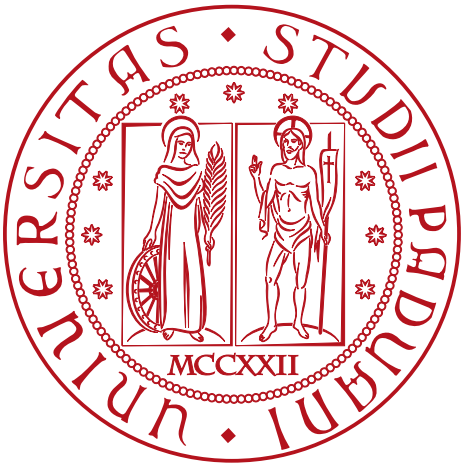
Università degli studi di Padova

Dipartimento di Matematica "Tullio Levi-Civita"

Corso di Laurea in Informatica



VoIPDashboard - Strumento web per la visualizzazione e l'analisi di dati telefonici

Tesi di laurea triennale

Relatore

Prof. Marco Zanella

Laureando

Francesco Fragonas

Matricola 2076436

ANNO ACCADEMICO 2024-2025

Sommario

Il presente documento descrive il lavoro svolto durante il periodo di stage curricolare, della durata di circa trecentoventi ore, dal laureando Francesco Fragonas presso l'azienda Cinquenet SRL. Lo stage è stato condotto sotto la supervisione del tutor aziendale Fabio Baraldo, mentre il prof. Prof. Marco Zanella ha ricoperto il ruolo di tutor accademico.

Ringraziamenti

TESTO RINGRAZIAMENTI

Padova, Dicembre 2025 *Francesco Fragonas*

Indice

Elenco delle Figure

Elenco delle Tabelle

Elenco dei Codici Sorgente

Capitolo 1

Introduzione

1.1 L'azienda

Cinquenet S.r.l. è un'azienda specializzata nel settore delle Information and Communication Technology (ICT) con sede a Cerea, in provincia di Verona. Fondata da un team di professionisti con oltre vent'anni di esperienza consolidata nel settore delle telecomunicazioni, l'azienda si distingue nel panorama delle soluzioni tecnologiche per la combinazione di competenza tecnica, passione e attenzione al dettaglio. La filosofia aziendale di Cinquenet si basa sullo sviluppo di soluzioni ICT all'avanguardia, progettate per garantire elevata affidabilità operativa e stabilità nel tempo. Questo approccio ha permesso all'azienda di costruire una solida reputazione nel territorio veneto e di espandere la propria presenza nel mercato delle telecomunicazioni e dei servizi informatici.

1.1.1 Aree di specializzazione

L'offerta di Cinquenet si articola in tre macro-aree di competenza, che riflettono un approccio integrato alle esigenze tecnologiche delle aziende moderne:

Soluzioni telefoniche e di rete:

Il core business dell'azienda comprende la progettazione e implementazione di infrastrutture di telecomunicazione avanzate. Tra i servizi principali figurano la realizzazione di reti in fibra ottica, collegamenti ADSL, linee professionali dedicate, centralini virtuali e impianti telefonici aziendali. Particolare expertise viene dedicata alle soluzioni wireless e ai link radio, tecnologie fondamentali per garantire connettività in contesti dove le infrastrutture tradizionali risultano insufficienti.

Servizi informatici e cloud:

Cinquenet offre una gamma completa di soluzioni per la gestione dell'infrastruttura IT aziendale, includendo servizi di hosting, server cloud, backup dei dati e consulenza informatica specializzata. L'azienda si è inoltre posizionata come partner strategico per l'implementazione di

tecnologie emergenti quali Internet of Things (IoT) e soluzioni basate su intelligenza artificiale.

Sicurezza e protezione:

Un'area di crescente importanza nel portfolio aziendale è rappresentata dalla cybersecurity, settore nel quale Cinquenet sviluppa strategie di protezione integrate per la sicurezza informatica. Parallelamente, l'azienda opera nel campo della sicurezza fisica attraverso l'installazione di sistemi di videosorveglianza e impianti antintrusione.

1.2 Motivazone del progetto

Ho scelto di svolgere lo stage presso Cinquenet srl per diverse ragioni che rendevano questa opportunità particolarmente interessante dal punto di vista formativo e professionale. Innanzitutto, conoscevo già l'azienda e il suo approccio lavorativo, il che mi ha permesso di valutare positivamente l'ambiente e le metodologie operative.
La decisione di non optare per un'azienda tradizionalmente focalizzata sulla programmazione è stata dettata dal desiderio di ampliare le mie competenze in un contesto più diversificato. Mi intrigava l'idea di lavorare in un'azienda operante nel mondo dell'informatica ma con un focus specifico su reti, connessioni e centralini virtuali PABX, settori che offrono prospettive di crescita professionale complementari allo sviluppo software puro.
Il progetto di stage consiste nello sviluppo di una dashboard web che consente ai clienti di accedere a tutte le statistiche e i dettagli delle chiamate effettuate e ricevute. La piattaforma offre funzionalità di filtraggio avanzate per interno, ring group, DID e periodi temporali, presentando i dati attraverso statistiche, grafici e tabelle dettagliate. Tutti i contenuti sono progettati per essere facilmente stampabili ed esportabili in formato PDF e CSV.
Questo progetto rappresenta un'opportunità ideale per combinare competenze di programmazione web con la conoscenza del mondo dei centralini virtuali e delle telecomunicazioni aziendali, offrendo un'esperienza formativa completa e multidisciplinare.

1.3 Struttura della tesi

Il presente documento è strutturato secondo la seguente organizzazione:

Il secondo capitolo presenta una descrizione dettagliata dello stage, includendo l'organizzazione del lavoro, il rapporto con l'azienda e

il tutor aziendale, la metodologia adottata e l'analisi preventiva dei rischi.

Il terzo capitolo contiene l'analisi approfondita dei requisiti del sistema, suddivisi per tipologia e priorità, insieme all'identificazione degli stakeholder e dei casi d'uso principali.

Il quarto capitolo fornisce un'introduzione teorica alle tecnologie e agli strumenti utilizzati, presentando le motivazioni alla base delle scelte architetturali e tecnologiche adottate.

Il quinto capitolo descrive nel dettaglio il lavoro svolto durante il periodo di stage, illustrando le problematiche incontrate, le soluzioni implementate e le funzionalità sviluppate.

Il sesto capitolo presenta le conclusioni dell'esperienza, valutando il raggiungimento degli obiettivi prefissati, le conoscenze acquisite e i possibili sviluppi futuri del progetto.

Convenzioni tipografiche:

Per la stesura del documento sono state adottate le seguenti convenzioni:

- Gli acronimi, le abbreviazioni e i termini tecnici specialistici sono definiti nel glossario posto al termine del documento;
- La prima occorrenza dei termini presenti nel glossario è evidenziata con la seguente notazione: termine$\hat{}$G$\hat{}$;
- I termini in lingua straniera e il gergo tecnico sono riportati in corsivo.

Capitolo 2

Descrizione stage

In questo capitolo viene approfondita l'organizzazione dello stage, descrivendo il progetto realizzato, la metodologia di lavoro adottata, il rapporto con l'azienda e l'analisi preventiva dei rischi.

2.1 Introduzione al progetto

Il progetto di stage si inserisce nel contesto dei servizi di telefonia aziendale offerti da Cinquenet ai propri clienti. L'azienda fornisce soluzioni di centralini virtuali PABX che gestiscono le comunicazioni telefoniche di numerose realtà aziendali. Tuttavia, mancava uno strumento che permettesse ai clienti finali di analizzare autonomamente i dati relativi alle proprie chiamate telefoniche.
L'obiettivo del progetto è stato lo sviluppo di una dashboard web che consentisse ai clienti di visualizzare statistiche dettagliate sulle chiamate effettuate e ricevute, con la possibilità di applicare filtri avanzati e generare report personalizzati. Il sistema doveva integrarsi con l'infrastruttura esistente dei centralini PABX, estraendo i dati dalle basi dati operative e presentandoli in formato accessibile e intuitivo.
La realizzazione è avvenuta completamente ex novo, progettando sia il backend per l'elaborazione dei dati che il frontend per la visualizzazione. La sfida principale è consistita nel comprendere la struttura complessa dei dati telefonici, identificare le informazioni rilevanti e progettare query efficienti per l'estrazione e l'aggregazione delle statistiche.
L'approccio scelto prevedeva la realizzazione di un MVP (Minimum Viable Product) funzionante nel minor tempo possibile, che includesse già le funzionalità base del sistema, anche se non ancora completamente ottimizzate. Questo prototipo iniziale sarebbe poi servito come base per successive iterazioni di miglioramento e ampliamento.

2.2 Organizzazione del lavoro

Per lo sviluppo del progetto è stato adottato il Modello di Sviluppo Evolutivo, una metodologia particolarmente adatta quando i requisiti

non sono completamente definibili a priori e si desidera ottenere rapidamente versioni utilizzabili del sistema.

2.2.1 Il Modello di Sviluppo Evolutivo

Il Modello Evolutivo è un approccio incrementale in cui gli incrementi successivi costituiscono versioni prototipali utilizzabili e valutabili dagli stakeholder. A differenza di modelli sequenziali rigidi, questo approccio permette di:

- Rispondere a bisogni non inizialmente preventivabili: durante lo sviluppo possono emergere nuovi requisiti o modifiche a quelli esistenti
- Produrre prototipi utilizzabili: ogni iterazione rilascia una versione funzionante del sistema che può essere testata e valutata
- Ammettere iterazioni multiple: ogni fase può essere riattraversata più volte per raffinamenti successivi
- Gestire l'incertezza: particolarmente utile quando la complessità del dominio applicativo richiede esplorazione e apprendimento progressivo

Schema generale del Modello Evolutivo:

Il processo di sviluppo si articola in tre fasi principali:

1. Analisi preliminare
Questa fase iniziale è dedicata all'identificazione dei requisiti fondamentali e alla definizione dell'architettura di base del sistema, progettata per essere modulare e facilitare future evoluzioni. Viene inoltre pianificato il percorso di sviluppo suddividendo il lavoro in passi incrementali, e si procede con uno studio approfondito del dominio applicativo, in particolare della struttura dati dei sistemi PABX e della logica delle chiamate telefoniche.

2. Analisi e realizzazione iterativa
Il sistema viene progressivamente costruito attraverso cicli iterativi in cui l'analisi viene continuamente raffinata in base alle conoscenze acquisite. Ogni iterazione comprende progettazione, codifica e testing delle funzionalità, seguita dall'integrazione dei componenti sviluppati. Al termine di ogni ciclo, il lavoro viene validato attraverso sessioni di feedback con il tutor aziendale per verificare l'aderenza ai requisiti e identificare eventuali necessità di miglioramento.

3. Rilascio di prototipi
Ogni iterazione produce una versione funzionante del sistema che viene valutata dal tutor aziendale e, nelle fasi più mature, dal cliente finale. I

feedback raccolti guidano le iterazioni successive, orientando lo sviluppo verso le reali esigenze degli utilizzatori. Questo processo ciclico continua fino al raggiungimento di un livello di maturità soddisfacente per l'accettazione finale.

2.2.2 Applicazione del modello al progetto

Il lavoro è stato organizzato in sprint settimanali, cicli di sviluppo della durata di una settimana ciascuno, strutturati secondo le seguenti fasi:

Inizio sprint - Riunione di pianificazione
Ogni sprint iniziava con un incontro con il tutor aziendale in cui veniva effettuata una revisione del lavoro svolto nella settimana precedente, identificando eventuali criticità o problematiche emerse. Successivamente si procedeva alla definizione degli obiettivi dello sprint corrente e alla pianificazione dettagliata delle attività da svolgere.

Durante lo sprint
La fase centrale dello sprint era dedicata allo sviluppo vero e proprio delle funzionalità pianificate, accompagnato da attività continue di testing e debug per garantire la qualità del codice. Durante questa fase erano frequenti confronti informali con il tutor per risolvere dubbi tecnici o richiedere chiarimenti.

Fine sprint - Riunione di review
Al termine dello sprint veniva organizzata una sessione di review in cui le funzionalità implementate venivano dimostrate al tutor aziendale. Durante questo incontro si raccoglievano feedback e suggerimenti, si valutava il raggiungimento degli obiettivi prefissati e si identificavano le priorità per lo sprint successivo.

Nella fase iniziale del progetto, il supporto del tutor aziendale è stato fondamentale per acquisire le conoscenze necessarie sul dominio applicativo. Il tutor ha dedicato tempo significativo a spiegare la logica di funzionamento dei centralini PABX, illustrare la struttura del database e definire le modalità corrette di estrazione e interpretazione dei dati telefonici. Con il progredire dello stage e l'acquisizione di maggiore autonomia operativa, il ruolo del tutor si è progressivamente evoluto da formativo a consulenziale, focalizzandosi principalmente sulla validazione delle scelte progettuali e sulla definizione di nuovi requisiti emergenti.

2.3 Vincoli

Lo sviluppo del progetto è stato soggetto a diversi vincoli:

Vincoli temporali
- Durata complessiva dello stage: 320 ore
- Necessità di produrre un prototipo dimostrabile entro le prime settimane
- Scadenze settimanali per il completamento degli sprint

Vincoli tecnologici
- Integrazione obbligatoria con l'infrastruttura esistente di Cinquenet
- Utilizzo del database già in uso per i centralini PABX
- Requisiti di performance per la gestione di grandi volumi di dati storici

Vincoli architetturali
- Necessità di un'architettura modulare per future estensioni
- Personalizzazione grafica per ciascun cliente finale

2.4 Pianificazione

La pianificazione iniziale del progetto ha seguito lo schema del Modello Evolutivo, suddividendo il lavoro in macro-fasi:

Settimane 1-2 : Analisi preliminare e setup
- Studio del dominio applicativo (telefonia PABX)
- Analisi della struttura del database esistente
- Definizione dei requisiti fondamentali
- Setup dell'ambiente di sviluppo

Settimane 3-5 : Primo prototipo (MVP)
- Progettazione dell'architettura del sistema
- Implementazione delle query base per l'estrazione dati
- Sviluppo dell'interfaccia utente minimale
- Prime funzionalità di filtraggio e visualizzazione
- Prima dimostrazione al cliente finale

Settimane 6-7 : Raffinamento e ottimizzazione
- Ottimizzazione delle performance delle query
- Miglioramento dell'accuratezza dei dati
- Ampliamento delle funzionalità di filtraggio
- Implementazione dell'esportazione dati

- Implementazione eventuali feedback ricevuti

Settimana 8 : Personalizzazione e finalizzazione
- Sviluppo del sistema di personalizzazione grafica
- Testing completo con dati reali
- Documentazione finale
- Preparazione della presentazione finale al cliente

2.5 Analisi preventiva dei rischi

Durante la fase iniziale del progetto è stata condotta un'analisi dei rischi per identificare le potenziali criticità che avrebbero potuto compromettere il successo dello stage. Per ciascun rischio sono stati valutati la probabilità di occorrenza, l'impatto sul progetto e le strategie di mitigazione adottate.

R1 - Non rispetto delle tempistiche
- **Descrizione:** Impossibilità di completare le funzionalità pianificate entro le 320 ore di stage.
- **Probabilità:** Media
- **Impatto:** Alto
- **Cause potenziali:**
  - Sottostima della complessità tecnica
  - Difficoltà impreviste nell'integrazione con sistemi esistenti
  - Scarsa familiarità iniziale con il dominio applicativo
- **Strategie di mitigazione adottate:**
  - Adozione del Modello Evolutivo per rilasciare versioni funzionanti già dalle prime settimane
  - Pianificazione di sprint brevi (1 settimana) per identificare rapidamente eventuali ritardi
  - Definizione chiara delle priorità: focus sulle funzionalità core (MVP) prima delle funzionalità secondarie
  - Confronti settimanali con il tutor per ricalibrazione tempestiva degli obiettivi

R2 - Inaccuratezza dei dati mostrati
- **Descrizione:** Visualizzazione di statistiche e dati non corretti o fuorvianti per l'utente finale.
- **Probabilità:** Alta
- **Impatto:** Alto
- **Cause potenziali:**
  - Errata interpretazione della logica dei dati telefonici
  - Query SQL non corrette o incomplete

  - Mancata gestione di casi particolari nel dominio PABX
  - Problemi di aggregazione e calcolo delle statistiche
- **Strategie di mitigazione adottate:**
  - Sessioni approfondite con il tutor per comprendere la semantica dei dati PABX
  - Validazione incrociata dei risultati con report esistenti o conteggi manuali
  - Testing incrementale con dataset reali forniti dall'azienda
  - Revisione frequente delle query SQL con il tutor aziendale

R3 - Difficoltà nel recupero dei dati
- **Descrizione:** Complessità tecnica nell'estrazione efficiente dei dati dal database PABX.
- **Probabilità:** Media
- **Impatto:** Alto
- **Cause potenziali:**
  - Struttura del database complessa e non documentata
  - Performance scarse con query su grandi volumi di dati storici
  - Necessità di aggregazioni complesse
- **Strategie di mitigazione adottate:**
  - Fase iniziale dedicata esclusivamente allo studio del database
  - Creazione di query di test incrementali per validare la comprensione della struttura
  - Utilizzo di indici e ottimizzazioni progressive delle query

R4 - Scarsa esperienza con le tecnologie utilizzate
- **Descrizione:** Limitata familiarità con alcuni strumenti, linguaggi o framework necessari per il progetto.
- **Probabilità:** Bassa
- **Impatto:** Medio
- **Cause potenziali:**
  - Tecnologie non approfondite durante il percorso universitario
  - Specificità degli strumenti utilizzati in azienda
  - Curva di apprendimento necessaria per essere produttivi
- **Strategie di mitigazione adottate:**
  - Studio autonomo preliminare delle tecnologie principali
  - Utilizzo di documentazione ufficiale e tutorial
  - Refactoring progressivo del codice man mano che la padronanza aumentava

R5 - Requisiti poco chiari o in evoluzione
- **Descrizione:** Cambiamenti frequenti o ambiguità nei requisiti funzionali richiesti.

- **Probabilità:** Media
- **Impatto:** Medio
- **Cause potenziali:**
  - Necessità del cliente finale non completamente definite a priori
  - Feedback emergenti durante la visualizzazione dei prototipi
  - Nuove esigenze identificate durante lo sviluppo
- **Strategie di mitigazione adottate:**
  - Scelta del Modello Evolutivo proprio per gestire l'incertezza sui requisiti
  - Architettura modulare e flessibile per facilitare modifiche
  - Validazione frequente con il tutor e raccolta sistematica di feedback
  - Prioritizzazione dei requisiti: implementazione immediata dei requisiti certi, posticipazione di quelli incerti

R6 - Problemi di performance del sistema

- **Descrizione:** Tempi di risposta inaccettabili per l'utente finale, specialmente con grandi volumi di dati.
- **Probabilità:** Media
- **Impatto:** Basso
- **Cause potenziali:**
  - Query SQL non ottimizzate
  - Mancanza di indici appropriati
  - Caricamento di troppi dati contemporaneamente
  - Elaborazioni pesanti lato client
- **Strategie di mitigazione adottate:**
  - Test di performance fin dalle prime versioni con dataset realistici
  - Ottimizzazione progressiva delle query più critiche
  - Implementazione di paginazione e lazy loading
  - Monitoring dei tempi di esecuzione delle query principali

Capitolo 3

Analisi dei requisiti

In questo capitolo vengono analizzati e approfonditi i requisiti individuati per la realizzazione del progetto di dashboard per l'analisi delle chiamate telefoniche.

3.1 Introduzione ai requisiti

I requisiti del sistema sono stati identificati attraverso un processo di analisi condotto in collaborazione con il tutor aziendale e, successivamente, validati con il cliente finale durante la presentazione del primo prototipo funzionante. Questi sono stati suddivisi in due macro categorie principali:

Requisiti funzionali Rappresentano le funzionalità che il sistema deve offrire per rispondere alle esigenze degli utenti finali, definendo le operazioni che la dashboard deve essere in grado di svolgere. Si suddividono in:

- **Obbligatori**: indispensabili per il corretto funzionamento del sistema e per soddisfare le necessità primarie degli utenti
- **Desiderabili**: non strettamente necessari, tuttavia se implementati garantiscono una migliore esperienza utente e una maggiore usabilità del software
- **Opzionali**: la loro aggiunta non è essenziale per il funzionamento base del sistema, vengono implementati se rimane tempo a disposizione al termine dello sviluppo delle funzionalità prioritarie

Requisiti non funzionali Di questa categoria fanno parte i requisiti qualitativi e quelli di vincolo. I primi garantiscono una qualità maggiore del software dal punto di vista delle prestazioni, usabilità, affidabilità e sicurezza. I requisiti di vincolo invece stabiliscono limitazioni o condizioni che il sistema deve rispettare, come tecnologie da utilizzare, standard aziendali o normative da seguire.

3.2 Tracciamento dei requisiti

Per garantire una classificazione chiara e sistematica, i requisiti raccolti sono stati categorizzati in base alla loro tipologia e priorità, utilizzando la seguente notazione:

Sigla	**Significato**
F	Funzionale
N	Non funzionale
Q	Qualitativo
V	Vincolo
O	Obbligatorio
D	Desiderabile
P	Opzionale

Tabella 1: Legenda per la classificazione dei requisiti

Ogni requisito è stato identificato secondo il seguente schema di codifica:

R-XY-N

dove:

- **R** indica che si tratta di un requisito
- **X** indica se il requisito è funzionale (F) o non funzionale (N)
- **Y** indica il livello di importanza se il requisito è funzionale, oppure la tipologia se è non funzionale:
  - se X = F, allora Y può assumere i valori O (obbligatorio), D (desiderabile) o P (opzionale)
  - se X = N, allora Y può assumere i valori Q (qualitativo) o V (vincolo)
- **N** identifica in maniera univoca il requisito all'interno della sua macro categoria

3.3 Requisiti funzionali

Di seguito vengono elencati i requisiti funzionali identificati per il sistema di dashboard.

Codice	Descrizione
R-FO-1	Il sistema deve implementare un sistema di autenticazione sicuro tramite username e password
R-FO-2	Il sistema deve permettere l'accesso alle funzionalità solo ad utenti autenticati
R-FO-3	Il sistema deve visualizzare una pagina Dashboard principale contenente i dati generali delle chiamate con possibilità di filtraggio per periodo temporale
R-FO-4	Il sistema deve fornire una sezione Utenti contenente l'elenco completo degli interni telefonici e pagine di dettaglio per ciascun utente con statistiche e filtri per periodo
R-FO-5	Il sistema deve fornire una sezione Ring Group contenente l'elenco completo dei gruppi e pagine di dettaglio per ciascun ring group con statistiche e filtri per periodo
R-FO-6	Il sistema deve fornire una sezione DID contenente l'elenco completo dei Direct Inward Dialing e pagine di dettaglio per ciascun DID con statistiche e filtri per periodo
R-FO-7	Il sistema deve presentare i dati attraverso grafici per facilitarne l'interpretazione visuale
R-FO-8	Il sistema deve visualizzare metriche aggregate tramite card informative (chiamate totali, ricevute, risposte, perse, effettuate, tempo totale)
R-FO-9	Il sistema deve presentare l'elenco dettagliato delle chiamate in entrata e in uscita tramite tabelle

Codice	Descrizione
R-FO-10	Il sistema deve permettere l'esportazione dei dati delle tabelle in formato PDF e CSV
R-FD-1	Aggiunta di filtri nella Dashboard per utenti, ring group e did
R-FD-2	Creazione di una pagina dedicata alla gestione degli account, con inserimento, modifica ed eliminazione
R-FD-3	Personalizzazione del sistema con modifica loghi e nome cliente
R-FP-1	Implementazione della crittografia delle password memorizzate nel database
R-FP-2	Implementazione della modifica della password degli utenti (solo per admin)
R-FP-3	Creazione utente superadmin non modificabile ed eliminabile

Tabella 2: Tracciamento dei requisti funzionali.

3.4 Requisiti non funzionali

I requisiti non funzionali definiscono gli aspetti qualitativi e i vincoli tecnici del sistema.

Codice	Descrizione
R-NQ-1	Il sistema deve garantire tempi di risposta rapidi anche con volumi elevati di dati
R-NQ-2	L'interfaccia deve essere intuitiva senza necessità di formazione specifica
R-NQ-3	I messaggi di errore devono essere chiari e comprensibili
R-NQ-4	Il sistema deve avere una gestione corretta degli errori senza perdita di dati
R-NQ-5	L'accesso deve essere protetto contro accessi non autorizzati
R-NQ-6	Il sistema deve essere accessibile da dispositivi mobili e desktop
R-NV-1	Separazione tra frontend e backend tramite API RESTful
R-NV-2	Accessibilità tramite browser web moderni senza plugin aggiuntivi o installazioni locali
R-NV-3	Il sistema deve poter essere distribuito in ambienti containerizzati come Docker
R-NV-4	Possibilità di integrazione con sistemi di autenticazione esterni in futuro (LDAP, OAuth)

Tabella 3: Tracciamento dei requisti non funzionali.

3.5 Riepilogo dei requisiti

La tabella seguente riporta il riepilogo quantitativo dei requisiti identificati durante la fase di analisi.

Tipologia	**Quantità**
Requisiti funzionali	16
- Obbligatori	10
- Desiderabili	3
- Opzionali	3
Requisiti non funzionali	10
- Qualitativi	6
- Di vincolo	4
Totale	**26**

Tabella 4: Riepilogo dei requisiti di progetto

Capitolo 4

Tecnologie

In questo capitolo vengono presentate le tecnologie e gli strumenti utilizzati per lo sviluppo del progetto. Le scelte tecnologiche sono state effettuate sulla base dell'analisi dei requisiti del capitolo precedente, con particolare attenzione ai vincoli imposti dall'azienda ospitante e alla compatibilità con l'infrastruttura esistente.
Per ogni tecnologia viene fornita una breve descrizione e vengono spiegate le motivazioni che hanno portato alla sua adozione nel contesto specifico del progetto.

4.1 Linguaggi e Framework

4.1.1 HTML, CSS e JavaScript

Figura 1: Logo HTML5

Figura 2: Logo CSS3

Figura 3: Logo JavaScript

HTML (HyperText Markup Language), CSS (Cascading Style Sheets) e JavaScript sono i linguaggi fondamentali per lo sviluppo di applicazioni web. HTML fornisce la struttura semantica dei contenuti, CSS gestisce la presentazione visuale e il layout, mentre JavaScript implementa la logica interattiva e il comportamento dinamico dell'applicazione.

Motivazioni della scelta
La decisione di utilizzare le tecnologie web native, senza framework moderni come React Angular o Vue.js, è stata guidata da specifici vincoli aziendali e caratteristiche del progetto:

- **Vincolo aziendale:** L'azienda ospitante ha espresso la necessità di evitare l'adozione di framework complessi com curve di apprendimento ripide e dipendenze esterne. I dipendenti dell'azienda hanno una familiarità consolidata con HTML, CSS e JavaScript, rendendo più agevole la manutenzione e l'evoluzione del codice nel tempo.
- **Semplicità dell'interfaccia:** L'applicazione sviluppata presenta un'interfaccia utente relativamente semplice, che non richiede le funzionalità avanzate offerte dai framework moderni. L'uso diretto di HTML, CSS e JavaScript consente di mantenere il codice leggero e facilmente comprensibile.
- **Manutenibilità:** Per semplificare la manutenzione e lo sviluppo, sono state create classi JavaScript modulari e riutilizzabili per la generazione di elementi comuni come tabelle e grafici, garantendo coerenza nel codice senza la complessità aggiuntiva dei framework.
- **Prestazioni e scalabilità:** L'approccio nativo offre dimensioni ridotte dell'applicazione e tempi di caricamento più rapidi. Le tecnologie web standard mantengono inoltre una migliore compatibilità retroattiva con i browser, riducendo il rischio di obsolescenza del codice nel tempo.

4.1.2 Node.js

Figura 4: Logo Node.js

Node.js è un ambiente di esecuzione JavaScript lato server che consente di sviluppare applicazioni scalabili e ad alte prestazioni. Viene utilizzato per gestire il backend dell'applicazione, inclusa la logica di business, la gestione delle richieste HTTP e l'interazione con il database.

Motivazioni della scelta
La scelta di Node.js per lo sviluppo del backend è stata motivata da diverse considerazioni tecniche e stategiche:

- **Architettura API-first:** L'obiettivo del progetto era creare un backend basato su API RESTful che permettesse una netta separazione tra frontend e backend. Questa architettura facilita eventuali integrazioni future con altri sistemi software aziendali, consentendo di esporre le funzionalità del sistema telefonico attraverso endpoint ben definiti. Un'architettura basata su API rende inoltre il sistema più flessibile e manutenibile nel tempo.
- **Coerenza linguistica:** Mantenere JavaScript come linguaggio principale sia per il frontend che per il backend semplifica lo sviluppo e la manutenzione del codice. Gli sviluppatori possono lavorare su entrambe le parti dell'applicazione senza dover imparare linguaggi diversi, riducendo la curva di apprendimento e migliorando la produttività del team.
- **Ecosistema npm:** Node.js beneficia di npm, uno dei registri di pacchetti più grandi e attivi al mondo. Questo ecosistema offre una vasta gamma di librerie e strumenti che accelerano lo sviluppo, consentendo di integrare funzionalità complesse con facilità.
- **Prestazioni per operazioni I/O:** Node.js è particolarmente adatto per applicazioni che richiedono un'elevata gestione delle operazioni di input/output, come le API RESTful. La sua architettura basata su eventi e il modello non bloccante consentono di gestire un gran numero di connessioni simultanee in modo efficiente.

4.2 Database Management System

4.2.1 MySQL Server

Figura 5: Logo MySQL

MySQL è un database management system (DBMS) relazionale open source tra i più diffusi e utilizzati al mondo. Supporta il linguaggio SQL standart per la gestione e l'interrogazione dei dati, offrendo funzionalità avanzate come gestione delle transazioni ACID, meccanismi di backup e recovery, replicazione dei dati e ottimizzazione delle query. E' particolarmente adatto per applicazioni web grazie alla sua scalabilità, affidabilità e facilità di integrazione con vari linguaggi di programmazione, incluso JavaScript tramite Node.js.

Motivazioni della scelta
La decisione di adottare MySQL Server come DBMS per il progetto è stata guidata principalmente da ragioni di continuità tecnologica e compatibilità con l'infrastruttura esistente:

- **Coerenza con il sistema esistente:** Il centralino telefonico venduto da Cinquenet srl utilizza già MySQL come database per la gestione dei dati operativi (chiamate, utenti, configurazioni, ecc.). Mantenere la stessa tecnologia garantisce uniformità nell'infrastruttura IT aziendale e semplifica notevolmente la gestione complessiva dei sistemi.
- **Competenze interne:** Il personale tecnico dell'azienda possiede già familiarità con MySQL, riducendo la necessità di formazione aggiuntiva e facilitando la manutenzione e l'ottimizzazione del database nel tempo. Il team può gestire autonomamente backup, ottimizzazioni e troubleshooting senza necessità di acquisire nuove competenze su altri DBMS.
- **Leggerezza e prestazioni:** MySQL è noto per la sua efficienza e capacità di gestire carichi di lavoro elevati, rendendolo adatto per applicazioni web che richiedono accessi frequenti al database. La sua architettura ottimizzata consente di ottenere buone prestazioni anche con risorse hardware limitate.

4.2.2 MySQL Workbench

Figura 6: Logo MySQL Workbench

MySQL Workbench è lo strumento ufficiale di amministrazione e sviluppo per MySQL sviluppato da Oracle. Offre un'interfaccia grafica intuitiva per la gestione dei database, consentendo agli sviluppatori e agli amministratori di eseguire operazioni come la progettazione dello schema del database, la scrittura e l'esecuzione di query SQL, la gestione degli utenti e dei permessi, nonché il monitoraggio delle prestazioni del server MySQL.

Motivazioni della scelta
L'adozione di MySQL Workbench come strumento di gestione del database è stata motivata da diversi fattori chiave:

- **Interfaccia grafica intuitiva:** Workbench permette di gestire il database attraverso un'interfaccia visuale user-friendly, semplificando operazioni complesse come la progettazione dello schema ER (Entity-Relationship), la creazione e modifica di tabelle, l'esecuzione di query e la visualizzazione dei risultati. Questo risulta particolarmente utile durante lo sviluppo per verificare rapidamente la struttura dei dati e testare query.
- **Strumento ufficiale:** Essendo lo strumento ufficiale sviluppato da Oracle, MySQL Workbench garantisce piena compatibilità con tutte le funzionalità di MySQL. Questo assicura che tutte le operazioni eseguite tramite Workbench siano supportate e ottimizzate per il server MySQL.

4.3 Strumenti di Sviluppo

4.3.1 Postman

Figura 7: Logo Postman

Postman è una piattaforma completa per lo sviluppo e testing di API che consente di progettare, testare, documentare e monitorare interfacce REST atraverso un'interfaccia intuitiva. E' diventato lo standard de facto per il testing di API grazie alla sua semplicità d'uso e alle sue funzionalità avanzate.

Motivazioni della scelta
Postman è stato scelto come strumento principale per il testing delle API sviluppate nel progetto per diverse ragioni:

- **Testing efficiente delle API:** Durante lo sviluppo del backend basato su API REST, era fondamentale poter testare rapidamente gli endpoint senza dover sviluppare prima il frontend. Postman permette di inviare richieste HTTP (GET, POST, PUT, DELETE) con parametri personalizzati, headers e body in formato JSON, visualizzando immediatamente le risposte del server. Questo ha accelerato significativamente il ciclo di sviluppo e debug.
- **Gestione delle collections:** Postman consente di organizzare le richieste API in collezioni, facilitando la gestione e il riutilizzo dei test. Durante lo sviluppo, sono state create collezioni specifiche per ogni risorsa API, permettendo di eseguire test ripetitivi in modo strutturato.
- **Debugging efficiente:** La visualizzazione dettagliata delle risposte, inclusi status code, headers, body e tempi di risposta, facilita l'identificazione rapida di problemi e l'ottimizzazione delle performance delle API.

4.3.2 Suite JetBrains: IntelliJ IDEA e WebStorm

Figura 8: Logo JetBrains

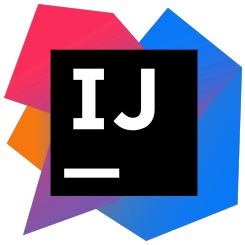


Figura 9: Logo IntelliJ IDEA

Figura 10: Logo WebStorm

JetBrains offre una suite di IDE (Integrated Development Enviroment) professionali specifici per linguaggi e tecnologie. IntelliJ IDEA è ottimizzato per lo sviluppo Java, ma supporta anche JavaScript e Node.js tramite plugin. WebStorm è un IDE specializzato per lo sviluppo web front-end e back-end con supporto nativo per HTML, CSS e JavaScript.

Motivazioni della scelta
L'adozione degli IDE JetBrains per lo sviluppo del progetto è stata motivata da diversi vantaggi chiave:

- **Intelligent code completion:** Gli IDE JetBrains offrono funzionalità avanzate di completamento del codice basate su analisi statica, che accelerano la scrittura del codice riducendo gli errori di sintassi e migliorando la produttività degli sviluppatori.
- **Refactoring avanzato:** Le potenti funzionalità di refactoring consentono di ristrutturare il codice in modo sicuro e efficiente, facilitando la manutenzione e l'evoluzione del progetto nel tempo.
- **Integrazione con strumenti:** Integrazione nativa con Git per il version control, npm per la gestione dei pacchetti, terminale integrato e supporto per l'esecuzione diretta di script Node.js. Questo centralizza il workflow di sviluppo in un'unica applicazione, evitando il continuo cambio di contesto tra diversi strumenti.
- **Specializzazione per contesto:** Utilizzare IntelliJ IDEA per il backend Node.js e WebStorm per il frontend web garantisce strumenti ottimizzati per ciascun stack tecnologico, con funzionalità, suggerimenti e plugin specifici per il tipo di sviluppo in corso.
- **Licenza accademica gratuita:** Come studente universitario, è possibile ottenere gratuitamente licenze educational per tutti i prodotti JetBrains, rendendo accessibile questa suite professionale senza costi aggiuntivi. Questo ha permesso di utilizzare strumenti di qualità enterprise durante lo sviluppo del progetto.

4.4 Versionamento del Codice

4.4.1 Git e GitHub

Figura 11: Logo Git

Figura 12: Logo GitHub

Git è un sistema di controllo versione distribuito che traccia le modifiche al codice sorgente durante lo sviluppo software. GitHub è una piattaforma di hosting per repository Git basata su cloud che aggiunge funzionalità collaborative, gestione progetti e strumenti di integrazione continua.

Motivazioni della scelta
Git e GitHub sono stati scelti come strumenti di versionamento del codice per diverse ragioni fondamentali:

- **Standard de facto:** Git è lo standard industriale per il version control, adottato dalla maggioranza dei progetti software moderni. La sua conoscenza è fondamentale per qualsiasi sviluppatore e la sua adozione garantisce compatibilità con praticamente qualsiasi workflow aziendale.
- **Tracciamento completo delle modifiche:** Ogni commit mantiene uno snapshot completo del progetto con metadata dettagliati (autore, data, messaggio descrittivo). Questo permette di ripercorrere l'intera evoluzione del software, comprendere le motivazioni dietro ogni modifica e identificare quando e dove sono stati introdotti eventuali bug.
- **Branching e sviluppo parallelo:** Il modello di branching di Git permette di lavorare su nuove funzionalità o correzioni in branch isolati senza interferire con il codice principale (branch main/master). Questo consente di sperimentare in sicurezza e di mantenere sempre una versione stabile del codice pronta per il deployment.
- **Backup distribuito:** La natura distribuita di Git garantisce che ogni clone del repository sia un backup completo della storia del progetto. Questo previene perdite di dati e permette di continuare a lavorare anche offline, sincronizzando le modifiche successivamente.

4.5 Documentazione

4.5.1 Typst

Figura 13: Logo Typst

Typst è un sistema di typesetting moderno, progettato come alternativa contemporanea a LaTeX. Utilizza una sintassi più intuitiva e leggera, tempi di compilazione significativamente più rapidi e un'architettura pensata per semplificare la creazione di documenti tecnici di alta qualità tipografica.

Motivazioni della scelta
Typst è stato scelto come strumento di documentazione per il progetto per diverse ragioni chiave:

- **Sintassi intuitiva:** La sintassi di Typst è progettata per essere più leggibile e facile da imparare rispetto a LaTeX. Questo ha permesso di concentrarsi maggiormente sul contenuto del documento piuttosto che sulla complessità della formattazione, accelerando il processo di scrittura.
- **Compilazione rapida:** Typst offre tempi di compilazione molto più veloci rispetto a LaTeX, consentendo di vedere rapidamente le modifiche apportate al documento. Questo consente un workflow iterativo più fluido con preview istantaneo delle modifiche, facilitando la correzione di errori di formattazione e l'aggiustamento del layout in tempo reale.
- **Facilità di apprendimento:** Per chi non ha esperienza pregressa con LaTeX, Typst risulta molto più accessibile e immediato da utilizzare. La documentazione è chiara e moderna, con esempi pratici che permettono di iniziare a produrre documenti professionali rapidamente.

4.6 Containerizzazione

4.6.1 Docker

Figura 14: Logo Docker

Docker è una piattaforma di containerizzazione che permette di pacchettizzare applicazioni con tutte le loro dipendenze in container isolati e portabili. I container sono ambienti di esecuzione leggeri e autosufficienti che garantiscono che l'applicazione funzioni allo stesso modo su qualsiasi sistema che supporti Docker.

Motivazioni della scelta
L'adozione di Docker per il progetto è stata motivata da diversi vantaggi significativi offerti dalla containerizzazione:

- **Ambiente consistente e riproducibile:** Docker elimina il classico problema del «funziona sulla mia macchina» garantendo che l'applicazione giri esattamente allo stesso modo in sviluppo, test e produzione. Questo riduce drasticamente i problemi legati a differenze di configurazione tra ambienti diversi.
- **Isolamento delle dipendenze:** Ogni componente dell'applicazione (backend Node.js, database MySQL, eventuali servizi aggiuntivi) può essere containerizzato separatamente con le proprie dipendenze specifiche, evitando conflitti tra versioni di librerie e semplificando la gestione complessiva del sistema.
- **Deployment semplificato:** Un'immagine Docker contiene tutto il necessario per eseguire l'applicazione: codice, runtime, librerie di sistema e configurazioni. Questo rende il processo di deployment consistente, ripetibile e molto più semplice rispetto all'installazione manuale di tutte le dipendenze su ogni macchina.
- **Facilitazione dello sviluppo:** Docker Compose permette di orchestrare facilmente servizi multipli (backend + database) con una semplice configurazione YAML, semplificando il setup dell'ambiente di sviluppo locale. Nuovi sviluppatori possono avviare l'intero stack con un singolo comando.

Glossario

Bibliografia