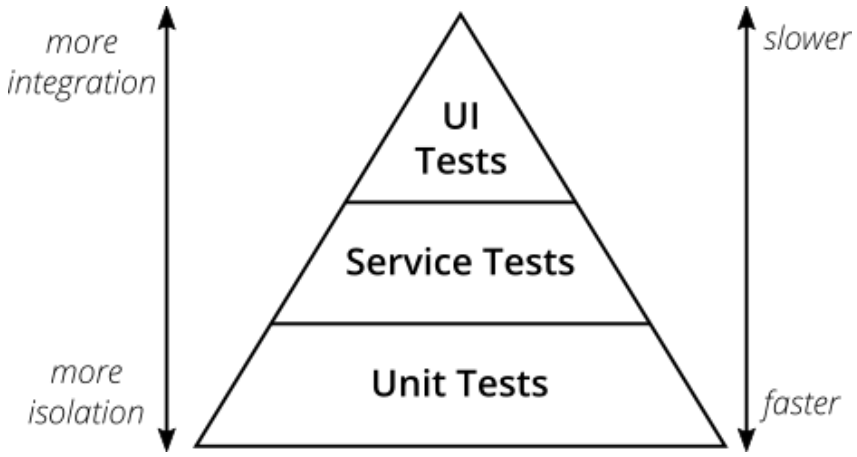


Figura 4.3: The Pyramid Test.

Mike Cohn è colui che ha ideato questo concetto nel suo libro Succeding with Agile. Corrisponde a una grande metafora visiva che spiega come pensare a diversi livelli di test e anche quanti test fare su ogni livello.

La piramide di test originale di Mike Cohn è composta dai tre strati che la suite di test dovrebbe comprendere (procedendo dal basso verso l'alto):

1. test unitari
2. test di servizio
3. test dell'interfaccia utente

Il concetto di base consiste quindi nel creare una suite di test veloce e mantenibile ovvero scrivere molti test di piccole unità che risultano veloci, scrivere solo alcuni test più approfonditi e pochi test ad alto livello che testano l'applicazione da un capo all'altro.

Per quanto riguarda la struttura, è consigliabile seguire il seguente ordine:

1. impostare i dati del test
2. chiamare il metodo sotto al test

3. creare gli assert di verifica tra i risultati attesi e quelli effettivamente restituiti

Attraverso questa struttura ho effettuato i controlli per testare il progetto riguardane la prenotazione piazzale. L'obiettivo era quello di visualizzare tutti gli assert come corretti per cui con la verifica andata a buon fine.

4.3 Continuous Integration e Distribuited Version Control System

L'integrazione continua *[3]* è una pratica di sviluppo software in cui i membri di un team integrano il proprio lavoro frequentemente e dove ognuna di queste integrazioni viene verificata da una build automatizzata per rilevare eventuali errori il più rapidamente possibile.

La maggior parte di questi progetti software che coinvolgono molti file da integrare tra loro per costruirne un prodotto completo richiedono l'uso di strumenti come i DVCS nonché sistemi distribuiti di controllo della versione. Un *Distribuited Version Control System* ha l'obiettivo di condividere le modifiche e le versioni del codice sorgente del software. In questo modo, gli sviluppatori possono collaborare individualmente e parallelamente allo sviluppo di un progetto comune.

Un DVCS facilita la gestione dei cambiamenti perché non richiede strategie di coordinamento per mantenere la coerenza all'interno di un progetto dopo che sopraggiungono modifiche al codice.

I principali vantaggi che si ottengono con l'adozione di tali metodologie riguardano le prestazioni, perché si permette a centinaia o migliaia di sviluppatori di contribuire allo stesso progetto in contemporanea, avendo a disposizione la cronologia completa delle modifiche (vedi Figura 4.4).

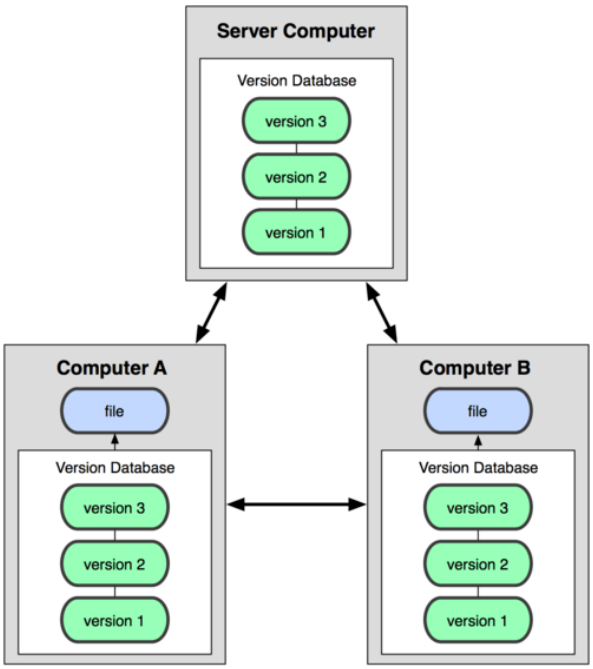


Figura 4.4: Schema del funzionamento di un distibuted version control.

Proprio per questi vantaggi è stato deciso di fare ricorso a esso anche per lo sviluppo di questa nuova funzionalità in modo da rendere ampliabile il progetto anche in un futuro da altri sviluppatori, come il resto del software "On.TMS".

In termini specifici il DVCS scelto per questo progetto è Git, uno dei più popolari software di controllo di versione distribuito open source esistenti attualmente *[1]*. In particolare, una estensione di Git per Windows nonché Git Extensions.

I suoi vantaggi sono:

- velocità;
- semplicità nell'utilizzo;
- forte supporto per lo sviluppo non lineare (permette molti branch);

Il funzionamento di Git è molto semplice, come mostrato in Figura 4.5, ogni volta che si fa commit o si salva lo stato del progetto, Git fa un'istantanea di come appaiono i file in quel momento senza memorizzare quelli che

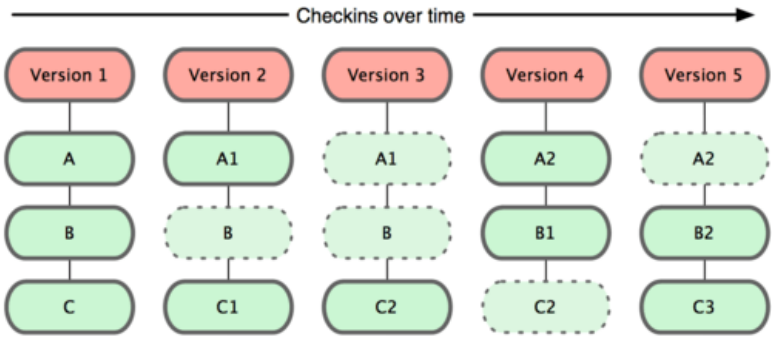


Figura 4.5: Memorizzazione dei dati con istantanee del progetto nel tempo tramite Git.

non hanno subito modifiche. Questa è la peculiarità che differenzia Git da un qualsiasi altro VCS.

Entrando più nello specifico, all'interno del progetto per la prenotazione piazzale è stato adottato un modello di branching per Git adatto alla collaborazione con un team di sviluppo.

Si prevedono dunque due branch principali, con una struttura simile a quella in Figura 4.6, dove:

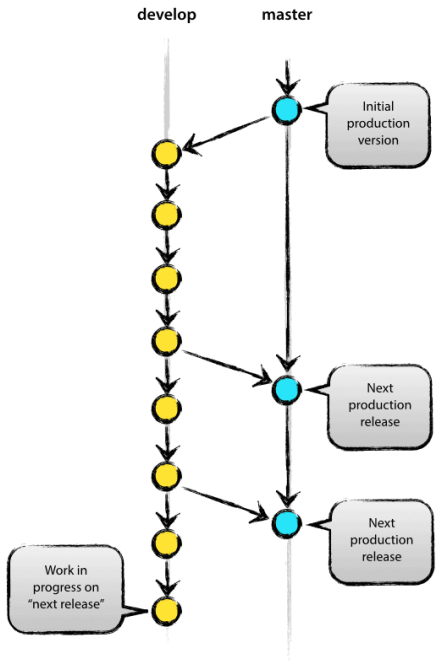
- il *master* è il branch in cui sarà presente il progetto pronto per la produzione e dove è collocato tutto il software già in uso di "On.TMS";

- il *portale_trasportatore* è il branch dove saranno presenti le ultime modifiche apportate al progetto per la prossima versione.

4.4 Build automation

Lo sviluppo di un software richiede lo svolgimento di una serie di attività come la compilazione del codice, l'esecuzione di test automatizzati, il deployment della soluzione e la documentazione di progetto.

Gli strumenti di *build automation* sono pensati per automatizzare tutte le fasi di sviluppo del software, perciò hanno il compito di convertire i file e le



[1] Figura 4.6: Diagramma che illustra la struttura dei branch nel progetto.

[1] https://nvie.com/posts/a-successful-git-branching-model/

risorse in un prodotto software nella sua forma finale. Il loro utilizzo è un prerequisito per un uso efficace della *Continuous Integration*, di cui ho parlato nei paragrafi precedenti. Le principali caratteristiche di questi strumenti sono:

- svolgono operazioni per il build dei file, cioè per la trasformazione dei file sorgente nella soluzione finale;

- organizzano le operazioni in task, ovvero in singole operazioni, per poter definire il flusso di svolgimento del processo build;

- gestiscono le dipendenze del software con altri progetti o librerie attraverso la rete, senza costringere il programmatore a dover mettere a disposizione tutti i sorgenti utilizzati. Ciò significa che, lo strumento di build automation scarica in automatico, durante la compilazione, le dipendenze sempre aggiornate o compatibili con il progetto realizzato, per poter far funzionare il programma.

Visti i numerosi vantaggi che il sistema build automation offre, la scelta di integrarlo e utilizzarlo all'interno del progetto è stata da subito presa in considerazione e infine attuata. In particolare, il sistema per l'automazione dello sviluppo scelto è stato Gradle.

Quest'ultimo è uno strumento di build automation open source, progettato per essere flessibile in modo da poter costruire quasi ogni tipo di software. Il motivo che ha indotto a utilizzare tale strumento all'interno del progetto risiede nelle sue caratteristiche:

- svolge le attività solo quando necessario, cioè quando input e output cambiano;

- rende facile la portabilità su piattaforme diverse in quanto si esegue sulla JVM;

- è possibile estenderlo per crearsi il proprio modello;

- è supportato da moltissimi IDE di sviluppo e interagisce con loro;

- permette subito di identificare i problemi di build;

4.5 Strumenti utilizzati

Gli strumenti utilizzati per sviluppare il codice, che saranno presentati qui di seguito, sono di due tipologie: l'IDE di sviluppo e le librerie esterne integrate nel progetto.

4.5.1 IDE di sviluppo

L'*Integrated Developement Evironment* (o semplicemente IDE) è un software che supporta lo sviluppo del codice sorgente di un programma, mettendo a disposizione una serie di strumenti e funzionalità utili allo sviluppo e al debugging.

Per lo sviluppo di questo progetto è stato scelto come IDE Visual Studio, si tratta di un ambiente di sviluppo integrato sviluppato da Microsoft.

Nelle più recenti versioni è nata la piattaforma .NET *[10]*, di cui io stessa ho usufruito per la realizzazione del software, che supporta diversi linguaggi di programmazione tra cui C#, il linguaggio scelto per la stesura del progetto. A differenza dei compilatori classici quello disponibile con il Framework .NET converte il codice scritto nell'IDE ovvero un *Intermediate Language* che rappresenta il nuovo linguaggio progettato per essere convertito in modo efficiente in codice macchina nativo. Si tratta quindi di un linguaggio di livello inferiore rispetto a C# o .NET ma allo stesso tempo di più alto livello rispetto all'assembly o al linguaggio macchina.

Questo ambiente di sviluppo integra perfettamente gli strumenti di build automation e fornisce un'interfaccia unificata per i principali sistemi di controllo di versione.

4.5.2 Framework

Nell'ambito dello sviluppo software, un *framework* rappresenta un'architettura logica di supporto su cui un software può essere progettato e realizzato, spesso facilitandone lo sviluppo da parte del programmatore.

È definito da un insieme di classi astratte e delle relazioni tra esse. Alla base di un framework esistono una serie di librerie di codice utilizzabili da

includere all'interno del proprio progetto. L'utilizzo di un framework impone dunque al programmatore una precisa metodologia di sviluppo software.

Qui di seguito, è riportato il framework utilizzato per la realizzazione delle pagine di front-end dell'applicativo.

Vue.js

Vue.js è un framework JavaScript *[11]*, dedicato alla realizzazione di *interfacce web reattive* che sfruttano il dual-binding tra modello dati e vista. Ciò significa che rende possibile implementare un'applicazione ragionando in termini di dati, variabili e oggetti, astraendosi rispetto all'implementazione e aggiornamento del DOM della pagina. Si differenzia per:

- performance
  implementa una modalità di rendering basato su *document fragment*

- leggibilità

- flessibilità

Vue.js implementa il pattern MVVM, una declinazione del più famoso Model-View-Controller (MVC).
I componenti di un'applicazione MVVM sono 3:

- il *Modello (Model)* che rappresenta l'implementazione del dominio dati gestito dall'applicazione, come per la 'M' in MVC;

- la *Vista (View)* che rappresenta il componente grafico renderizzato all'utente, composta da HTML e CSS;

- Il *Modello per la Vista (ViewModel)* che rappresenta il collante tra i precedenti componenti e fornisce alla view i dati in un formato

consono alla presentazione. e il comportamento di alcuni componenti dinamici

Capitolo 5

Implementazione

Successivamente alla progettazione e definizione dell'architettura, si è passati all'implementazione vera e propria del progetto. Durante questa fase è stato realizzato il sistema fisico nel rispetto delle scelte prese in precedenza. Un primo passo per avviare la fase implementativa ha riguardato la scelta del linguaggio di programmazione nonché C#, usato anche per il resto del software "On.TMS". Lato client, si è deciso di integrare al linguaggio HTML, CSS e JavaScript, l'utilizzo del framework Vue.js per ottenere un'interfaccia estremamente reattiva.

L'implementazione ha previsto lo svolgimento di più fasi, che hanno permesso di ottenere come risultato finale il software inerente alla nuova feature.

Qui di seguito saranno messe in evidenza le principali scelte strategiche che hanno permesso di sviluppare le funzionalità del sistema richieste.

5.1 Test sui casi d'uso

Una prima parte della fase implementativa è stata occupata dalla realizzazione dei test riguardanti i casi d'uso definiti nei capitoli precedenti, che ha occupato

una buona parte del lavoro di sviluppo. Si è trattato di un percorso incrementale dove, a partire da piccoli e banali test, si è costruita l'intera logica di sviluppo. Il processo di fatti ha seguito la struttura TDD (*Test Driven Development*), secondo la quale in base alle funzionalità e specifiche che sarà necessario implementare, si creano prima i test e poi il codice sorgente.

Nel nostro caso specifico, ad ogni test è stato associato un controllo o *Assert* che ne determina l'esito finale, true o false, del test stesso.

A partire da ognuno di questi, sono state create le classi di cui è necessario usufruire affinché il risultato di quel test sia verificabile. Complessivamente l'intero sviluppo consiste quindi nella creazione di un test associato a un caso d'uso tra quelli presi in esame, all'implementazione dell'Assert finale e le classi di cui usufruire.

Nello specifico la fase riguardante la validazione dei test verrà analizzata nel Capitolo 6 denominato Sperimentazione. Questo tipo di scelta consiste quindi nella definizione iniziale di numerosi test con i quali si sviluppa sempre più il sistema nella sua interezza, a loro volta rappresenteranno test fondamentali per la verifica e la sperimentazione finale del software realizzato.

5.2 Introduzione ai service e comandi

Dopo aver impostato i test sopra descritti ci si è soffermati sulla verifica della correttezza di questi ultimi usufruendo della persistenza dei dati mediante il database non relazionale RavenDB.

Riprendendo i casi d'uso di cui si è beneficiato anche nelle fasi precedenti sono stati definiti i comandi principali di cui sia l'utente dipendente dell'azienda produttrice sia il trasportatore avrebbero usufruito. Quando si parla di comandi si vuole intendere quelle azioni consentite all'utente attraverso l'interfaccia grafica e il cui risultato richiede una aggiunta, una modifica o una eliminazione dei dati all'interno del database.

Anche in questo caso, qui di seguito, verranno riportate le definizioni associate ai comandi:

- **ConfiguraCalendario**

Questo primo comando nasce a partire dall'interazione tra l'utente dell'azienda e il calendario da configurare all'interno del sistema

"On.TMS". Le componenti fondamentali di cui si fa riferimento corrispondo all'ampiezza e alla capacità che all'utente viene richiesto di specificare per ogni slot selezionato all'interno del calendario di base. In questo caso al comando è associato l'inserimento di un nuovo documento all'interno del database o la modifica di quest'ultimo con nuovi valori inseriti dall'utente.

- **PrenotaSlot**

Un secondo caso appartenente al portale trasportatore, riguarda l'interazione tra il trasportatore stesso e la pagina di ModificaDettaglio di un determinato viaggio. In questo caso la prenotazione è resa possibile dall'interfaccia raffigurante il calendario avente slot temporali liberi e prenotabili. Il comando corrispondente, a partire da una data specifica, richiama dal database il documento *PrenotazioniInGiorno* avente come Id la data stessa, nel caso in cui ancora non esista ne crea uno nuovo e inserisce i dati della prenotazione dello slot, infine salva tutto sul database e aggiorna eventuali modifiche di dettaglio.

- **AnnullaSlotPrenotato**

Similmente alla prenotazione di uno slot, viene gestito l'annullamento di quest'ultimo. In questo caso viene recuperato dal database il documento *PrenotazioniInGiorno* avente come Id la data dello slot da annullare, si cerca tra la lista di slot prenotati in quel giorno quello che il trasportatore vuole annullare e lo si elimina dal database.

5.2.1 Query di ricerca

Fino a ora si è parlato di comandi che richiedono la modifica dei dati a livello di database, mentre spesso per il processo di un sistema è necessario usufruire di query che restituiscono dati reperiti sempre dal database ma sui quali non si effettua alcuna modifica.

Qui di seguito sono specificate alcune tra quelle di cui si è usufruito:

- **CercaPrenotazione**

Questa query nasce a partire dall'idea che sia l'utente che il trasportatore possano cercare tra le prenotazioni una in particolare. Per questo motivo in base alla specifica di un giorno preciso è possibile caricare dal database la prenotazione di uno slot contenuto all'interno del documento *PrenotazioniInGiorno*.

- **CercaSlotLiberiInUnGiorno**

Per la ricerca di slot liberi a partire dall'inserimento di una data in un calendario, si è pensato all'azione del trasportatore per la prenotazione di uno slot libero per l'accettazione di un viaggio. A partire dalla data e dal valore della capacità del mezzo con il quale il trasportatore prenderà in carico il viaggio, è possibile filtrare tra i dati persistenti nel database quegli slot che risulteranno ancora liberi e disponibili per essere prenotati.

5.3 Test sui comandi

Dopo aver definito i comandi e le query sopra analizzate, è stato necessario verificare che i dati fossero correttamente manipolati all'interno del database mediante l'implementazione di nuovi test di verifica.

Anche in questo caso ogni test è composto da un assert finale il quale determinerà il corretto funzionamento dello stesso. Per ognuno di questi test, prima di chiamare il comando di cui si vuole testare la persistenza è necessario configurare il calendario si cui poi saranno effettuate le modifiche.

I test di verifica effettuati sono i seguenti:

- **Verifica_configurazione_calendario()**

Questo primo test consiste nella verifica di avvenuta configurazione del calendario a partire dalla chiamata al comando ConfiguraCalendario con la specifica dei valori di ampiezza e capacità inseriti dall'utente.

- **Verifica_prenotazione_slot()**

Il seguente test permette di verificare che il comando di prenotazione, da parte di un trasportatore all'interno del portale, vada effettivamente a buon fine. Questo tipo di verifica viene effettuata mediante la chiamata a un Assert.NotNull() a cui viene passato il documento di *PrenotazioniInGiorno*

ricavato dal database e avente come identificativo la data della prenotazione slot effettuato.

- **Verifica_annullamento_slot()**

Similmente alla prenotazione di uno slot, al trasportatore è permesso annullare uno slot precedentemente prenotato. La verifica corrispondente al comando di annullamento usufruisce di una query finale di ricerca, nonché *CercaPrenotazione*, il cui risultato sarà inserito in un Assert.False() che controlla la corretta inesistenza della prenotazione.

- **Verifica_slot_liberi()**

Questo tipo di test è stato realizzato con la volontà di voler testare per numerose prenotazioni la persistenza del calendario in relazione alle prenotazioni effettuate in precedenza, il controllo ha richiesto l'utilizzo oltre che della prenotazione di slot anche del valore della capacità del mezzo con il quale il trasportatore ha scelto di portare a termine i viaggi a lui assegnati.

5.4 Interfaccia grafica

Come ultima fase implementativa ci si è soffermati sulla grafica del software e la popolazione di essa con i dati prelevati dal database, usufruendo dei comandi analizzati nel paragrafo precedente.

Anche in questo caso è stato scelto di procedere in maniera incrementale e similmente allo sviluppo dei test, di seguito sarà visibile lo sviluppo degli elementi grafici secondo il criterio da noi adottato:

- **Configurazione calendario**

  Vedi Figura 5.1, Figura 5.2, Figura 5.3 e Figura 5.4.

Gli utenti, nonché dipendenti, incaricati per la configurazione del calendario aziendale dovranno loggarsi come amministratori all'interno del software "On.TMS" precisamente nel portale traportatore. Troveranno poi la selezione *Configurazione* attraverso la quale potranno accedere alla pagina ConfiguraCalendario mediante la quale modificare il calendario secondo a caratteristiche precise per ogni unità temporale.

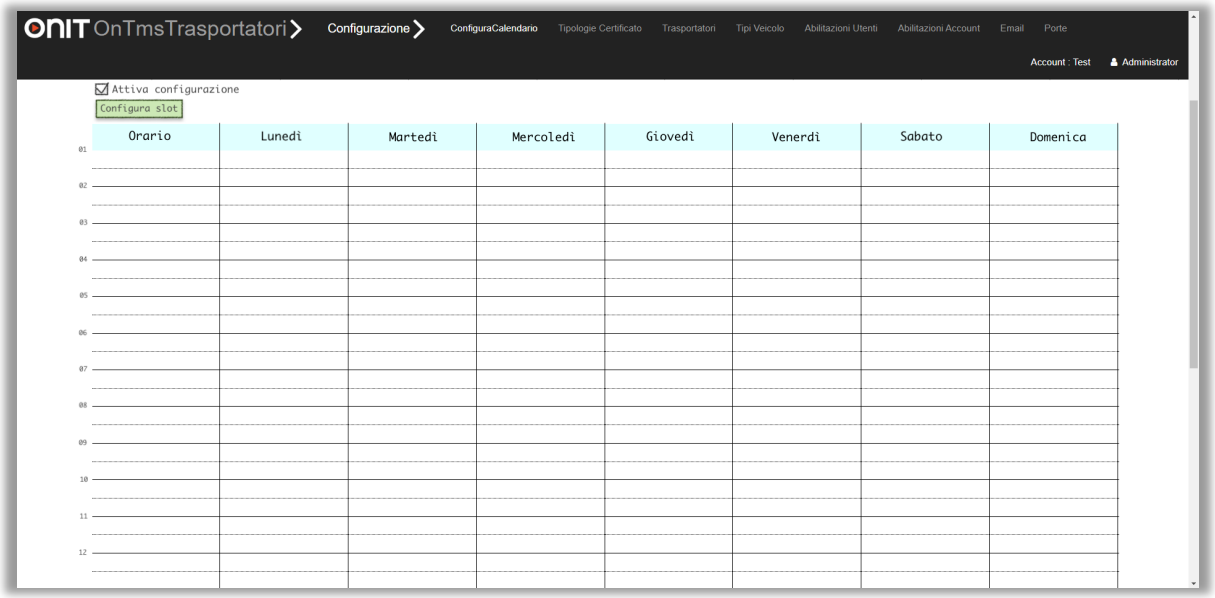


Figura 5.1: Sw "On.TMS" Portale Trasportatore – Selezione Configurazione Calendario.

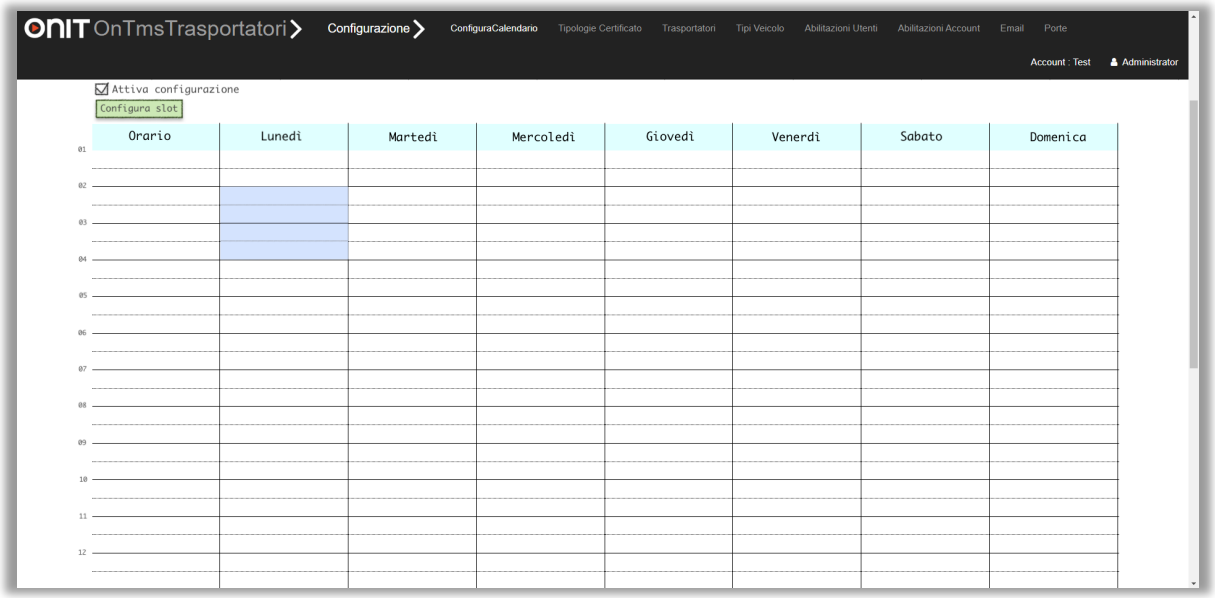


Figura 5.2: Sw "On.TMS" Portale Trasportatore – Selezione due unità di prenotazione.

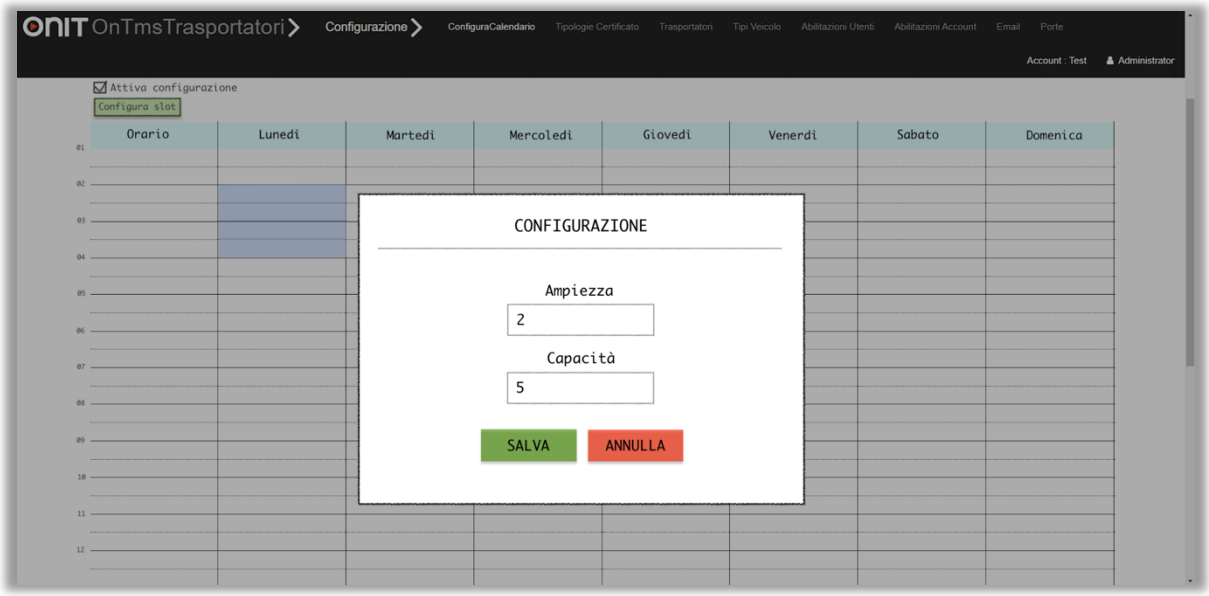


Figura 5.3: Sw "On.TMS" Portale Trasportatore – Modifica configurazione unità selezionate.

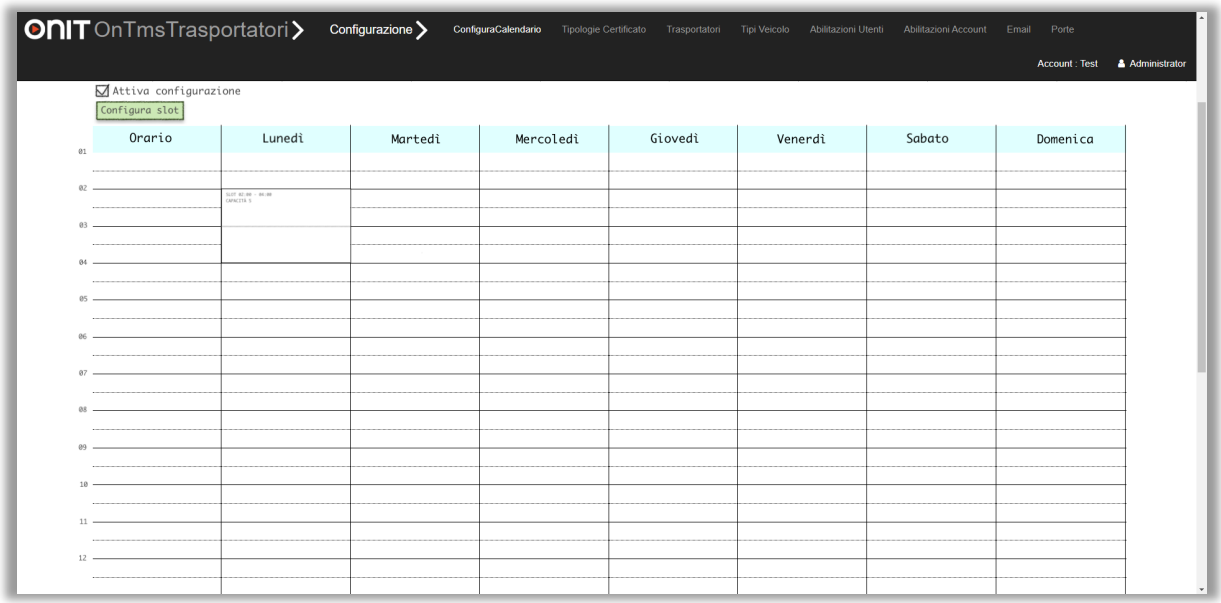


Figura 5.4: Sw "On.TMS" Portale Trasportatore – Nuova visualizzazione slot configurato.

- **Visualizzazione slot liberi nel portale trapostatori:**
  Vedi Figura 5.5, Figura 5.6 e Figura 5.7.

Il trasportatore potrà visualizzare la lista di viaggi ad egli assegnati. Per ognuno di questi selezionerà il *Dettaglio* in modo da poter modificare o selezionare una serie di componenti tra le quali lo slot temporale nel quale prenotarsi. All'interno della pagina di dettaglio viaggio sarà necessario che il trasportatore selezioni una data nel calendario e conseguentemente verrà popolata una lista di slot temporali aventi colori differenti in base alla disponibilità di prenotazione. Ogni slot renderà visibile solo la propria fascia oraria. Una volta effettuata la selezione sarà necessario salvare le modifiche.

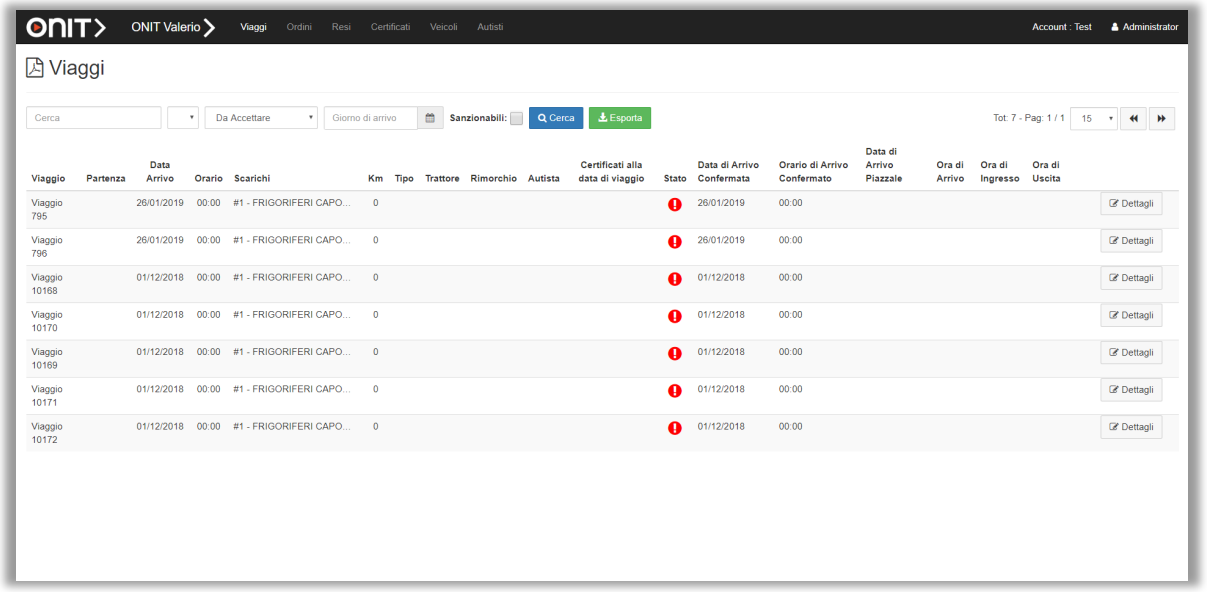


Figura 5.5: Sw "On.TMS" Portale Trasportatore – Selezione Dettaglio viaggio.

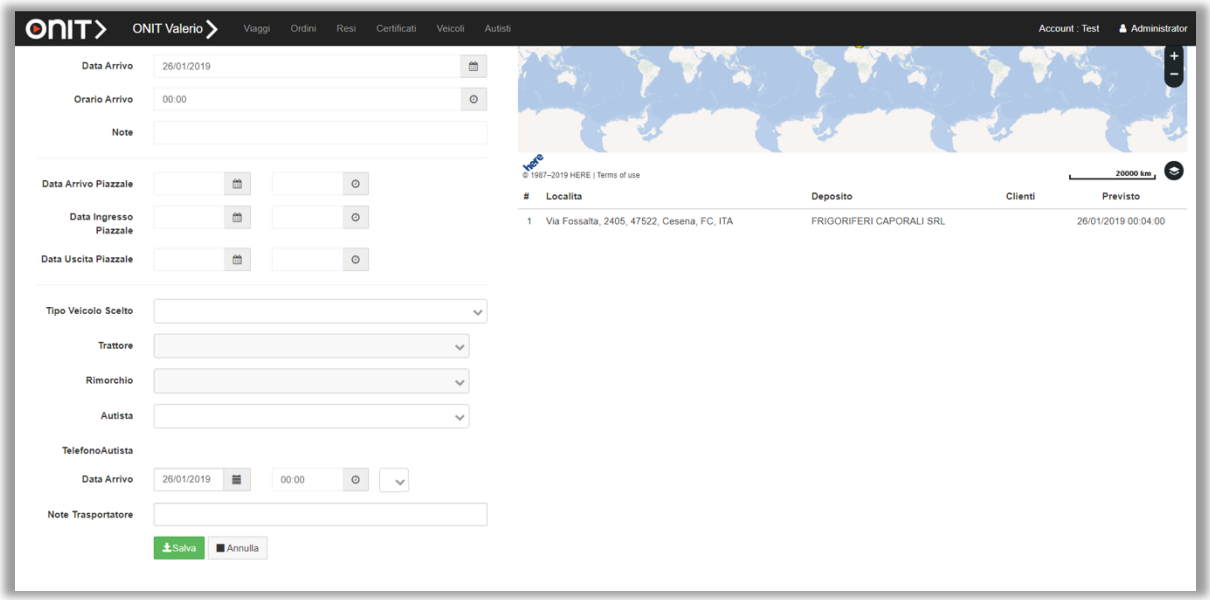


Figura 5.6: Sw "On.TMS" Portale Trasportatore – Selezione dettaglio viaggio prenotazione Slot.

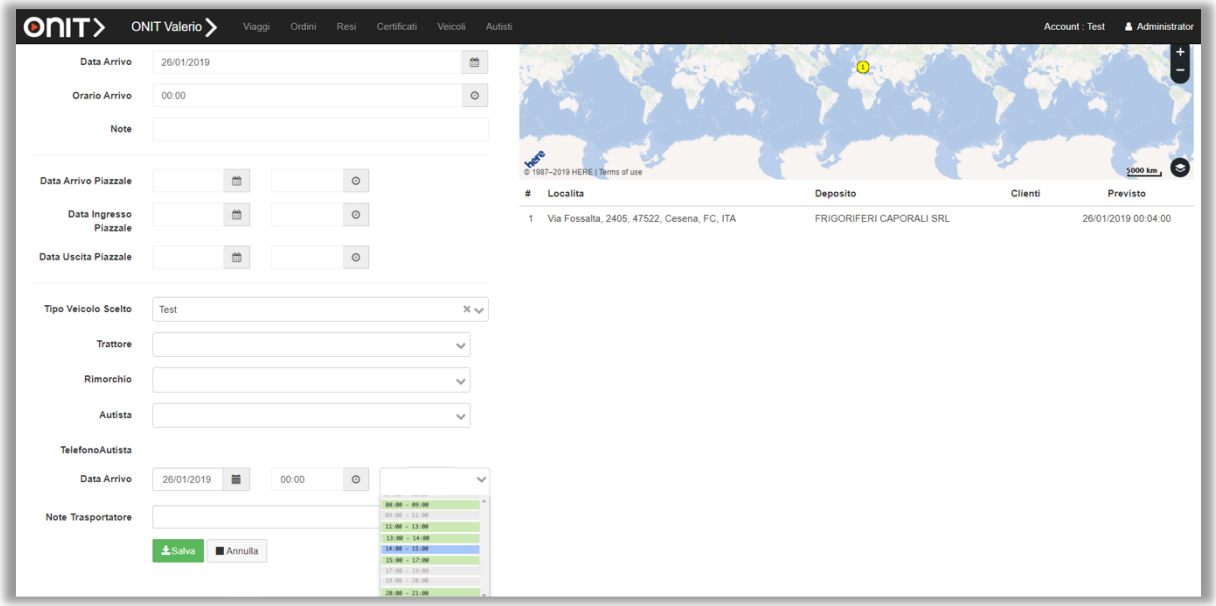


Figura 5.7: Sw "On.TMS" Portale Trasportatore – Prenotazione Slot.

- **Visualizzazione calendario prenotazioni per l'azienda produttrice**

Ogni azienda produttrice avrà la possibilità di visualizzare nel calendario la lista di prenotazioni settimana per settimana, in base alle caratteristiche del calendario configurato. Si tratta di una funzionalità in fase terminale di sviluppo per la quale però non è stato possibile visualizzare in maniera corretta l'output grafico. Complessivamente il caso d'uso risulta simile a quello studiato in precedenza, sarà costituito da una pagina di visualizzazione calendario nella quale saranno collocate le prenotazioni in una griglia settimanale come per la configurazione calendario, così che ogni azienda possa visualizzare le prenotazioni per ogni slot temporale.

Capitolo 6

Sperimentazione

La progettazione e la realizzazione di un software possono portare a ottenere idealmente un buon progetto, ma non è detto che, nella pratica, tale software sia effettivamente efficiente soddisfi e requisiti per cui è stato pensato.

Per mostrare l'effettiva applicabilità del progetto riguardante la prenotazione di slot temporali per il carico e lo scarico della merce in magazzino, sono stati eseguiti dei test volti a verificare le operazioni di base: lo scopo è quello di vedere come il sistema reagisce in base a determinate selezioni o specifiche di configurazione.

Qui di seguito verranno presentati i principali assert di verifica con i quali è stato testato il progetto e i risultati ottenuti da tale sperimentazione.

È importante specificare che questi test sono stati utili durante la fase di sviluppo iniziale per la verifica dei requisiti e delle funzionalità del sistema, ma la loro utilità potrà persistere anche in seguito alla messa in produzione del software in caso vengano segnalati problemi da parte dei committenti di cui ne faranno uso.

6.1 Validazione Test

La sperimentazione inerente allo sviluppo della nuova feature all'interno di "On.TMS" ha occupato gran parte della fase iniziale di progettazione ma non solo, ha richiesto numerose modifiche al codice e hai test inizialmente definiti.

Qui di seguito saranno analizzati le validazioni finali ottenute:

- **Configura_slot1h_calendario_di_una_settimana()**

Uno dei primi test definiti nasce a partire dalla volontà di verificare che, dopo la configurazione di un oggetto di tipo *calendario*, le prenotazioni fossero del tutto vuote usufruendo della verifica del tipo Assert.Empty().

- **Verifica_slot_liberi_in_base_alla_capacita_mezzo()**

Questo test nasce a partire dalla necessità di verificare che ogni trasportatore in base al mezzo con il quale scelgono di accettare un viaggio nel portale, possa visualizzare in calendario solo gli slot ancora disponibili aventi capacità maggiore o uguale alla capacità del mezzo di trasporto. In questo secondo test viene chiesto di verificare se un oggetto di tipo *slot*, di cui verranno specificate la data, l'ora di inizio e quella di fine e altre proprietà ancora, risulta libero o viceversa occupato in base alla capacità del mezzo di trasporto con il quale il trasportatore ha scelto di prendere in carico il viaggio. È importante ricordare che, in ognuno di questi test, prima di effettuare i controlli e le verifiche finali è necessario richiamare sempre l'oggetto *calendario* e la sua configurazione mediante la specifica di due campi nonché ampiezza e capacità con cui configurare gli slot temporali.

- **Prenota_uno_slot()**

Come definisce il nome del test stesso, si è preso in considerazione il caso d'uso in cui il trasportatore per confermare la presa in carico di un viaggio a lui assegnato, prenota uno slot disponibile nel calendario. A livello di codice nel test si chiama un nuovo oggetto *slotInPrenotazione* che conterrà una lista di slot prenotati in una certa data e con orari di inizio e fine specifici. A questo punto il controllo finale usufruisce di un Assert.Equals() che effettua la verifica corrispondente a quel preciso slot temporale o meglio controlla che effettivamente la capacità nel calendario corrispondente allo slot prenotato, sia pari alla capacità iniziale diminuita del valore della capacità corrispondente al mezzo di trasporto con il quale il trasportatore ha effettuato la prenotazione. Anche in questo caso, come per gli altri test, è importante specificare che l'uso delle classi più comuni denominate *calendario, slotConfigurato o slotInPrenotazione* è affiancato da altre classi come *Booking o Regole* che a loro volta consentono, mediante

metodi specifici, di effettuare controlli e verifiche affinché le operazioni più comuni siano possibili.

- **Prenota_slot_periodici_di_una_settimana()**

Nel caso della prenotazione di slot di una intera settimana si riproduce un comportamento simile al test di prenotazione di uno slot, usufruendo di un ciclo che aggiunga tale prenotazione per ogni giorno di un'interna settimana. Il test finale consiste nella verifica dell'effettiva prenotazione di slot temporali a partire dalla configurazione iniziale del calendario.

- **Annulla_prenotazione_slot()**

Similmente al test di prenotazione slot, era necessario definire un test di annullamento di prenotazione. In questo caso si parte sempre con la configurazione di un calendario mediante la specifica dei valori di ampiezza e capacità per ogni slot, si prenota uno slot disponibile in una certa data e fascia oraria, infine si annulla la prenotazione dello stesso eliminandolo dalla lista di slot prenotati e si verifica che la capacità del calendario per quello slot sia uguale alla capacità iniziale configurata usufruendo di un Assert.Equal().

Capitolo 7

Conclusioni

Il fine ultimo del lavoro svolto in questa tesi è quello di inglobare la nuova funzionalità software inerente alla gestione piazzale in "On.TMS", con l'obiettivo di portarla in produzione nei sistemi TMS dei clienti che l'hanno commissionata. Questa funzionalità dovrà occuparsi della prenotazione di slot temporali collocati in un calendario configurato dalle aziende produttrici. Questo nuovo sistema porterà una serie di migliorie tra le quali: una organizzazione vantaggiosa per la gestione del carico e scarico merce nei magazzini, una diminuzione dei tempi di attesa e degli afflussi contemporanei che si verificano durante le ore di punta nell'arco della giornata, una gestione più automatizzata ed efficiente in caso di imprevisti.

I requisiti definiti in fase di analisi sono stati tutti soddisfatti, permettendo di ottenere un primo prototipo software funzionante e conforme alle necessità richieste dai committenti. L'utilizzo degli strumenti, tecnologie e pattern architetturali adottati renderanno possibile l'estensione e la modifica del sistema per eventuali sviluppi futuri da parte del team TMS di Onit.

L'interfaccia grafica del software è stata realizzata seguendo le linee guida già utilizzate per la realizzazione dell'applicativo software "On.TMS", ottenendo così un'interfaccia user-friendly, semplice e intuitiva.

Il cuore pulsante di tutto il progetto consiste nei numerosi test di verifica che hanno affiancato l'intero sviluppo progettuale, soprattutto per la parte riguardante la configurazione del calendario in relazione alle prenotazioni di slot temporali ancora disponibili.

Complessivamente il progetto non è da considerarsi totalmente concluso e pronto alla messa in produzione in quanto è ancora necessario

sperimentare la funzionalità dello stesso sotto nuovi aspetti. Arrivati a questa fase è comunque possibile individuare eventuali sviluppi futuri anche in relazione all'uso che ne faranno i committenti.
In conclusione, possiamo dire che il lavoro di tesi svolto ha portato alla realizzazione di un sistema vicino alla distribuzione al pubblico, perché mette a disposizione le funzionalità base richieste e utili a migliorare le condizioni di gestione del carico e scarico merce nei magazzini delle aziende produttrici. Grazie ai feedback ricevuti dagli ingegneri di Onit del team trasporti, possiamo anche affermare che l'esecuzione è risultata efficiente e che l'obiettivo iniziale è stato raggiunto.

Bibliografia

[1] onit.it– “Soluzioni Informatiche per l’Industria” [Online]
URL: https://www.onit.it/wp-content/uploads/2018/09/brochure_AREA_INDUSTRIA.pdf

[2] smet.it – “Che cos’è la logistica e come si articola nella gestione aziendale” [Online]
URL: https://www.smet.it/blog/logistica/

[3] logisticaefficiente.it – “Perché la logistica dei trasporti è vitale” [Online]
URL: https://www.logisticaefficiente.it/flexis/network-e-trasporti/gestione-trasporti/perche-logistica-trasporti-vitale.html

[4] onit.it – ““On.TMS” (Transportation Management System)” [Online]
URL: https://www.onit.it/industria/trasporti-tms-software-di-gestione-trasporti

[5] <html>.it – “Il pattern MVC” [Online]
URL: https://www.html.it/pag/18299/il-pattern-mvc/

[6] martin.Fowler.com – “Agile Software Guide” [Online]
URL: https://martinfowler.com/agile.html

[7] martin.Fowler.com – “The Practical Test Pyramid” [Online]
URL: https://martinfowler.com/articles/practical-test-pyramid.html

[8] martin.Fowler.com – “Continuos Integration” [Online]
URL: https://martinfowler.com/articles/continuousIntegration.html

[9] Git - Fast-version-control, “Il controllo di versione” [Online]
URL: https://git-scm.com/book/it/v1/Per-Iniziare-Il-Controllo-di-Versione

[10] .NET – “ASP.NET MVC Pattern” [Online]
URL: https://dotnet.microsoft.com/apps/aspnet/mvc

[11] Vue.js – “The Progressive JavaScript Framework” [Online]
URL: https://vuejs.org/v2/guide/

[12] Git - Fast-version-control, “Basi di Git” [Online]
URL: https://git-scm.com/book/it/v1/Per-Iniziare-Basi-di-Git

[13] martin.Fowler.com – “Integration Test” [Online]
URL: https://martinfowler.com/bliki/IntegrationTest.html

Ringraziamenti

Arrivata al termine di questo percorso di laurea triennale vorrei ringraziare tutti coloro che mi hanno sostenuto, aiutato e incoraggiato ad andare avanti per raggiungere questo traguardo.

In primo luogo, ringrazio il professore e relatore Mirko Viroli per avermi seguito durante la realizzazione di questo progetto. Lo ringrazio soprattutto per la sua disponibilità, per l'aiuto e i consigli ricevuti durante il corso di questa esperienza.

Un ringraziamento va anche all'azienda Onit Group che mi ha permesso di partecipare alla realizzazione di questo nuovo progetto in ambito TMS. Nello specifico ringrazio il responsabile dell'Area Industria Claudio Gambetti, il responsabile del team nonché capo progetto e correlatore di questa tesi Valerio Borioni e tutto il resto dei colleghi del team con i quali ho condiviso mesi di lavoro e che sono sempre stati disponibili e pazienti.