UNIVERSITÀ POLITECNICA DELLE MARCHE
FACOLTÀ DI INGEGNERIA
Dipartimento di Ingegneria dell'Informazione
Corso di Laurea in Ingegneria Informatica e dell'Automazione

TESI DI LAUREA

Progettazione e implementazione di un sistema informativo a supporto della gestione differenziata dei rifiuti

Design and implementation of an information system to support differentiated waste management

Relatore

Prof. Domenico Ursino

Candidato

Edoardo Cecchini

ANNO ACCADEMICO 2023-2024

Si muore tutte le sere, si rinasce tutte le mattine: è così. E tra le due cose c'è il mondo dei sogni.

Henri Cartier-Bresson

Sommario

Negli ultimi anni, la questione ambientale ha assunto un ruolo centrale nel dibattito globale. Un tassello fondamentale nella lotta al cambiamento climatico, che ha raggiunto livelli allarmanti, è la gestione efficace della raccolta differenziata dei rifiuti. Pertanto, risulta necessario un cambio di paradigma, che porti a vedere i rifiuti prodotti non più come oggetti da smaltire, ma come risorse preziose. Questo approccio, noto come "economia circolare", rappresenta l'unico modello sostenibile per mantenere lo stile di vita attuale. In questa tesi, si è effettuata un'analisi dettagliata delle operazioni di un'azienda locale specializzata nei servizi di gestione dei rifiuti, terminando con lo sviluppo di un sistema informatico moderno per supportarne le fasi amministrative e operative. In particolare, nell'intero processo sono state impiegate le metodologie suggerite dall'ingegneria del software, includendo l'analisi dei requisiti, la progettazione della componente dati, la progettazione della componente applicativa e, infine, l'implementazione effettiva del software.

Keyword: Gestione dei Rifiuti, Sostenibilità, Economia Circolare, Giri di Raccolta, Green, Riciclo, Futuro, Web App, Ingegneria del software, Kaizen.

Indice

Elenco delle figure

Elenco delle tabelle

Introduzione

Negli ultimi decenni, la questione ambientale è diventata uno dei temi più rilevanti e urgenti a livello globale. Una delle sfide principali è l'adozione di un meccanismo di gestione dei rifiuti che sia il più sostenibile ed efficiente possibile. Il volume dei rifiuti prodotti a livello mondiale nel 2023 è stimato in circa 2,3 miliardi di tonnellate. Per dare un'idea della portata di questo problema, l'Empire State Building, uno dei grattacieli più iconici e alti del mondo, pesa circa 365.000 tonnellate. Ciò significa che produciamo, in rifiuti, circa 6.300 Empire State Buildings ogni anno. Senza interventi urgenti e mirati, questo dato è destinato a crescere esponenzialmente nei prossimi anni, aggravando ulteriormente l'impatto ambientale e la pressione sulle risorse naturali. È, quindi, chiaro che ora, più che mai, è essenziale una corretta gestione dei rifiuti per mitigare questi effetti negativi.

Due strumenti molto potenti a nostra disposizione sono la raccolta differenziata e il riciclaggio, i quali consentono di ridurre il volume di rifiuti destinati alle discariche e agli inceneritori, minimizzando, così, l'inquinamento del suolo, dell'acqua e dell'aria. Il riciclaggio consente, inoltre, di recuperare materiali preziosi, riducendo la necessità di estrarre nuove risorse naturali e abbassando le emissioni di gas serra associate alla produzione di nuovi materiali. Considerando il fabbisogno attuale di risorse, l'unico modo per evitare danni irreversibili è quello di creare economie circolari, dove i materiali vengono riutilizzati e riciclati in un ciclo continuo, promuovendo, così, la sostenibilità a lungo termine. Questo approccio non solo riduce l'impatto ambientale, ma crea anche opportunità economiche e posti di lavoro nel settore del riciclaggio e della gestione dei rifiuti.

In Italia, la raccolta differenziata ha fatto notevoli progressi negli ultimi anni; tuttavia, resta ancora molto da fare, soprattutto per quanto riguarda l'informatizzazione e l'ottimizzazione dei processi. Molte realtà locali operano ancora con sistemi gestionali obsoleti o manuali, che limitano significativamente l'efficienza complessiva delle operazioni. L'adozione di soluzioni informatiche avanzate può portare numerosi benefici, tra cui una gestione più efficace delle risorse umane e logistiche, l'automatizzazione di vari processi ripetitivi e una maggiore chiarezza da parte delle aziende sulle esigenze territoriali. In un mondo in costante cambiamento, in cui ci troviamo a fronteggiare sfide sempre più impegnative, l'unico modo per rimanere competitivi e fornire servizi di qualità, è quello di *adattarsi*.

Le motivazioni che hanno guidato la realizzazione di questo progetto, dunque, sono quelle di sviluppare un sistema in grado di supportare in modo ottimizzato e centrato sull'esperienza utente le attività di un'organizzazione operante in un contesto complesso, come quello precedentemente delineato.

In questo elaborato viene delineato il processo di sviluppo di un'applicazione web dedicata al supporto della gestione della raccolta dei rifiuti presso l'azienda Marche Multiservizi.

Inizialmente, è stata condotta un'analisi dettagliata del contesto operativo dell'azienda, delimitando le aree di maggior interesse legate al nostro progetto. Successivamente, sono stati definiti i requisiti funzionali che l'applicazione deve soddisfare e i criteri che hanno portato alla realizzazione della componente dati.

Una volta completata questa fase, è stata eseguita la progettazione della componente applicativa. Questa fase ha compreso la definizione dell'architettura del sistema e la selezione delle tecnologie più appropriate, nonché la creazione di mockup finalizzati a delineare l'interfaccia grafica dell'applicativo.

Terminata la fase di progettazione, è stata illustrata l'implementazione pratica, includendo esempi di codice per evidenziare come sono state realizzate funzionalità rilevanti all'interno dell'applicazione. Infine, è stato redatto un manuale utente dettagliato con l'obiettivo di facilitare l'adozione del nuovo sistema ed è stato realizzato un confronto con i sistemi di altre località per trarne spunti interessanti.

La presente tesi è composta da sette capitoli strutturati come di seguito specificato:

- Nel Capitolo 1 viene fornita una panoramica del contesto generale di riferimento, analizzando in dettaglio i processi interni di un'azienda di gestione dei rifiuti.
- Nel Capitolo 2 si definiscono i requisiti funzionali e non funzionali che l'applicazione deve soddisfare, secondo le necessità dell'azienda.
- Nel Capitolo 3 ci si concentra sull'individuazione dei dati rilevanti da raccogliere e sulla progettazione della struttura dati necessaria per memorizzarli e gestirli.
- Nel Capitolo 4 viene progettata la componente applicativa, descrivendo l'architettura del sistema e le tecnologie necessarie per la sua realizzazione.
- Nel Capitolo 5 si descrive il processo di realizzazione pratica del sistema informatico, con dettagli sull'implementazione di alcune funzionalità attraverso porzioni di codice.
- Nel Capitolo 6 viene mostrato il manuale utente, fornendo una guida dettagliata per gli utenti finali del sistema su come navigare tra le diverse pagine e utilizzare le funzionalità disponibili.
- Nel Capitolo 7 è presente un confronto critico tra il sistema sviluppato e altri sistemi di gestione dei rifiuti esistenti, analizzando punti di forza, debolezze e potenziali miglioramenti.

CAPITOLO 1

Descrizione del contesto di riferimento

In questo capitolo introduttivo, l'obiettivo è ottenere una visione chiara del mondo che circonda l'argomento trattato per comprendere appieno gli elementi con cui è stato necessario interfacciarsi e le motivazioni dietro la realizzazione del sistema informativo. Si partirà con un'analisi del funzionamento generale del contesto di riferimento, ovvero la gestione dei rifiuti, continuando con le modalità attraverso le quali l'azienda presa in esame percepisce e gestisce questi elementi. Verranno, poi, esaminati i processi interni e il funzionamento dell'azienda per delineare con chiarezza le dinamiche coinvolte.

Una volta acquisita una visione completa del contesto e dei processi aziendali, si procederà a delimitare un'area di interesse specifica, concentrandosi sugli aspetti a noi più rilevanti e gestibili del problema. Tutto ciò ci fornirà le informazioni necessarie per dare il via alla vera e propria parte di progettazione del sistema informativo.

1.1 Descrizione del contesto di riferimento

Il contesto analizzato riguarda la gestione dei rifiuti urbani nella regione Marche, un'area che si estende dalle coste del Mar Adriatico alle colline dell'entroterra.

Questo territorio, ricco di diversità paesaggistiche e ambientali, combina aree urbane densamente popolate con zone rurali e costiere, ognuna con esigenze specifiche per quanto riguarda la gestione dei rifiuti.

1.1.1 La raccolta di prossimità

Esistono vari approcci per portare a termine questo compito, che possono variare in base alle esigenze.

Il focus del progetto è, però, posto sulla *raccolta di prossimità*, un sistema che prevede il dislocamento strategico dei contenitori per lo smaltimento di rifiuti su tutto il territorio urbano, con l'obiettivo di facilitare la raccolta differenziata da parte dei cittadini.

Questo sistema include contenitori specifici e ben caratterizati per le diverse tipologie di rifiuti, ovvero:

- carta;
- plastica;
- vetro;
- organico;

- indifferenziata.

Parte fondamentale di questo meccanismo sono i *giri di raccolta*, durante i quali vengono scaricati i bidoni e i materiali raccolti vengono portati ai differenti punti di scarico. Questi giri sono particolarmente complessi da organizzare, vista la grande quantità di variabili in gioco.

1.1.2 Sfide logistiche nella raccolta dei rifiuti

Come già accennato, in un territorio così particolare, l'organizzazione dei giri di raccolta è inevitabilmente influenzata da diversi fattori. Tra questi citiamo:

- Demografia: la densità abitativa varia notevolmente tra le diverse zone, influenzando notevolmente la frequenza e le modalità di svuotamento dei bidoni.
- Topografia: la regione presenta una topografia diversificata, che spazia dalle zone coostiere a zone più montane, con aree più complesse da raggiungere che richiedono soluzioni e mezzi differenti.

Da ciò deriva l'importanza di adottare un sistema che sia il più flessibile e ottimizzato possibile, per poter fronteggiare al meglio queste esigenze senza spreco di risorse preziose.

1.1.3 Situazione attuale e opportunità

In termini di raccolta differenziata, la regione Marche ha raggiunto una percentuale del 71,6%, superando l'obiettivo del 65% fissato dalla normativa per il 2012. Questo risultato posiziona le Marche tra le regioni più virtuose in Italia.

Nonostante questi successi, è fondamentale continuare a ricercare una costante ottimizzazione dei processi. Esiste una grande opportunità di miglioramento nell'adozione di piattaforme digitali moderne che lavorino in sinergia con sistemi di monitoraggio avanzati (IoT) e che siano in grado di immagazzinare e sfruttare al meglio i dati.

Partendo dai singoli enti, si potrebbe poi puntare all'integrazione di un sistema comune nazionale, per affrontare una delle più insinuose problematiche presenti attualmente, ovvero la frammentazione degli enti di gestione dei rifiuti. In molte parti d'Italia, la gestione è suddivisa tra numerosi consorzi e aziende locali. Questa frammentazione può portare a inefficienze operative e a una disomogeneità nei livelli di servizio offerti ai cittadini.

1.2 Introduzione all'azienda

Nell'ottica di apportare un contributo che sia di valore ad un organizzazione così complessa, risulta essenziale conoscerne a fondo il funzionamento, le entità coinvolte e le relazioni che intercorrono tra di esse.

1.2.1 Settore di operazione

Il primo passo in questa analisi consiste nell'esaminare le diverse entità che costituiscono il settore in cui opera l'azienda oggetto di studio. Questa analisi approfondita permetterà di acquisire una comprensione più dettagliata e articolata delle dinamiche e delle sfide con cui l'azienda si confronta quotidianamente.

Raffigurate in Tabella 1.1 troviamo queste entità, seguite da una breve descrizione per ognuna di esse.

Le entità riportate rappresentano una semplificazione necessaria della realtà studiata, volta a focalizzare l'attenzione sugli elementi di maggiore rilevanza per la nostra analisi.

Nome	Descrizione	Sinonimi
bidone	recipiente per la raccolta di rifiuti di vario genere	cassonetto
operatore	personale addetto allo scarico di rifiuti	dipendente
giro di raccolta	percorso prestabilito dettato dall'ordine di bidoni che vanno scaricati	itinerario
mezzo	veicolo specifico utilizzato durante i giri di raccolta per scaricare i bidoni	camion
turno di lavoro	periodo di attività di un operatore	-
anomalia	irregolarità durante lo svolgimento di un giro di raccolta	-

Tabella 1.1: Glossario delle entità coinvolte

Questo ci permette di concentrare le energie sugli aspetti a noi cruciali del settore, evitando di disperderci tra le innumerevoli casistiche e sezioni che inevitabilmente caratterizzano un ambito così complesso.

In particolare, ci siamo concentrati sulle entità rilevanti per i lavoratori del settore, trascurando volutamente la parte relativa all'interfacciamento con i clienti, che, seppur importante, esula dagli obiettivi di creazione di un sistema gestionale interno e non aperto al pubblico.

1.2.2 Implementazione delle entità nel contesto aziendale

Una volta appresi gli elementi principali con cui sarà necessario interfacciarsi nel settore di riferimento, ogni ente implementa le metodologie che ritiene più opportune.

Vediamo, quindi, nel dettaglio come queste entità (riportate in Tabella 1.1) prendono vita nel nostro contesto, con particolare attenzione posta ai dati da raccogliere, fondamentali per gli step successivi:

- Bidoni: tutti i bidoni sono identificati da un tag (codice) univoco, che ne permette la corretta identificazione. Essi presentano dei colori ben riconoscibili, associati alla tipologia di rifiuto che devono contenere. Risulterà molto importarte, poi, per la creazione di giri di raccolta, tenere traccia della loro collocazione (indirizzo in cui si trovano).

- Operatori: esistono molti dipendenti che contribuiscono alle attività dell'azienda; nel nostro caso specifico, però, faremo sempre riferimento a coloro che montano sui mezzi appositi ed effettuano i giri di raccolta. Qui i dati importanti sono quelli anagrafici; vedremo, poi, che verrà fornito loro un usernmane univoco per accedere all'applicativo web.

- Giri di raccolta: i giri di raccolta rappresentano dei percorsi standard che dovranno essere effettuati dagli operatori. Sono definiti da una lista ordinata di bidoni, con stessa tipologia di rifiuto, da svuotare. Essi non variano significativemente nel tempo, ma c'è un costante lavoro di modifica e ottimizzazione, basato sul rendimento degli itinerari già effettuati. Con l'obiettivo di fornire un servizio completo, i giri di raccolta sono svariati e si differenziano per fascia oraria, frequenza settimanale e giorni in cui devono essere svolti.

- Mezzi: i mezzi sono i camion utilizzati dagli operatori per eseguire correttamente i giri di raccolta; essi presentano allestimenti diversi in base alle varie esigenze. Da notare che alcuni mezzi più complessi, richiedono all'operatore una patente di tipo C.

- Turni di lavoro: i turni di lavoro sono gli orari prestabiliti in cui ogni operatore lavora; è necessaria quindi una combinazione minuziosa con i giri di raccolta da eseguire, per evitare lo spreco di risorse.

- Anomalie: le anomalie sono degli eventi inattesi che vengono segnalati dagli operatori una volta terminato il giro di raccolta. Questo meccanismo è importante in quanto genera dei feedback per eventuali miglioramenti.

1.3 Analisi dei processi interni

In Figura 1.1 è riportato un diagramma realizzato al fine di avere una rappresentazione visiva e di insieme del flusso di azioni e delle relazioni che intercorrono tra le "parti" dell'azienda presa in esame. L'utilizzo di colori diversi è stato adottato per differenziare gli attori principali delle varie azioni; in particolare:

- l'azzurro evidenzia le azioni svolte dall'operatore;
- il giallo denota le azioni di competenza del settore. amministrativo;

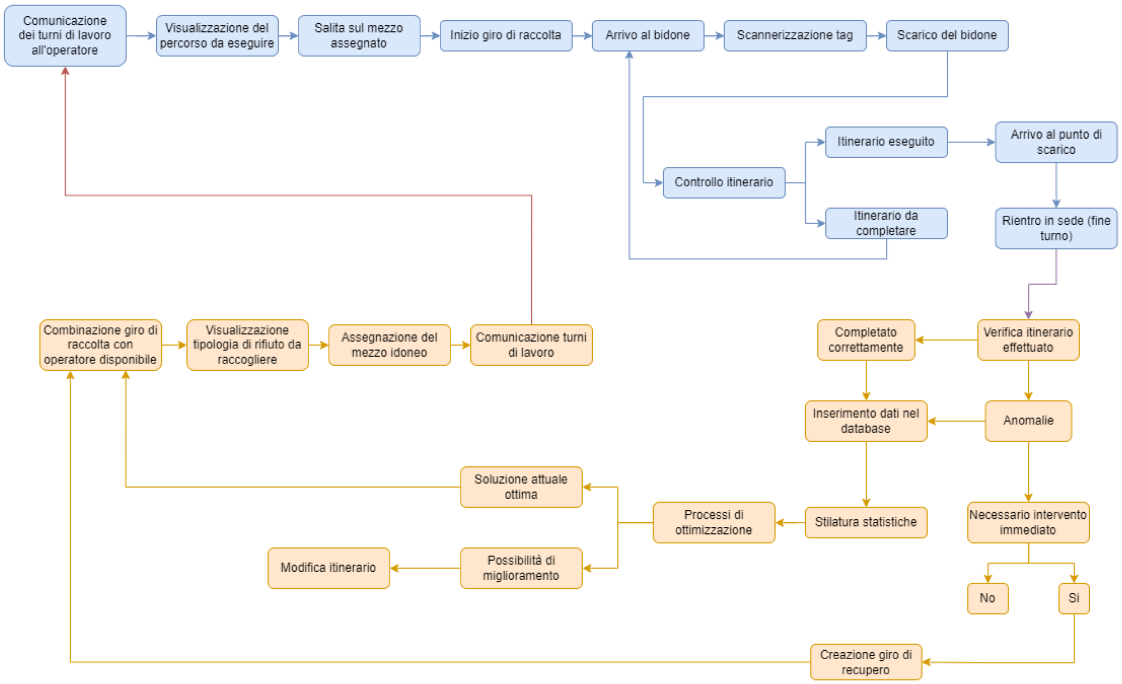

Figura 1.1: Diagramma del flusso complessivo dei processi interni

Ora che abbiamo un supporto visivo sui passaggi chiave svolti dalle varie entità, forniamo una spiegazione più dettagliata delle principali fasi e interazioni tra le diverse componenti.

1.3.1 Descrizione del flusso di azioni dell'operatore

Ad ogni inizio della giornata lavorativa, gli operatori troveranno sul proprio dispositivo il turno di lavoro assegnato, comprensivo del giro di raccolta da eseguire. Essi possono visualizzarne tutti i dettagli, con particolare attenzione al percorso da seguire e al mezzo apposito da utilizzare. Inizia, così, il vero e proprio giro di raccolta; raggiunto il primo bidone della lista, l'operatore scannerizza il tag univoco del cassonetto e procede con loscarico.

Questo procedimento di arrivo al bidone, scannerizzazione e scarico si ripete per ogni cassonetto fino al completamento del giro di raccolta.

Una volta terminato il giro, ovvero terminati i bidoni da scaricare, l'operatore raggiunge il punto di scarico stabilito e poi rientra in sede; questo sarà il momento che sancisce la fine effettiva del suo turno di lavoro.

Al termine della raccolta, l'operatore ha la possibilità di segnalare eventuali anomalie riscontrate durante l'itinerario appena effettuato. Lo storico di queste informazioni, come accennato precedentemente, è importante per i successivi processi di ottimizzazione interna.

1.3.2 Descrizione del flusso di azioni amministrativo

Analizziamo, ora, le azioni svolte dalla parte amministrativa; esse si concentrano su due tipologie di attività differenti, ma entrambe legate ai giri di raccolta:

- pianificazione;
- ottimizzazione.

La pianificazione delle risorse disponibili consiste in un'efficace combinazione continua tra i giri di raccolta da eseguire, gli operatori che possono svolgerli e i mezzi disponibili.

Nel dettaglio, una volta assegnato un giro di raccolta al turno di lavoro di un operatore, si tiene conto della tipologia di rifiuto che quel giro ha l'obiettivo di raccogliere, e, di conseguenza, viene assegnato all'operatore il mezzo idoneo per svolgere tale compito.

Questo processo viene ripetuto ciclicamente per tutti i giri di raccolta da eseguire, assicurando un'allocazione ottimale delle risorse e una gestione efficiente dei turni di lavoro.

L'ottimizzazione dei processi di raccolta è un altro elemento importante, che non può essere raggiunto senza una fase preliminare di prove, test e analisi approfondite sulla grande quantità di dati fornita dagli itinerari effettuati. Questo è il motivo per cui, nel diagramma in Figura 1.1, l'inizio del processo di analisi dei dati è strettamente collegato al termine del giro di raccolta svolto dagli operatori.

Al termine di un giro di raccolta, come detto in precedenza, l'operatore ha la possibilità di segnalare l'andamento dell'itinerario appena concluso. Ogni itinerario, sia che si svolga come pianificato sia che presenti anomalie, viene registrato nel database dell'azienda per creare uno storico dettagliato.

Nel caso in cui un giro di raccolta presenti un'anomalia, viene analizzata la natura del problema e, se necessario, vengono pianificati interventi immediati. Ad esempio, se un giro non viene completato e rimangono bidoni da scaricare, viene creato un giro di raccolta speciale di recupero, assegnato a un operatore disponibile oppure a un operatore a cui è assegnato un giro con priorità minore.

L'enorme quantità di dati derivanti da tutti gli itinerari effettuati viene utilizzata per stilare delle statistiche che evidenziano le criticità nei giri di raccolta. Ad esempio, potrebbe emergere che un determinato giro non viene mai completato entro il tempo stabilito o che uno specifico bidone non viene raccolto con regolarità. Grazie a questa analisi dei dati, possono essere implementati processi di ottimizzazione, modificando i giri di raccolta in base alle informazioni raccolte.

Questo processo è ciclico e ha l'obiettivo di garantire il miglior servizio possibile, riducendo al minimo gli sprechi e migliorando continuamente l'efficienza operativa.

1.4 Delimitazione dell'area di interesse

Le regioni precedenti ci hanno fornito una chiara comprensione del contesto operativo dell'azienda e del funzionamento delle sue principali componenti interne. Tuttavia, data la natura "didattica" del progetto di tirocinio, è essenziale definire chiaramente i limiti dei compiti da svolgere per garantire un output qualitativo e non dispersivo.

Il dominio sul quale si basa la costruzione del sistema informativo è rappresentato in Figura 1.2:

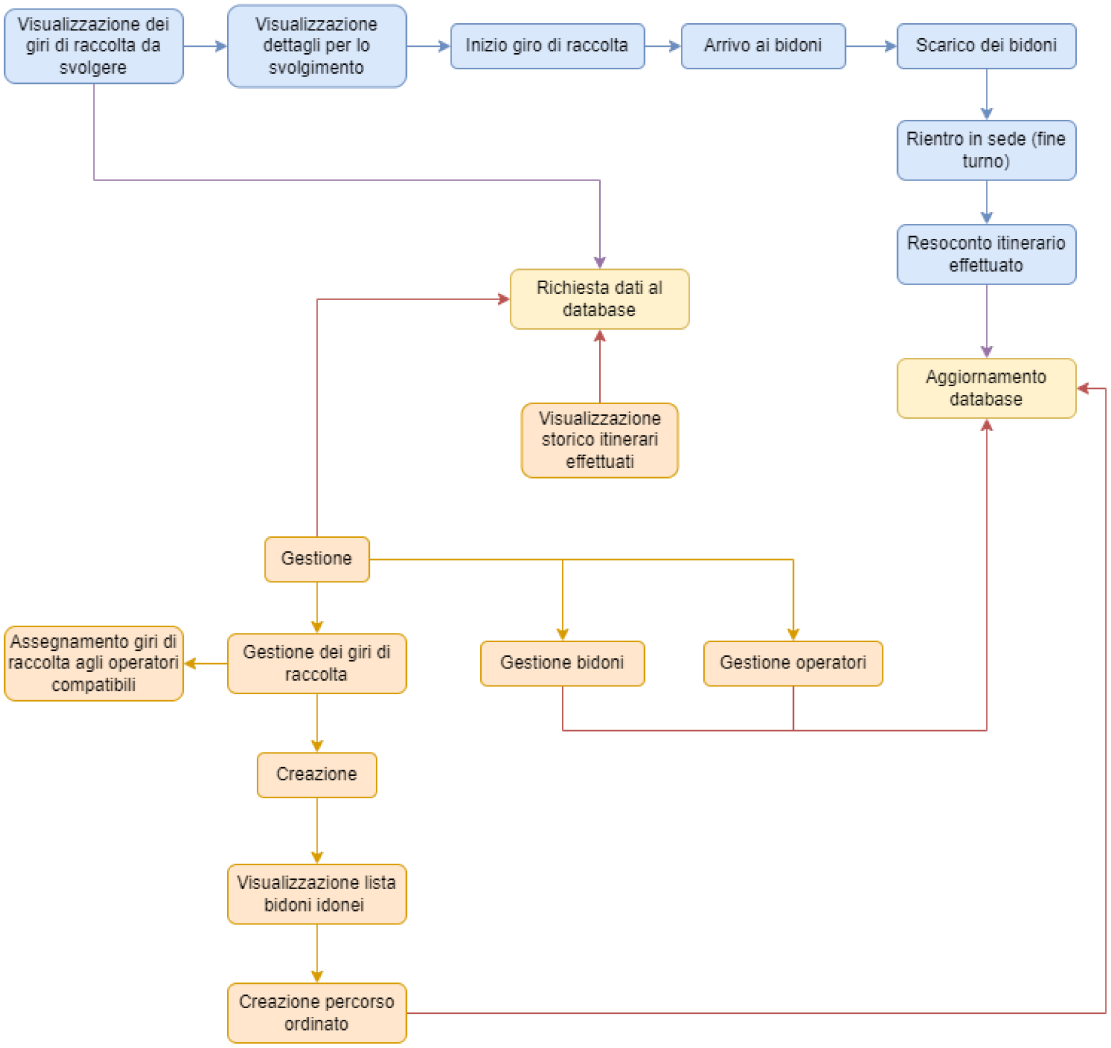

Figura 1.2: Diagramma di flusso finale

Osservando il diagramma relativo alla Figura 1.2, emerge chiaramente che il sistema informativo che verrà realizzato è un gestionale, pensato più per supportare le operazioni di pianificazione (amministrativa) ed esecuzione (operatori) dei giri di raccolta descritti in precedenza. La missione principale del sistema che verrà realizzato resta quella di rendere semplici e dinamiche queste operazioni.

1.4.1 Differenze

La prima differenza rispetto alla realtà operativa consiste nella rimozione dei turni di lavoro. Nell'analisi proposta, questi saranno integrati nell'assegnazione dei giri di raccolta agli operatori. In pratica, la durata dei giri di raccolta assegnati a un operatore determinerà implicitamente il suo turno di lavoro.

Per semplificare ulteriormente questo processo, mantenendo un livello di accuratezza adeguato, si è optato per la standardizzazione dei giri di raccolta a una durata di 6 ore ciascuno. Conseguentemente, i percorsi di raccolta dei bidoni saranno progettati per adattarsi a questa durata.

Sebbene si sia deciso di tenere conto degli itinerari effettuati e quindi di gestire le anomalie e i possibili giri di raccolta di recupero, l'analisi delle statistiche e l'ottimizzazione dei processi

non verranno trattate. Questi aspetti avrebbero richiesto uno studio matematico complesso per essere implementati efficacemente.

Un'ulteriore differenza, dovuta a limitazioni di risorse, è la rimozione della gestione dettagliata dei mezzi. Implementare correttamente questa funzionalità avrebbe richiesto uno studio approfondito sulle diverse tipologie di allestimenti e sulle normative in vigore.

Con queste premesse, possiamo ora procedere alla fase di progettazione, in cui tradurremo tutte queste informazioni in requisiti funzionali per realizzare il sistema informativo.

CAPITOLO 2

Specifica dei requisiti

In questo capitolo, l'obiettivo è quello di definire chiaramente ciò che desideriamo che la nostra applicazione consenta di fare, in modo da rispondere alle necessità dell'azienda. Partendo dall'analisi dettagliata delle entità coinvolte, svolta nel capitolo precedente, ci concentreremo sui requisiti funzionali e non funzionali che guideranno la progettazione e lo sviluppo del sistema informativo.

Esamineremo, prima di tutto, gli obiettivi di natura amministrativa, che mirano a garantire una gestione efficace e pianificata dei giri di raccolta. Successivamente, affronteremo gli obiettivi di natura operativa, volti a facilitare il lavoro quotidiano degli operatori sul campo. Per ciascuna di queste due sezioni, saranno descritti i requisiti funzionali e non funzionali che il sistema deve soddisfare.

Infine, per i requisiti più complessi, verranno definiti i casi d'uso, illustrando le interazioni degli utenti necessarie per ottenere i risultati desiderati.

2.1 Obiettivi in linguaggio naturale

Nel capitolo precedente, abbiamo delineato l'area di azione del nostro progetto, abbiamo analizzato le entità coinvolte e studiato come l'azienda per la quale produrremo il sistema informativo interagisce con queste entità.

Il passo successivo è quello di definire gli obiettivi della nostra applicazione, ovvero ciò che desideriamo che il nostro sistema consenta di fare. Questo include sia il supporto agli operatori durante i giri di raccolta, sia la pianificazione e la gestione complessiva di tutto ciò che ruota attorno agli itinerari, da parte dell'amministrazione. È essenziale, in questa fase, ascoltare le esigenze di entrambe le parti e fornire un portale unico e comunicante che permetta di organizzare al meglio l'intero processo.

Pertanto, ora procederemo a definire gli obiettivi che guideranno la progettazione e lo sviluppo della nostra applicazione. Tali obiettivi rappresentano le linee guida che ci assicureranno di rimanere focalizzati sulle necessità operative dell'azienda e dei propri lavoratori durante tutto il processo di realizzazione del sistema informativo.

2.1.1 Obiettivi dal lato amministrativo

Gli obiettivi dell'applicazione lato amministrativo mirano a garantire un'efficace gestione e pianificazione dei giri di raccolta, nonché a fornire un supporto completo all'amministrazione per la supervisione e il controllo delle operazioni. Di seguito, vengono illustrati gli obiettivi principali:

1. Accesso riservato:

- Consentire all'amministratore di accedere alla propria area riservata attraverso un sistema di autenticazione sicuro, il quale gli fornisca uno username e una password dedicati.

2. Gestione dei giri di raccolta:
   - Fornire una sezione principale in cui l'amministratore abbia una visione chiara di tutti i giri di raccolta e del loro stato attuale e, inoltre, possa interagirci, filtrarli, visualizzarne i dettagli ed eventualmente, associarli agli operatori disponibili.
   - Possibilità di creare, modificare ed eliminare facilmente i giri di raccolta.
3. Gestione degli operatori:
   - Cercare e gestire gli operatori, con possibilità di visualizzarne le informazioni dettagliate (anche relative alla loro disponibilità).
   - Possibilità di creare, modificare ed eliminare facilmente gli operatori.
4. Gestione dei bidoni:
   - Visualizzare efficacemente tutti i bidoni, con le informazioni relative.
   - Possibilità di creare, modificare ed eliminare facilmente i bidoni.
5. Storico degli itinerari:
   - Visualizzare uno storico dettagliato degli itinerari effettuati, con possibilità di ricerca in base a vari parametri, come, ad esempio, l'eventuale presenza di anomalie e i bidoni coinvolti.

2.1.2 Obiettivi lato operativo

Gli obiettivi dell'applicazione dal lato operativo mirano a facilitare il lavoro degli operatori, migliorando l'efficienza nelle operazioni quotidiane di raccolta dei rifiuti. Ecco gli obiettivi prefissati:

1. Accesso riservato:
   - L'operatore potrà accedere alla propria area riservata utilizzando uno username (fornito dall'amministrazione) dove sarà presente tutto il necessario per lo svolgimento delle sue attività.
2. Visualizzazione dei giri assegnati:
   - Fornire una sezione in cui l'operatore abbia una visione chiara di tutti i giri di raccolta a lui assegnati, con ogni dettaglio necessario all'esecuzione dell'itinerario, compresa la lista di bidoni da seguire.
3. Dichiarazione dell'itinerario:
   - Possibilità di dichiarare l'inizio e la fine dell'itinerario, con la registrazione precisa degli orari.
   - Possibilità, al termine del giro, di visualizzare un resoconto dell'itinerario effettuato e dichiarare eventuali anomalie, specificando il bidone e la tipologia di anomalia riscontrata.
4. Gestione dell'account:

- Visualizzare i propri dati personali, in un'area dedicata.
- Possibilità di segnalare all'amministrazione problemi legati ai propri dati.

5. Usabilità del sito web:

   - Vista la mobilità che caratterizza il lavoro degli operatori, l'applicativo deve essere ben ottimizzato per dispositivi mobili, in modo da poter essere consultato durante tutta l'operazione di scarico dei rifiuti.

2.2 Requisiti funzionali e non funzionali

Definiti gli obiettivi, il passo successivo è la traduzione di questi in requisiti funzionali e non funzionali.

Tale processo è molto importante, in quanto è in questa fase che l'aproccio diventa più tecnico e preciso.

I requisiti funzionali, infatti, sono specifiche dettagliate che descrivono le funzionalità e i comportamenti che il sistema deve implementare per soddisfare le necessità degli utenti e per raggiungere gli obiettivi prefissati.

2.2.1 Requisiti funzionali lato amministrativo

RF1. L'amministratore deve poter accedere alla propria area riservata utilizzando uno username e una password dedicati.

RF2. L'amministratore deve poter effettuare il logout dal sito web.

RF3. L'amministratore deve poter gestire tutti i giri di raccolta, potendoli consultare con l'aiuto di filtri specifici.

RF4. L'amministratore deve poter creare nuovi giri di raccolta.

RF5. L'amministratore deve poter modificare i giri di raccolta esistenti.

RF6. L'amministratore deve poter eliminare i giri di raccolta esistenti.

RF7. L'amministratore deve poter visualizzare i dettagli di ogni giro di raccolta.

RF8. L'amministratore deve poter assegnare i giri di raccolta agli operatori disponibili.

RF9. L'amministratore deve poter dissociare i giri di raccolta dagli operatori.

RF10. L'amministratore deve poter visualizzare uno storico di tutti i giri di raccolta effettuati, con dettagli che includono orari e anomalie.

RF11. L'amministratore deve poter filtrare e ordinare lo storico degli itinerari per specifici criteri.

RF12. L'amministratore deve poter visualizzare una lista di tutti gli operatori.

RF13. L'amministratore deve poter cercare gli operatori per nome o cognome.

RF14. L'amministratore deve poter visualizzare i dettagli di ogni operatore.

RF15. L'amministratore deve poter visualizzare le informazioni sugli operatori assegnati ai giri di raccolta.

RF16. L'amministratore deve poter aggiungiere nuovi operatori.

RF17. L'amministratore deve poter modificare le informazioni degli operatori.

RF18. L'amministratore deve poter eliminare gli operatori.

RF19. L'amministratore deve poter visualizzare una lista di tutti i bidoni.

RF20. L'amministratore deve poter filtrare i bidoni per tipologia di rifiuto.

RF21. L'amministratore deve poter aggiungere nuovi bidoni.

RF22. L'amministratore deve poter modificare le informazioni dei bidoni.

RF23. L'amministratore deve poter eliminare i bidoni.

2.2.2 Requisiti funzionali lato operativo

RF1. L'operatore deve poter accedere alla propria area riservata utilizzando il proprio username e la propria password.

RF2. L'operatore deve poter effettuare il logout dal sito web.

RF3. L'operatore deve poter accedere a una sezione in cui può visualizzare i propri dati personali.

RF4. L'operatore deve poter visualizzare i giri di raccolta assegnatigli, con tutti i dettagli per lo svolgimento.

RF5. L'operatore deve poter dichiarare l'inizio dell'itinerario, con registrazione dell'ora di partenza.

RF6. L'operatore deve poter visualizzare un resoconto dell'itinerario effettuato e dichiarare eventuali anomalie.

RF7. L'operatore deve poter dichiarare il termine dell'itinerario, con registrazione dell'ora di arrivo.

RF8. L'operatore deve poter visualizzare uno storico dei giri di raccolta che ha eseguito.

2.2.3 Requisiti non funzionali

I requisiti non funzionali descrivono come il sistema deve comportarsi e sono generalmente legati a criteri di qualità. Per quanto riguarda il nostro sistema, i requisiti non funzionali sono i seguenti:

RNF1. Il sito web deve essere responsive, garantendo una fruibilità ottimale da dispositivi mobili.

RNF2. Il sistema deve essere sviluppato come un'applicazione web, utilizzando tecnologie moderne come HTML5, CSS3, JavaScript, e un framework di backend appropriato (Laravel).

RNF3. L'interfaccia utente deve essere progettata per essere semplice e intuitiva, in modo da garantire un'esperienza utente positiva e ridurre al minimo il tempo necessario per l'apprendimento del sistema.

2.3 Casi d'uso

Definiti tutti i requisiti che l'applicativo web deve soddisfare, vediamo nel dettaglio il funzionamento e l'implementazione di essi nel nostro sistema attraverso i *casi d'uso*.

2.3.1 Definizione e struttura

Un caso d'uso descrive come un utente interagisce con un sistema per raggiungere un obiettivo specifico. Esso si concentra sulle azioni e interazioni dell'utente con il sistema, delineando la sequenza di eventi necessari per completare una funzione specifica.

La struttura utilizzata per la stesura dei casi d'uso è la seguente:

1. Titolo: un nome breve e significativo per il caso d'uso.
2. Attore Primario: chi interagisce direttamente con il sistema (ad esempio, Amministratore, Operatore).
3. Obiettivo: l'obiettivo che l'attore primario vuole raggiungere.
4. Pre-condizioni: lo stato del sistema prima che il caso d'uso possa iniziare.
5. Post-condizioni: lo stato del sistema dopo che il caso d'uso è stato completato.

6. Flusso Principale degli Eventi: la sequenza di passi che descrivono come l'attore interagisce con il sistema per raggiungere l'obiettivo.

7. Flussi Alternativi: eventuali variazioni o eccezioni al flusso principale.

Infine, l'analisi dei casi d'uso verrà eseguita solo su quei requisiti che non risultano immediati e presentano un flusso di eventi più complesso.

2.3.2 Premessa

Prima di illustrare i casi d'uso, è doveroso spiegare alcune scelte progettuali fatte, ma non implicite del contesto che si sta informatizzando.

Al giro di raccolta è stato attribuito uno *stato*, che varia durante la grande maggioranza delle operazioni svolte dalle due parti. Questo attributo è stato introdotto al fine di dare un idea chiara di tutto ciò che sta accadendo (all'amministratore) durante l'intero flusso di lavoro.

In particolare, gli stati che può assumere un giro di raccolta sono:

- *Non attivo*: questo è uno stato pensato soltanto per un fine gestionale dell'amministratore, in quanto l'amministratore sarà in grado di creare dei giri di raccolta che non necessariamente devono essere subito in funzione.
- *Da eseguire*: questo stato rappresenta il fatto che il giro di raccolta è in attesa di esecuzione e può, quindi, essere assegnato a un operatore compatibile.
- *Assegnato*: in questo stato il giro di raccolta si trova già assegnato a un operatore.
- *Eseguito*: stato che viene assunto dal giro di raccolta quando un operatore termina il suo itinerario; esso ritornerà automaticamente tra i giri "da eseguire" appena necessario.
- *Non terminato*: questo stato subentra quando un giro di raccolta presenta un'anomalia che richieda un intervento diretto.

Detto ciò passiamo a illustrare i casi d'uso.

2.3.3 Casi d'uso con attore l'amministratore

Relativamente al lato amministrativo si è deciso di mostrare i casi d'uso dei seguenti requisiti *RF3*, *RF4*, *RF8*, *RF10*.

UC3: Gestione dei Giri di Raccolta

- *Attore primario*: amministratore.
- *Obiettivo*: l'amministratore vuole visualizzare e poter interagire con tutti i giri di raccolta.
- *Pre-condizioni*: l'amministratore è autenticato.
- *Post-condizioni*: i giri di raccolta sono visualizzati e filtrati come richiesto.
- *Flusso principale degli eventi*:
    1. l'amministratore naviga alla sezione di gestione dei giri di raccolta;
    2. egli visualizza l'elenco totale dei giri di raccolta;
    3. `if` l'amministratore decide di applicare filtri specifici:

   (a) l'amministratore specifica i filtri desiderati;
   (b) il sistema aggiorna l'elenco secondo i filtri applicati;
4. `if` l'amministratore decide di interagire con un giro:
   (a) il sistema mostra il dettaglio del giro selezionato.

UC4: Creazione di un nuovo giro di raccolta

- *Attore primario*: amministratore.
- *Obiettivo*: l'amministratore vuole creare un nuovo giro di raccolta.
- *Pre-condizioni*: l'amministratore è autenticato.
- *Post-condizioni*: un nuovo giro di raccolta è creato e salvato nel database del sistema.
- *Flusso principale degli eventi*:
  1. l'amministratore naviga alla sezione di creazione di un nuovo giro di raccolta;
  2. egli inserisce i dati richiesti;
  3. egli interagisce con la sezione di creazione di un percorso di bidoni;
  4. il sistema mostra la lista dei bidoni pertinenti ai dati inseriti precedentemente;
  5. l'amministratore ordina i bidoni selezionati;
  6. `if` l'amministratore decide di attivare immediatamente il giro:
     (a) il sistema aggiorna lo stato del giro da "non attivo" a "da eseguire";
  7. l'amministratore conferma l'operazione di creazione.

UC8: Assegnazione dei giri di raccolta agli operatori

- *Attore primario*: amministratore.
- *Obiettivo*: l'amministratore vuole assegnare un giro di raccolta a un operatore idoneo.
- *Pre-condizioni*: l'amministratore è autenticato.
- *Post-condizioni*: lo stato del giro di raccolta passerà ad "assegnato" e l'operatore potrà interagirci.
- *Flusso principale degli eventi*:
  1. l'amministratore naviga nei dettagli del giro di raccolta che vuole assegnare;
  2. egli interagisce con la sezione di assegnazione;
  3. il sistema mostra la lista degli operatori attualmente disponibili;
  4. l'amministratore seleziona uno degli operatori;
  5. l'amministratore conferma l'operazione.

UC10: Visualizzazione dello storico dei giri di raccolta

- *Attore primario*: amministratore.
- *Obiettivo*: l'amministratore vuole visualizzare uno storico dei giri di raccolta effettuati, con tutti i dettagli.
- *Pre-condizioni*: l'amministratore è autenticato.

- *Flusso principale degli eventi*:
  1. l'amministratore accede alla sezione dello storico dei giri di raccolta;
  2. egli visualizza l'elenco totale degli itinerari effettuati, con le informazione sull'andamento e sull'operatore che l'ha svolto;
  3. `if` l'itinerario effettuato presenta anomalie:
     (a) il sistema permette di mostrare anche dettagli sulle anomalie interagendo con il giro.

Per quanto riguarda tutti i restanti requisiti funzionali (lato amministrativo), essi risultano intuitivi e immediati, e non necessitano, quindi, di ulteriori dettagli.

2.3.4 Casi d'uso con attore l'operatore

Relativamente al lato operativo si è optato per mostrare solo i casi d'uso del requisito funzionale *RF5* e *RF6*.

UC5: Dichiarazione inizio itinerario

- *Attore primario*: operatore.
- *Obiettivo*: l'operatore vuole dichiarare l'inizio del giro di raccolta che andrà a eseguire.
- *Pre-condizioni*: l'operatore è autenticato.
- *Post-condizioni*: il giro di raccolta risulterà iniziato, con il dato sull'ora di inizio.
- *Flusso principale degli eventi*:
  1. l'operatore seleziona il giro di raccolta da svolgere;
  2. egli interagisce con il pulsante per iniziare;
  3. il sistema mostra l'ora effettiva di inizio del giro e il pulsante per terminarlo.

UC6: Resoconto itinerario effettuato

- *Attore primario*: operatore.
- *Obiettivo*: l'operatore vuole visualizzare un resoconto del giro di raccolta appena svolto e segnalare eventuali anomalie.
- *Pre-condizioni*: l'operatore è autenticato.
  1. l'operatore termina il giro di raccolta;
  2. il sistema visualizza il percorso eseguito e tutti i bidoni svuotati;
  3. `if` l'operatore ha riscontrato anomalie:
     (a) l'operatore indica i bidoni e il tipo di anomalia avvenuta su di essi;
     (b) il sistema registra l'anomalia.

Terminato questo processo abbiamo una chiara roadmap da seguire durante lo sviluppo del sistema informativo, assicurandoci di mantenere il focus sulle necessità operative dell'azienda e dei suoi lavoratori.

Nel prossimo capitolo ci concentreremo su quali sono le informazioni che ci interessa gestire per ogni entità. Progettando, quindi, la componente dati (database) del nostro applicativo web.

CAPITOLO 3

Progettazione della componente dati

In questo capitolo ci concentreremo sulla definizione e l'organizzazione dei dati necessari per il corretto funzionamento della nostra applicazione web, nonché sulla loro modellazione concettuale. Una raccolta e gestione accurata dei dati è essenziale per il successo del sistema, poiché costituisce la base per tutte le operazioni future. Partendo dai requisiti funzionali delineati nel capitolo precedente, approfondiremo l'identificazione delle entità principali, definendone gli attributi e le relazioni.

Questo processo di analisi e definizione ci condurrà alla costruzione di un modello concettuale chiaro e strutturato, attraverso la creazione di uno schema Entity-Relationship (E-R). Tale schema ci permetterà di visualizzare graficamente le entità e le loro interconnessioni, facilitando la comprensione della struttura e dell'immagazzinamento delle informazioni.

Alla fine di questo capitolo, avremo un modello concettuale solido che fungerà da base per le fasi successive di progettazione fisica e implementazione del database. Questo ci consentirà di realizzare un database completo, pronto a supportare tutte le funzionalità previste dalla nostra applicazione web.

3.1 Acquisizione dei dati

La raccolta e la gestione dei dati sono elementi cruciali per il successo di qualsiasi sistema informatico. Dati accurati e ben strutturati rappresentano una delle risorse più preziose per un'organizzazione. Nel contesto della nostra applicazione web, è essenziale identificare con precisione i dati necessari per ogni entità delineata nei capitoli precedenti. Questa fase di definizione dei dati è strettamente guidata dai requisiti funzionali, descritti nel capitolo precedente, e ha l'obiettivo di garantire che gli utenti finali possano svolgere tutte le attività necessarie in modo efficiente e preciso.

3.1.1 Dati di interesse

Per ogni entità individuata, saranno, quindi, elencati tutti i dati di nostro interesse:
Degli *operatori* ci interessa conservare i seguenti dati anagrafici:

- nome;
- cognome;
- data di nascita;
- residenza.

Relativamente ai *giri di raccolta*, i dati necessari da conservare includono:

- ora di inizio;
- ora di fine;
- tipologia di rifiuto da raccogliere;
- frequenza settimanale;
- giorni in cui è necessario effettuare il giro di raccolta;
- percorso, formato dalla lista ordinata dei bidoni.

Passando ai *bidoni*, per il tipo di sistema che verrà realizzato, ci interessano:

- tag univoco;
- tipologia di rifiuto;
- indirizzo di collocazione.

Per quanto riguarda gli *itinerari effettuati* dagli operatori, sono necessari:

- data di esecuzione;
- giro di raccolta di riferimento;
- ora di inizio effettiva;
- ora di fine effettiva;
- presenza di anomalie;
- operatore che ha eseguito l'itinerario.

Infine, per gestire le *anomalie*, i dati da acquisire comprendono:

- itinerario effettuato;
- bidone su cui è avvenuta l'anomalia;
- tipologia di anomalia;
- breve descrizione dell'accaduto.

3.1.2 Motivazioni sulle scelte dei dati

Il nostro sistema informatico si concentra sul supporto della *programmazione* dei giri di raccolta e sul loro *svolgimento* da parte degli operatori. Per soddisfare questi requisiti, è essenziale conservare i dati temporali dei giri di raccolta al fine di gestire la loro programmazione nel tempo; allo stesso modo risulta importante tenere traccia della tipologia di rifiuto da raccogliere e del percorso di bidoni da seguire, dati necessari agli operatori per eseguire i giri di raccolta assegnati.

Come accennato in precedenza, un percorso di un giro di raccolta è una lista ordinata di *bidoni*. Pertanto, i dati relativi alla collocazione dei bidoni e alla tipologia di rifiuto che contengono devono essere acquisiti. Il dato sulla collocazione dei bidoni permette di creare

il tracciato del giro, mentre la tipologia di rifiuto è importante per limitare i bidoni da raccogliere, poiché, ad esempio, un giro di raccolta di plastica includerà solo bidoni che contengono plastica.

Per quanto riguarda gli *operatori*, esclusa la gestione dei mezzi, non è necessario verificare il tipo di patente posseduta. Di conseguenza, la loro disponibilità per l'assegnazione di giri di raccolta è determinata solo dal fatto che non siano impegnati in un altro giro.

Al fine di creare uno storico degli *itinerari effettuati* e gestire eventuali anomalie urgenti, è necessario che l'operatore possa segnalare come prima cosa l'orario di inizio e di fine effettivo del giro. Questi dati verranno confrontati con quelli prestabiliti al termine del giro di raccolta. Inoltre, per differenziare gli itinerari effettuati tra loro e avere una visione più di insieme, verrà registrata la data in cui è stato svolto l'itinerario, chi è l'operatore che l'ha eseguito e a quale giro di raccolta si riferisce. Infatti, sebbene i giri di raccolta siano unici, gli itinerari effettuati relativi a quei giri sono molteplici e ripetuti nel tempo; quindi, senza l'acquisizione di questi dati aggiuntivi, potrebbe risultare difficoltoso navigare tra di essi.

Infine, per quanto riguarda le *anomalie*, è importante registrare i dati relativi all'itinerario effettuato, i bidoni coinvolti e, per una migliore categorizzazione, consentire all'operatore di selezionare una tipologia di anomalia prestabilita o di fornire una breve descrizione sull'accaduto, qualora l'anomalia non rientri tra quelle standard.

3.1.3 Spunti per le fasi successive

Ora che sono stati chiariti i dati importanti da acquisire per ogni entità coinvolta, possiamo fare alcune osservazioni utili per la progettazione della base di dati. Spesso, i dati necessari per una determinata entità sono relativi ad altre entità. Ad esempio, tra i dati del giro di raccolta troviamo la lista dei bidoni oppure nei dati degli itinerari effettuati sono presenti informazioni relative al giro di raccolta e all'operatore che l'ha eseguito.

Questa osservazione sottolinea l'importanza di progettare il database in modo tale da evitare ridondanze e garantire la coerenza dei dati.

3.2 Progettazione concettuale

Arrivati a questo punto siamo in possesso di tutti gli elementi necessari alla realizzazione di un modello concettuale per la nostra base di dati.

3.2.1 Perché è importante?

Il modello concettuale di un database è una rappresentazione ad alto livello dei dati che verranno memorizzati nel sistema. Esso serve come strumento per definire chiaramente le entità, gli attributi e le corrispettive relazioni, senza preoccuparsi dei dettagli tecnici dell'implementazione.

Uno dei vantaggi più significativi di questo modello, oltre a fornire una chiara comprensione del dominio applicativo, è la sua indipendenza dalla tecnologia specifica di implementazione. Questo permette di utilizzarlo come base per la progettazione di sistemi su diverse piattaforme.

In conclusione, il modello concettuale rappresenta il punto di partenza per la successiva fase di progettazione logica, dove vengono definiti dettagli più tecnici e specifici.

3.2.2 Modello E-R introduzione

Il modello E-R (Entity-Relationship) è una rappresentazione grafica dei dati di un sistema informativo. Esso prevede i seguenti costrutti fondamentali:

- *Entità*: rappresentate da rettangoli, modellano i concetti principali del sistema.
- *Attributi*: rappresentati da cerchi, descrivono le proprietà delle entità.
- *Relazioni*: rappresentate da rombi, indicano come le entità interagiscono tra loro.

Per la realizzazione del modello E-R, si è optato per una strategia ibrida (usando elementi delle due strategie bottom-up e top-down). Pertanto, una volta individuate le entità principali che interagiscono nel nostro sistema, queste sono state decomposte (quando necessario), integrate e dettagliate.

3.2.3 Entità principali e relazioni

Le entità principali presenti nel nostro contesto erano già state individuate nei capitoli precedenti e sono quelle rappresentate in Figura 3.1; in particolare esse sono: Operatore, Giro di raccolta, Itinerario effettuato, Bidone e Anomalia.

Figura 3.1: Entità coinvolte nel nostro contesto

Definite quali sono le entità principali, vediamo ora le relazioni che intercorrono tra di esse, così da realizzare uno schema scheletro (Figura 3.2) dal quale partire per poi realizzare il vero e proprio modello E-R.
Spieghiamo nel dettaglio le relazioni presenti in Figura 3.2:

- *Assegnamento*: agli operatori disponibili vengono *assegnati* i giri di raccolta da eseguire.
- *Inserimento*: una volta terminato il giro di raccolta, l'operatore *inserirà* all'interno del database tutte le informazioni relative all'itinerario che ha appena effettuato.
- *Riferimento*: un itinerario effettuato si *riferisce* a un giro di raccolta.
- *Riscontro*: un'anomalia si *riscontra* in un itinerario effettuato.
- *Definizione*: un'anomalia è *definita* su un bidone presente nel percorso dell'itinerario effettuato.

Proseguiamo la progettazione della nostra componente dati analizzando un'entità alla volta. A ciascuna entità verranno associati i corrispettivi attributi e la cardinalità delle relazioni che le legano; termineremo con l'aggiunta di entità "di supporto", dove necessario.

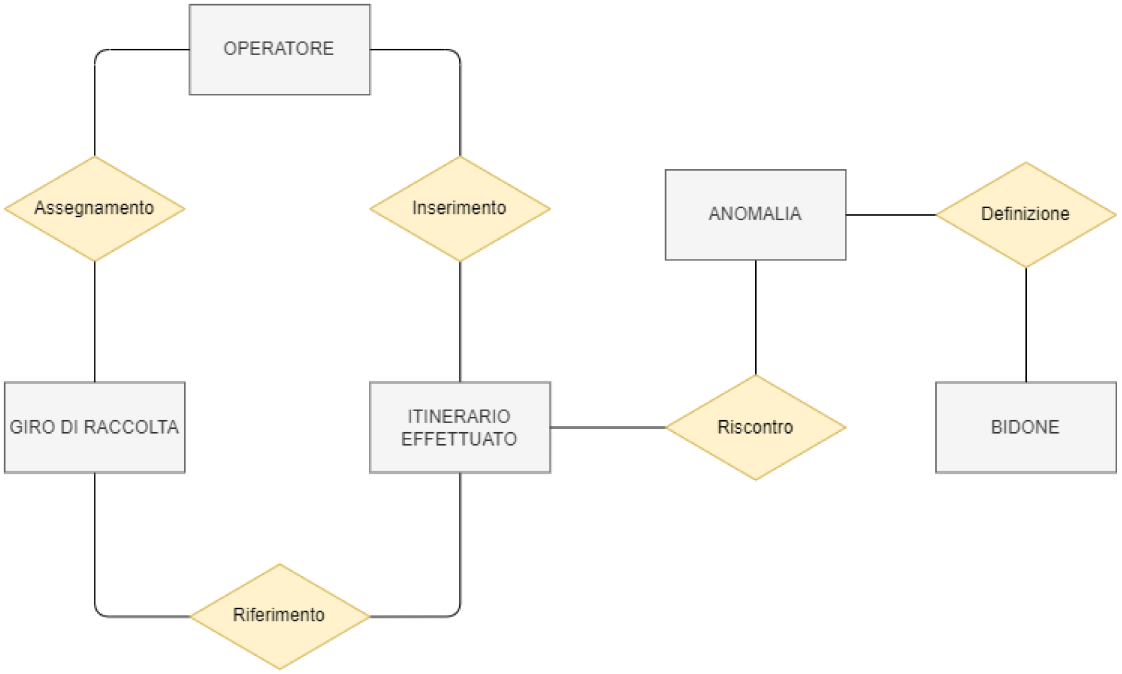

Figura 3.2: Schema scheletro relativo al modello E-R

3.2.4 Operatore

Per quanto riguarda l'entità Operatore, dalle sezioni precedenti sappiamo di dover salvare nella base di dati le informazioni relative all'anagrafica: nome, cognome, data di nascita, residenza.

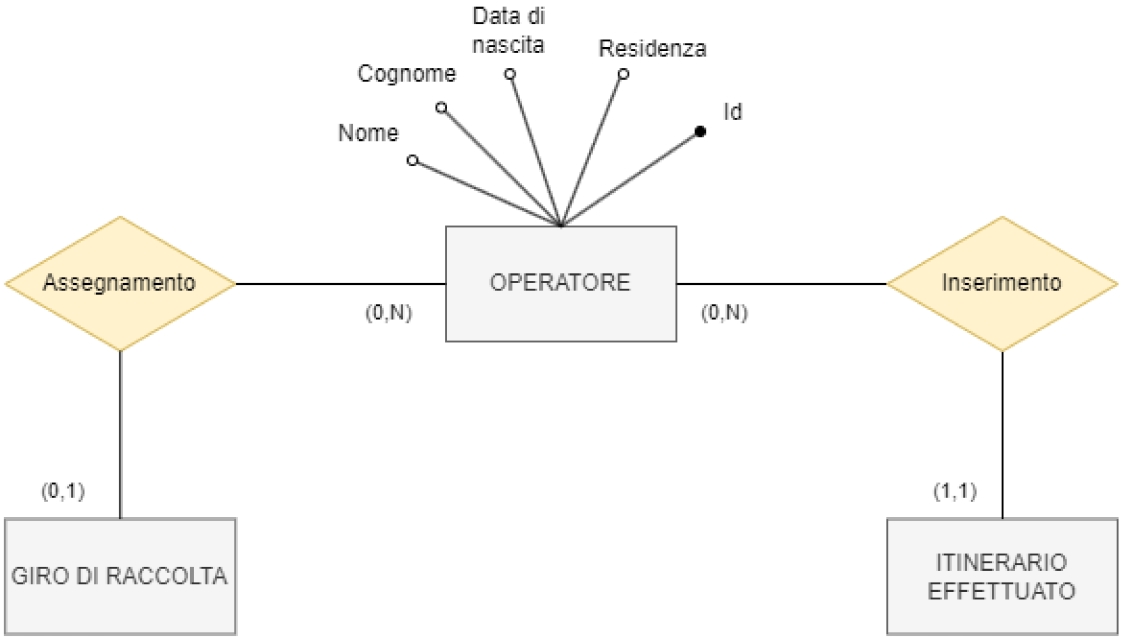

Figura 3.3: Schema relativo all'operatore

La cardinalità della relazione *assegnamento*, mostrata in Figura 3.3, indica che l'assegnamento di uno o più giri di raccolta a un operatore è opzionale e senza una limitazione massima predefinita. D'altra parte, ogni giro di raccolta può essere assegnato, nello stesso istante, al massimo a un operatore.

Per quanto riguarda, invece, la cardinalità della relazione *inserimento*, presente tra l'operatore e l'itinerario effettuato, si è considerato che ogni operatore può inserire nella base dati un numero illimitato di itinerari effettuati; invece, ogni singolo itinerario effettuato può essere inserito da un solo operatore (la presenza di un operatore non è opzionale).

L'attributo *id* rappresenta un identificatore univoco che permetterà al nostro database di riconoscere univocamente ogni operatore.

3.2.5 Giro di raccolta

Nei giri di raccolta, i dati che ci interessa conservare includono: ora di inizio, ora di fine, tipologia di rifiuto, frequenza settimanale, giorni e percorso dei bidoni. Tuttavia, salvare efficientemente tutti questi dati presenta alcune complessità.

Pertanto, sono stati adottati degli stratagemmi per codificare correttamente le seguenti situazioni:

1. Lo stesso giro di raccolta può essere effettuato in più giorni della settimana.
2. Il percorso di un giro di raccolta è costituito da una lista ordinata di bidoni.

Per comprendere come procedere, è essenziale capire che le entità descritte saranno successivamente tradotte in tabelle del database, e ogni istanza di un'entità rappresenterà una riga specifica della tabella. Di conseguenza, ogni riga può contenere un singolo dato per ogni attributo.

Da questo principio deriva che non è possibile rappresentare i giorni in cui un giro di raccolta deve essere eseguito come un singolo attributo dell'entità Giro di Raccolta, poiché in tal modo il giro sarebbe legato solo a un unico giorno della settimana. Lo stesso vale per la lista dei bidoni. Inserendo il percorso come attributo dell'entità Giro di Raccolta, non sarebbe possibile associare una lista di bidoni, ma soltanto un singolo bidone. Per gestire le casistiche descritte, è necessario creare nuove entità e stabilire relazioni uno-a-molti con l'entità di partenza.

L'approccio utilizzato è illustrato in Figura 3.4. In questo schema, si può osservare la creazione di una nuova entità chiamata *Giorno*, che contiene come unico attributo il nome del giorno corrispondente. Questa entità, il cui scopo è di rappresentare tutti i giorni della settimana, è collegata all'entità Giro di Raccolta tramite la relazione *Programmazione*. Tale relazione consente di programmare i giri di raccolta nei vari giorni della settimana. La cardinalità della relazione *Programmazione* indica che lo stesso giro di raccolta può essere programmato per più giorni, riflettendo esattamente il comportamento desiderato.

Per quanto riguarda il percorso, l'approccio adottato è simile, ma coinvolgendo anche l'entità Bidone. È stata creata una nuova entità denominata *Operazione di Raccolta*, la quale possiede un attributo ordinale (rappresentante l'ordine del bidone nella lista) e si trova in relazione sia con l'entità Giro di Raccolta che con l'entità Bidone.

Le relazioni *Specificazione* e *Ordinamento*, visibili in Figura 3.4, e le loro cardinalità, indicano il funzionamento di un'operazione di raccolta (intesa come una tupla di dati: giro di raccolta, bidone, ordinale), la quale consente di definire correttamente un percorso ordinato di bidoni, legato a un giro di raccolta, senza creare problemi di ridondanza o intaccare l'atomicità delle entità principali.

È importante notare che un'operazione di raccolta, per esistere, richiede la presenza di almeno un giro di raccolta e almeno un bidone, rendendo la loro presenza obbligatoria.

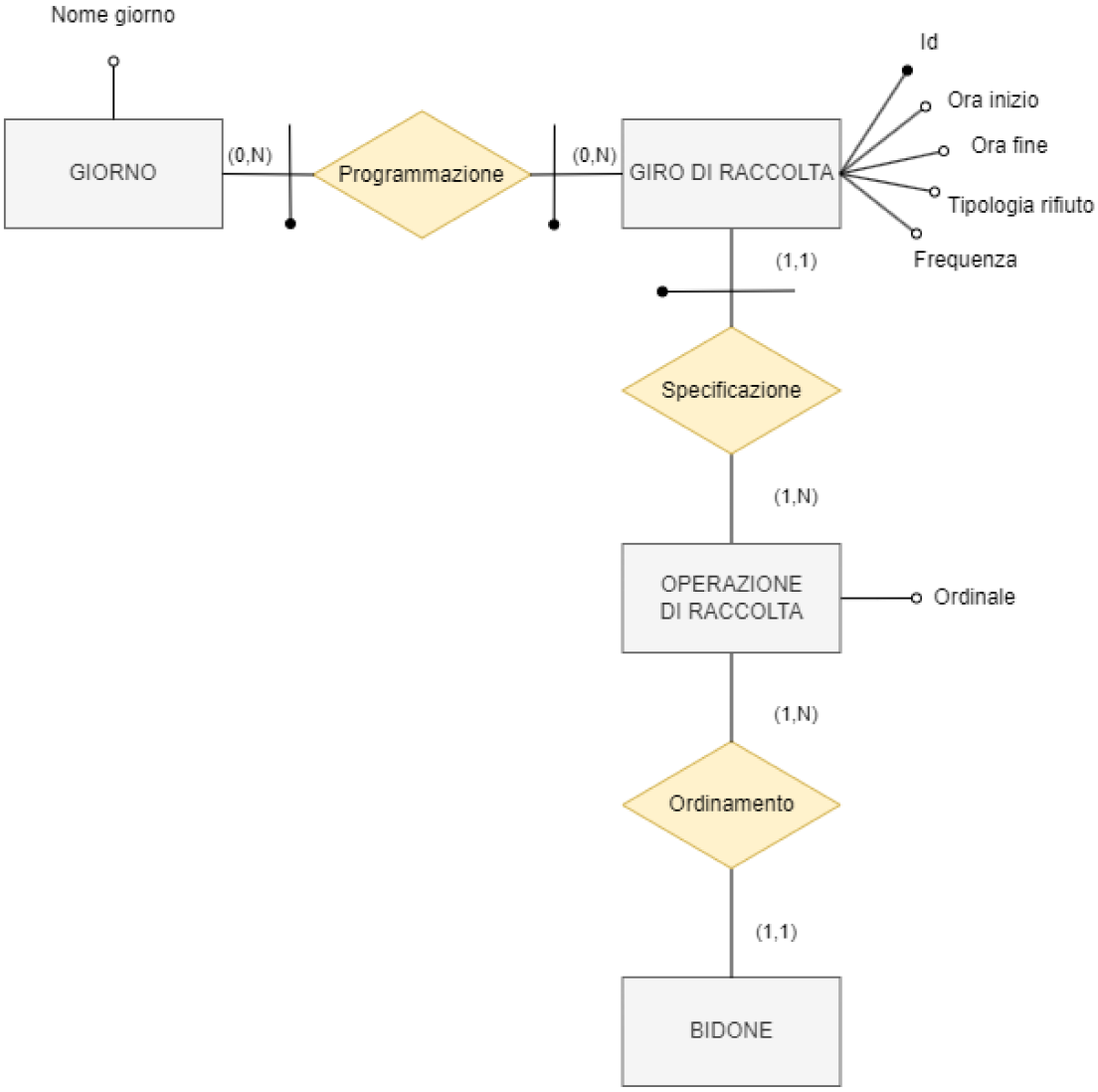

Figura 3.4: Schema relativo al giro di raccolta

3.2.6 Itinerario effettuato

Per gestire correttamente gli itinerari effettuati, è necessario raccogliere i seguenti dati: data di esecuzione, ora di inizio, ora di fine, giro di raccolta, anomalie e operatore. La sezione dello schema E-R relativa agli itinerari effettuati è mostrata in Figura 3.5.

Attraverso le relazioni *Riferimento* e *Inserimento* ci assicuriamo che ogni itinerario effettuato contenga tutte le informazioni richieste.

Per quanto riguarda le anomalie, l'entità Itinerario Effettuato include un attributo booleano "anomalia" che indica se, durante l'itinerario, sono state riscontrate anomalie. I dettagli specifici delle anomalie saranno, poi, gestiti tramite l'entità principale con lo stesso nome.

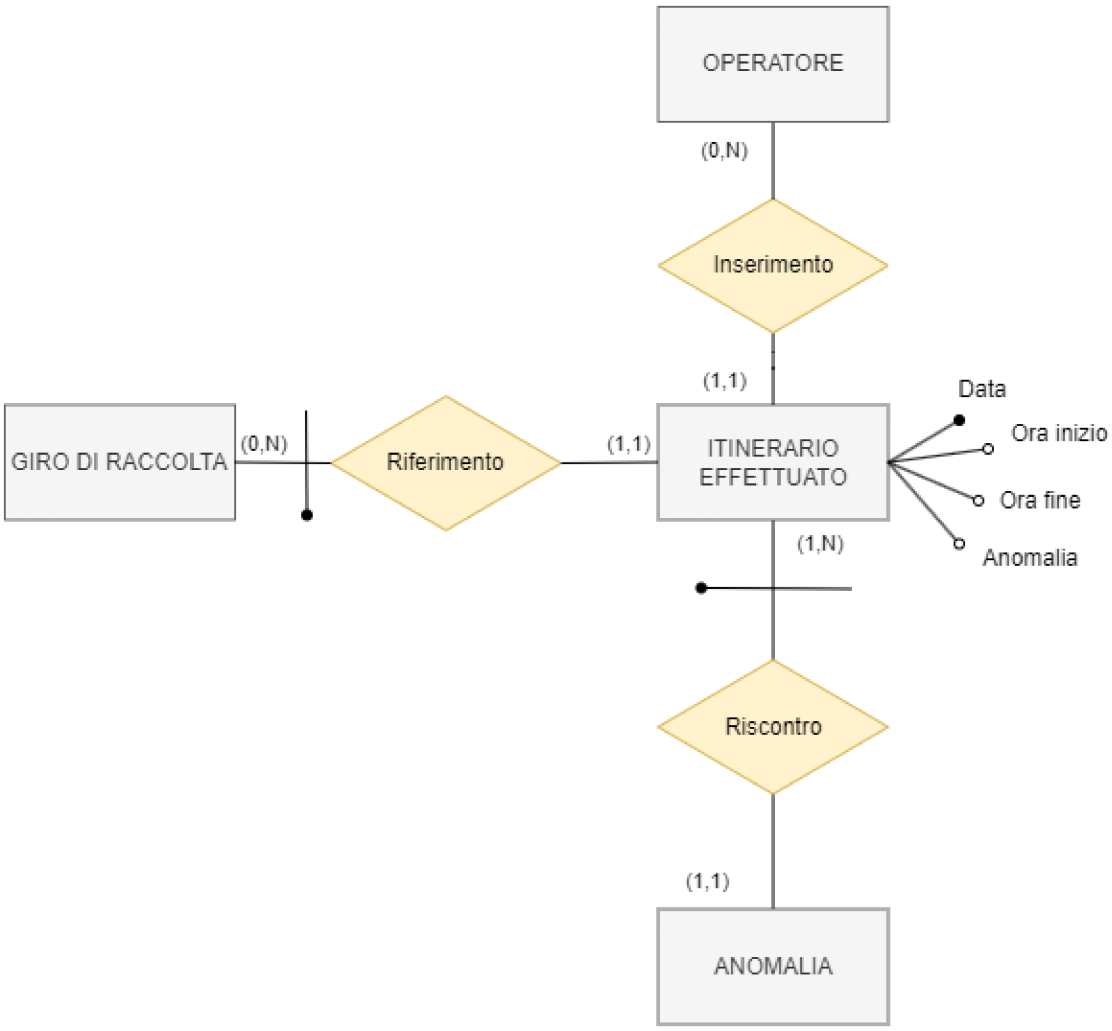

Figura 3.5: Schema relativo all'itinerario effettuato

3.2.7 Bidone e Anomalia

I dati relativi ai bidoni da conservare includono: tag, tipologia di rifiuto, e indirizzo di collocazione. Per le anomalie, i dati da salvare sono: tipologia, descrizione, itinerario effettuato relativo e bidone su cui è avvenuta l'anomalia. La soluzione adottata è rappresentata in Figura 3.6.

In questo schema, attraverso le relazioni *Riscontro* e *Definizione*, le informazioni dell'entità Itinerario Effettuato e dell'entità Bidone confluiscono all'interno dell'entità Anomalia. In questo modo otteniamo il risultato desiderato.

Una volta completata la creazione delle varie componenti dello schema E-R, queste verranno integrate per formare lo schema E-R finale. Durante questo processo, verranno apportate le necessarie modifiche per ottimizzare il database in base alla natura web del nostro applicativo.

Successivamente, lo schema E-R sarà tradotto nel modello logico, trasformando le entità in tabelle. Questo passaggio di traduzione è molto importante poiché le tabelle costituiranno il database operativo, che verrà utilizzato per memorizzare tutti i dati del nostro sistema informatico.

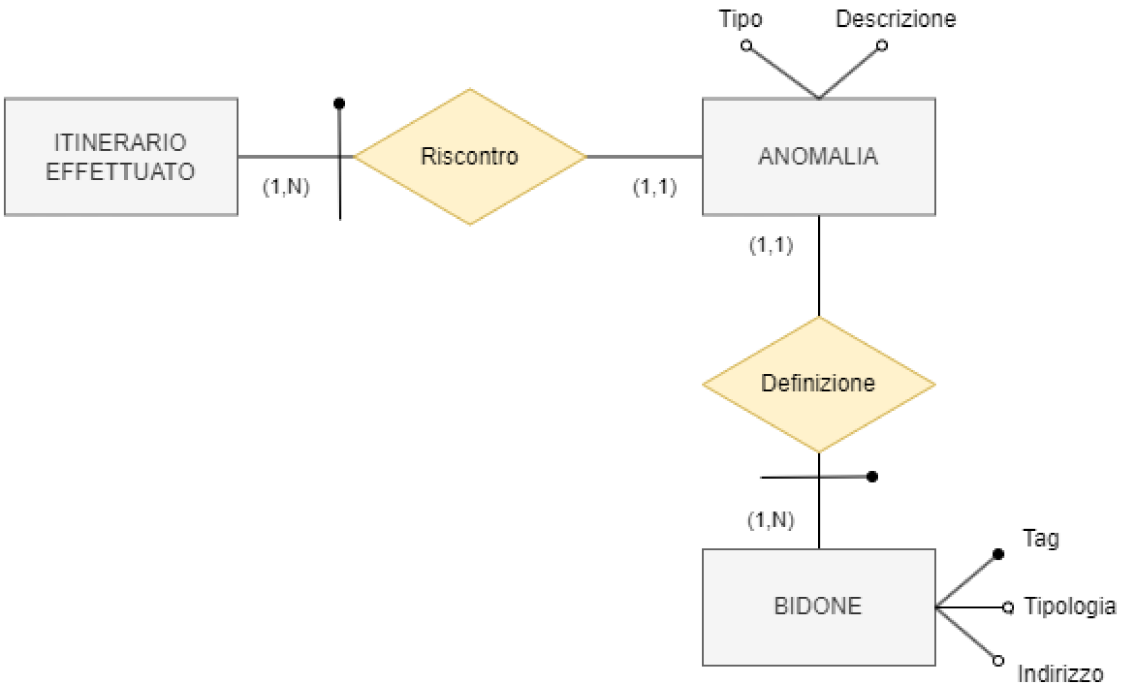

Figura 3.6: Schema relativo al bidone e all'anomalia

3.3 Progettazione logica

Il passo successivo da compiere è quello di tradurre il modello concettuale in un modello logico che rappresenti gli stessi dati espressi precedentemente in maniera più efficiente e funzionale. Prima di fare ciò, però, introduciamo le modifiche effettuate allo schema E-R in modo che sia funzionale all'applicativo web che verrà realizzato.

3.3.1 Modifiche schema E-R

All'entità Operatore sarà aggiunto un attributo *username*, fondamentale per l'accesso all'area riservata all'interno dell'applicazione. Considerando che, come descritto nel capitolo precedente, gli utenti finali si dividono in Operatori e Amministratori, l'entità Operatore includerà anche un attributo *ruolo*. Questo attributo sarà essenziale per distinguere tra i diversi tipi di utenti all'interno del sito web, consentendo di mostrare i dati corretti e di fornire le funzionalità appropriate in base al ruolo dell'utente connesso.

Un'altra modifica derivante dai requisiti definiti è l'inclusione dell'attributo *stato* nell'entità Giro di Raccolta. Questo attributo permetterà di tenere traccia dello stato di ciascun giro di raccolta durante tutte le operazioni svolte dagli amministratori e dagli operatori.

Apportate queste modifiche, vediamo ora lo schema E-R finale, che servirà da base per la traduzione nello schema logico.

3.3.2 Schema E-R finale

In Figura 3.7 è riportato lo schema finale, ottenuto dalla fusione di tutti i processi svolti in precedenza.

Notiamo che nello schema E-R, sono presenti un numero considerevole di chiavi esterne; queste permettono di comprendere meglio lo schema, ma risultano particolarmente inefficienti dal punto di vista operativo.

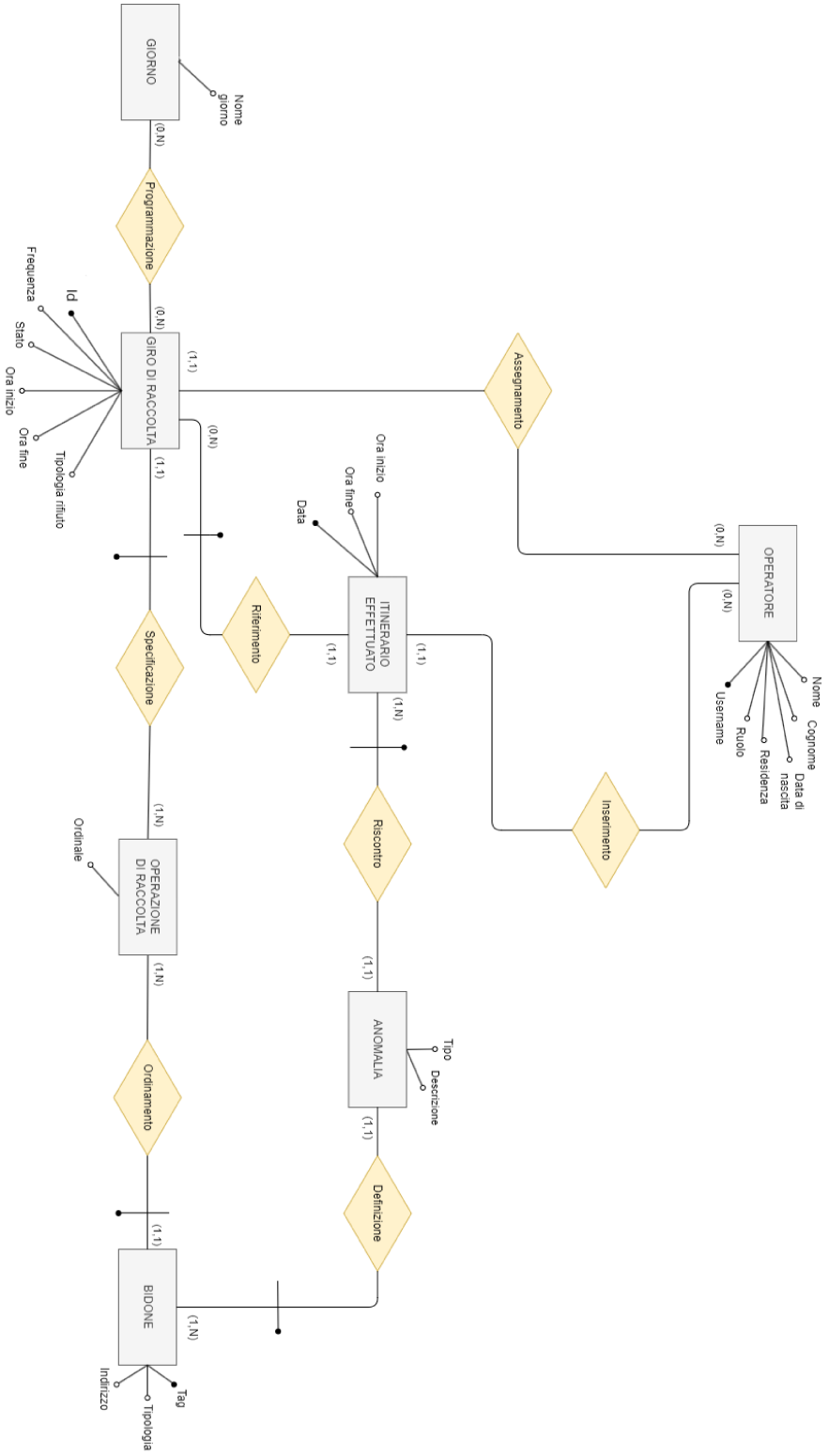

Figura 3.7: Schema E-R finale

Per questo motivo, nella traduzione al modello logico, ogni entità sarà dotata di un proprio id univoco come *chiave primaria*. Questo accorgimento consente di eseguire le operazioni di recupero dei dati tra le varie entità in maniera più precisa e meno dispendiosa.

3.3.3 Traduzione al modello logico

Come menzionato precedentemente, l'entità *Operatore* è stata tradotta in *Utente* nel modello logico, rendendola universale per entrambi i ruoli gestiti dalla nostra applicazione; inoltre, ogni entità nel modello logico ha un attributo *id* che funge da chiave primaria.

I passaggi necessari alla traduzione dello schema E-R nelle tabelle del database (modello logico), sono i seguenti:

1. Entità e Attributi:
    - ogni entità nel modello E-R è stata tradotta in una tabella nel modello logico;
    - gli attributi delle entità sono diventati colonne delle rispettive tabelle.
2. Relazioni:
    - Le relazioni nel modello E-R sono state tradotte in chiavi esterne nelle tabelle del modello logico, mantenendo le referenze tra le entità.

Ecco un dettaglio delle entità con gli attributi aggiuntivi e i vincoli di riferimento:

- *Itinerario Effettuato*: include gli attributi *giro di raccolta* e *operatore*, legati da un vincolo di riferimento agli id delle entità Giro di Raccolta e Utente.
- *Anomalia*: include gli attributi *itinerario effettuato* e *bidone*, legati da un vincolo di riferimento agli id delle entità Itinerario Effettuato e Bidone.

Le relazioni molti-a-molti sono state tradotte in tabelle di collegamento che contengono chiavi esterne delle tabelle coinvolte:

- *Relazione Programmazione*: tradotta nella tabella *programmazione* con gli attributi id, giro di raccolta, giorno, dove *giro_raccolta* e *giorno* sono chiavi esterne.
- *Relazione Assegnamento*: tradotta nella tabella assegnamento con gli attributi id, giro di raccolta, operatore, dove *giro_raccolta* e *operatore* sono chiavi esterne.

Per quanto riguarda la gestione della lista dei bidoni, la relazione era già precedentemente stata trasformata nell'entità Operazione di Raccolta, che, nello schema logico, viene rappresentata dalla tabella *operazione_raccolta*, con gli attributi: *id*, *ordinale*, *bidone* e *giro_raccolta*, dove *bidone* e *giro_raccolta* sono chiavi esterne.

Relativamente alla tabella *assegnamento* invece, all'interno dell'applicazione verrà implementato un vincolo che impedirà di assegnare lo stesso giro di raccolta a più di un operatore contemporaneamente.

Una rappresentazione completa delle entità e delle relazioni, con i rispettivi vincoli di riferimento, è illustrata nella Figura 3.8. Questa figura mostra come tutte le entità e relazioni siano state tradotte e come gli attributi siano legati da vincoli di riferimento.

3.3.4 Regole di vincolo

Alcuni dei dati che conserveremo nella nostra base di dati hanno vincoli sul loro contenuto. Tali vincoli sono i seguenti:

- Il valore associato all'attributo *ora_inizio* di un giro di raccolta deve essere compreso tra le ore 00:00 e le ore 18:00. Questo perché, essendo la durata standard dei giri di raccolta di 6 ore, un inizio dopo le 18:00 comporterebbe il passaggio al giorno successivo, creando problemi con la programmazione giornaliera dei giri di raccolta.

Entità / Relazioni	Traduzione	Vincoli di riferimento
UTENTE	users (**id**, username, nome, cognome, data_di_nascita, residenza,ruolo)	
GIRO DI RACCOLTA	giro_raccolta (**Id**, ora_inizio, ora_fine, tipologia_rifiuto, frequenza, stato)	
ITINERARIO EFFETTUATO	itinerario_effettuato (**id**, data_itinerario, *giro_raccolta*, ora_inizio, ora_fine, anomalia, *operatore*)	giro_raccolta -> giro_raccolta.id operatore -> users.id
BIDONE	bidone (**id**, tipologia rifiuto, indirizzo)	
ANOMALIA	anomalia (**id**, *itinerario_effettuato*, *bidone*, tipo, descrizione)	itinerario_effettuato -> itinerario_effettuato.id bidone -> bidone.id
GIORNO	giorno (**id**, nome)	
OPERAZIONE DI RACCOLTA	operazione_raccolta (**id**, ordinale, *bidone*, *giro_raccolta*)	giro_raccolta -> giro_raccolta.id bidone -> bidone.id
PROGRAMMAZIONE	programmazione (**id**, *giro_raccolta*, *giorno*)	giro_raccolta -> giro_raccolta.id giorno -> giorno.id
ASSEGNAMENTO	assegnamento (**id**, *giro_raccolta*, *operatore*)	giro_raccolta -> giro_raccolta.id operatore -> users.id

Figura 3.8: Modello logico

- L'attributo *nome* della tabella *giorno*, può assumere solo i seguenti valori: lunedì, martedì, mercoledì, giovedì, venerdì, sabato e domenica.
- La *tipologia_rifiuto* sia nella tabella *giro_raccolta* che nella tabella *bidone*, può essere solamente: carta, plastica, vetro, organico e indifferenziata.
- L'attributo *tipo* nella tabella *anomalia* può assumere solo i seguenti valori prefissati: "non ritirato", "ritirato parzialmente", "altro".
- Il valore assunto dall'attributo *ora_fine* nelle tabelle *giro_raccolta* e *itinerario_effettuato* non può precedere il valore dell'attributo *ora_inizio*.
- L'attributo *frequenza* nella tabella *giro_raccolta*, essendo relativo alla frequenza settimanale, può variare da un minimo di 1 a un massimo di 7.
- L'attributo *stato* nella tabella *giro_raccolta*, può assumere solo i seguenti valori: "non attivo", "da eseguire", "assegnato", "eseguito", "non terminato".

3.3.5 Codifica SQL

L'implementazione del database nel progetto avviene attraverso MySQL. Di seguito vengono presentate alcune query per la creazione delle tabelle relative alle nostre entità principali.

Tabella UTENTE:

```
CREATE TABLE users(
  id INT PRIMARY KEY AUTO_INCREMENT,
  username VARCHAR(30) NOT NULL,
  nome VARCHAR(30) NOT NULL,
  cognome VARCHAR(30) NOT NULL,
  dataNascita DATE NOT NULL,
  residenza VARCHAR(50) NOT NULL,
  ruolo VARCHAR(30) NOT NULL DEFAULT “operatore”
);
```

Tabella GIRO DI RACCOLTA:

```
CREATE TABLE giro_raccolta (
  id INT PRIMARY KEY AUTO_INCREMENT,
  ora_inizio TIME NOT NULL,
  ora_fine TIME NOT NULL,
  tipologia_rifiuto VARCHAR(30) NOT NULL,
  frequenza INT NOT NULL CHECK (frequenza BETWEEN 1 AND 7),
  stato VARCHAR(30) NOT NULL CHECK (stato IN (“non_attivo”,
  “da_eseguire”, “assegnato”, “eseguito”, “non_terminato”))
);
```

Tabella ITINERARIO EFFETTUATO:

```
CREATE TABLE itinerario_effettuato (
  id INT PRIMARY KEY AUTO_INCREMENT,
  data_itinerario DATE NOT NULL,
  giro_raccolta INT NOT NULL,
  ora_inizio TIME NOT NULL,
  ora_fine TIME,
  anomalia BOOLEAN NOT NULL DEFAULT FALSE,
  operatore INT NOT NULL,
  FOREIGN KEY (giro_raccolta) REFERENCES giro_raccolta(id),
  FOREIGN KEY (operatore) REFERENCES users(id)
);
```

Tabella BIDONE:

```
CREATE TABLE bidone (
  id INT PRIMARY KEY AUTO_INCREMENT,
  tipologia_rifiuto VARCHAR(30) NOT NULL,
  indirizzo VARCHAR(50) NOT NULL
);
```

Tabella OPERAZIONE DI RACCOLTA:

```
CREATE TABLE operazione_raccolta (
  id INT PRIMARY KEY AUTO_INCREMENT,
  ordinale INT NOT NULL,
  bidone INT NOT NULL,
  giro_raccolta INT NOT NULL,
  FOREIGN KEY (bidone) REFERENCES bidone(id),
  FOREIGN KEY (giro_raccolta) REFERENCES giro_raccolta(id)
);
```

La struttura descritta in questo capitolo per la base di dati garantisce un'organizzazione coerente e logica dei dati, facilitando le operazioni di inserimento, aggiornamento e interrogazione. La fase successiva di progettazione riguarderà l'implementazione della componente applicativa, dove il database svolgerà un ruolo chiave nell'architettura del sistema.

CAPITOLO 4

Progettazione della componente applicativa

In questo capitolo, l'obiettivo è ottenere una visione chiara sulle scelte progettuali della componente applicativa del sistema informativo, concentrandosi sui dettagli tecnici e architetturali che ne definiscono il funzionamento. Esploreremo i principi di base e le scelte tecnologiche che guideranno lo sviluppo del sistema, garantendo un'integrazione fluida tra il frontend e il backend.

Inizieremo con una panoramica dell'architettura del sistema, analizzando in particolare il pattern MVC (Model-View-Controller) che costituisce la struttura portante della nostra applicazione. Successivamente, discuteremo il framework scelto per facilitare lo sviluppo e la manutenzione del sistema.

Proseguiremo con la progettazione del frontend, evidenziando le tecnologie utilizzate e una regola di design che consente un utilizzo logico dei colori durante la realizzazione dei mockup. Presenteremo, quindi, i vari mockup che illustrano le diverse viste dell'applicazione.

Infine, ci concentreremo sulla progettazione del backend, descrivendo le tecnologie impiegate, il ruolo del database e dell'ORM, e le misure di sicurezza adottate per proteggere i dati e garantire l'integrità del sistema.

4.1 Architettura del sistema

Per comprendere a fondo la struttura della nostra applicazione web è essenziale analizzare l'architettura del sistema. Questo ci permetterà di avere una visione chiara dei componenti principali che costituiscono l'applicazione e di come essi interagiscono tra loro garantendo un funzionamento coerente ed efficiente.

L'architettura di un'applicazione web moderna si compone generalmente di tre elementi fondamentali, ciascuno con un ruolo specifico ma interdipendente. Comprendere questi elementi ci consente di progettare un sistema robusto e scalabile, capace di soddisfare tutti i requisiti definiti in precedenza. In particolare, i tre elementi coinvolti sono:

1. *Frontend (lato client)*: il frontend rappresenta la parte dell'applicazione che interagisce direttamente con l'utente finale. Utilizzando tecnologie come HTML, CSS e JavaScript, il frontend è responsabile della presentazione dei dati e della gestione delle interazioni utente. La sua progettazione deve garantire un'esperienza utente fluida e intuitiva, nonché la validazione dei dati prima che questi vengano inviati al backend.

2. *Backend (lato server)*: il backend è il cuore pulsante dell'applicazione, dove risiede la logica di business. Il server riceve le richieste dal frontend, elabora i dati, esegue le operazioni necessarie e restituisce le risposte appropriate. Inoltre, durante i vari processi, comunica con il database per memorizzare, recuperare e aggiornare i dati persistenti dell'applicazione. Per lo sviluppo di questa componente, verrà utilizzato un framework

server-side che faciliterà l'interazione con il database e garantirà una gestione efficiente delle operazioni richieste.

3. *Database*: il database è la componente responsabile della gestione dei dati persistenti, utilizzata per memorizzare e recuperare le informazioni necessarie al funzionamento dell'applicazione. Esso deve garantire l'integrità e la sicurezza dei dati, come discusso nel Capitolo 3.

Questa tripartizione dell'architettura favorisce una chiara separazione delle responsabilità. In Figura 4.1, viene rappresentato un diagramma esplicativo dell'architettura sopra descritta.

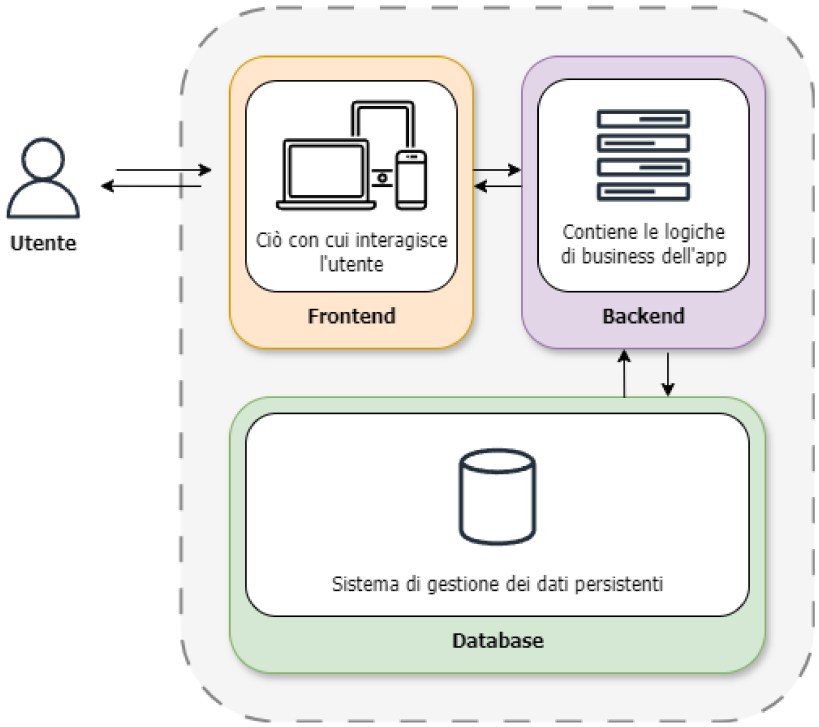

Figura 4.1: Diagramma dell'architettura dell'applicazione web

4.1.1 Architettura MVC (Model-View-Controller)

Nella progettazione di applicazioni web odierne, la struttura appena descritta viene comunemente implementata tramite l'architettura Model-View-Controller (MVC). Questo pattern architetturale rappresenta un approccio consolidato per la suddivisione delle responsabilità all'interno di un'applicazione web. Esso si basa sulla separazione chiara e distinta dei tre principali componenti: il Modello (Model), la Vista (View) e il Controllore (Controller).

Questa separazione delle responsabilità non solo promuove la modularità e la manutenibilità del software, ma facilita anche la scalabilità dell'applicazione, consentendo una gestione efficiente delle complessità che possono emergere durante lo sviluppo e l'evoluzione del sistema. Vediamo più nel dettaglio i tre componenti:

1. *Modello (Model)*: il Model rappresenta la struttura dei dati dell'applicazione, definendo come questi ultimi sono organizzati e possono essere acceduti. Nel Model è anche inclusa la logica di business che definisce come i dati sono manipolabili e quali regole devono essere rispettate durante queste manipolazioni.

2. *Vista (View)*: la View si occupa della presentazione dei dati agli utenti e interagisce con loro attraverso un'interfaccia utente. Questa componente non contiene logica di business, ma si limita a mostrare i dati e a catturare le azioni degli utenti.

3. *Controllore (Controller)*: il Controller funge da intermediario tra il Model e la View, gestendo il flusso delle richieste e delle risposte. Riceve le richieste dall'utente attraverso la View, interpreta queste richieste e decide quale Model utilizzare (per ricavare i dati) e quale View restituire (per mostrare quei dati).

In Figura 4.2 è raffigurato uno schema che sintetizza quanto detto.

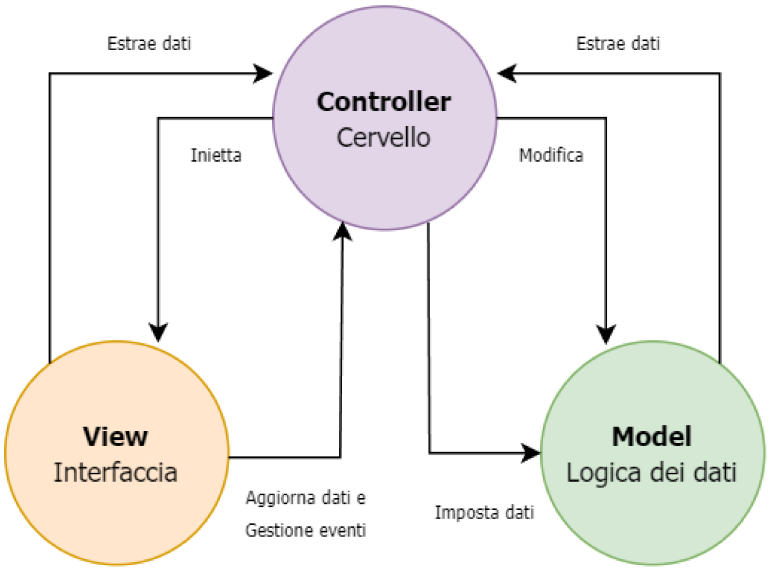

Figura 4.2: Schema dell'architettura MVC

Come detto in precedenza, non sarà necessario predisporre manualmente l'applicazione per seguire i principi di questo pattern, in quanto verrà utilizzato un framework che definirà in maniera ottimale la struttura del progetto.

4.1.2 Framework

Il framework appena citato è *Laravel*, un potente framework PHP open-source progettato per agevolare lo sviluppo di applicazioni web con architettura MVC (Model-View-Controller). Laravel mette a disposizione una vasta gamma di strumenti e funzionalità integrate, fondamentali per coprire molte delle attività comuni nello sviluppo web, come l'autenticazione, la gestione delle sessioni e la memorizzazione nella cache. Nel contesto della nostra applicazione di gestione dei rifiuti, l'utilizzo di Laravel risulta particolarmente vantaggioso, poiché necessitiamo di solide funzionalità di autenticazione per garantire l'accesso sicuro e personalizzato agli utenti, nonché di efficienti meccanismi di gestione delle sessioni, per mantenere un'interazione fluida e coerente con il sistema.

Vedremo come l'utilizzo di questo framework apporta benefici significativi in ciascuno dei tre elementi principali che compongono l'architettura del sistema esaminato. In particolare, esploreremo come Laravel faciliti la gestione e l'integrazione del frontend e del backend, fornendo una base solida e modulare che consente uno sviluppo rapido ed efficiente.

Con questa comprensione dettagliata della struttura generale della nostra applicazione web, possiamo ora approfondire le specifiche del *frontend* e del *backend*, delineando chiaramente le tecnologie e le metodologie che verranno impiegate.

4.2 Progettazione del frontend

Il frontend rappresenta la parte visibile e interattiva della nostra applicazione web, quella con cui gli utenti interagiscono direttamente. È una componente fondamentale che determina l'esperienza utente (UX) attraverso l'interfaccia utente (UI). La progettazione del frontend non riguarda solo l'estetica, ma anche la funzionalità, l'usabilità e l'accessibilità dell'applicazione. Un altro aspetto rilevante relativo al frontend è la validazione dei dati inseriti dall'utente nei vari form presenti, garantendo che le informazioni siano corrette e sicure prima di essere inviate al server.

La parte interessata dell'architettura MVC (Model-View-Controller) nel frontend si rispecchia principalmente sulla View. Vediamo, ora, come è possibile creare questo tipo di interfacce, sfruttando i benefici del pattern e, quindi, del framework Laravel.

4.2.1 Tecnologie utilizzate

Per la realizzazione del frontend dell'applicazione verranno utilizzate le seguenti tecnologie principali:

- *HTML (HyperText Markup Language)*: è il linguaggio standard per creare le pagine web. HTML fornisce la struttura del contenuto attraverso l'uso di tag e attributi, permettendo di definire elementi come paragrafi, titoli, link, immagini e form.
- *CSS (Cascading Style Sheets)*: è utilizzato per definire lo stile e il layout delle pagine web, controllando aspetti come i colori, i font, il posizionamento degli elementi e le animazioni.
- *JavaScript*: è un linguaggio di scripting che consente di creare contenuti web dinamici e interattivi; esso viene largamente utilizzato nelle applicazioni web odierne.

Questi linguaggi costituiscono la base della stragrande maggioranza delle applicazioni web. Vediamo ora i framework e le librerie che ci consentono di sfruttarli al massimo, in relazione al tipo di applicazione che vogliamo realizzare.

- *Blade di Laravel*: anche se Laravel è principalmente associato al backend, Blade è un motore di template che viene utilizzato per la generazione delle viste nel frontend. L'utilizzo di Blade permette di includere componenti riutilizzabili, come header, footer e sidebar, e di gestire i dati dinamici attraverso direttive semplici ed intuitive.
- *Tailwind CSS*: Tailwind è un framework CSS utility-first progettato per semplificare e velocizzare lo sviluppo del frontend, con la particolarità di fornire una serie di classi utility che permettono di applicare stili direttamente nei tag HTML. Questo approccio aiuta a ridurre il peso complessivo dei file CSS, migliora le performance dell'applicazione e velocizza l'intero processo di realizzazione delle viste.
- *jQuery*: jQuery è una libreria che semplifica la manipolazione del DOM, la gestione degli eventi e l'esecuzione delle chiamate AJAX, permettendo di creare esperienze utente fluide e reattive.

Definite le tecnologie che verranno utilizzate per la realizzazione del sito web, possiamo ora esaminare alcuni mockup relativi alle varie viste che comporranno la nostra applicazione finale.

4.2.2 Regola 60-30-10

Prima di proseguire con i mockup del sito, è opportuno condividere una regola di design che si rivela estremamente efficace nella creazione di interfacce intuitive e visivamente gradevoli.

La regola 60-30-10 è un principio di design utilizzato per creare interfacce minimali e intuitive, basandosi sull'uso di una palette di tre colori distribuiti in proporzioni specifiche. In questo approccio ogni colore ha un ruolo specifico; ciò permette di mantenere un equilibrio visivo e guidare l'attenzione dell'utente in modo naturale.

Vediamo le percentuali di utilizzo di ciascun colore e le sue caratteristiche:

- *60%*: il colore che copre circa il 60% della pagina deve essere un colore neutro e di base, fungendo da sfondo per il contenuto. Questo colore deve assicurare leggibilità e non deve disturbare visivamente l'utente.
- *30%*: il colore che copre circa il 30% della pagina dovrebbe avere più carattere e distinguersi chiaramente dal colore di base. Questo colore viene utilizzato per evidenziare le aree importanti della pagina, migliorando la distinzione visiva delle diverse sezioni.
- *10%*: il colore che copre circa il 10% della pagina è il punto forte e per questo motivo deve essere utilizzato con parsimonia. Questo colore ha il compito di attirare l'attenzione dell'utente e viene impiegato per elementi considerati importanti con cui egli molto spesso può interagire.

La palette di colori utilizzata, con le corrispettive percentuali, è mostrata in Figura 4.3.

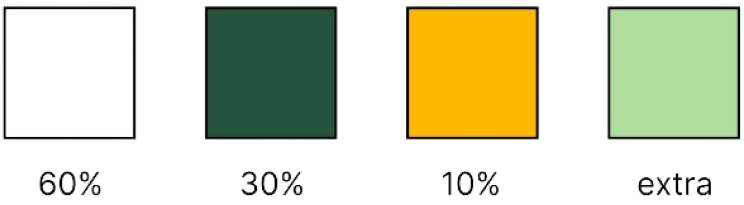

Figura 4.3: Palette di colori da noi utilizzata

Nella prossima sezione, verranno presentati i mockup del sito, mostrando come è stato applicato questo principio di design.

4.2.3 Mockup generali

I mockup sono rappresentazioni grafiche statiche dell'interfaccia utente di un'applicazione. Questi sono importanti nella fase di progettazione, poiché permettono di visualizzare e valutare l'aspetto finale dell'applicazione prima della fase di sviluppo. I mockup risultano strumenti molto utili per ottenere un feedback precoce e per garantire che il design dell'interfaccia utente sia allineato con le aspettative degli utenti.

L'intero processo di creazione dei mockup è stato guidato dall'obiettivo di trovare le soluzioni più semplici e intuitive per soddisfare tutti i requisiti funzionali dell'applicazione, come definiti nel Capitolo 2.

Analogamente a quanto fatto nel capitolo appena citato, anche i mockup sono stati organizzati in base alle diverse pagine visibili agli utenti con ruolo di *Amministratore* e a quelli con ruolo di *Operatore*. Questa suddivisione permette di affrontare le specifiche esigenze e

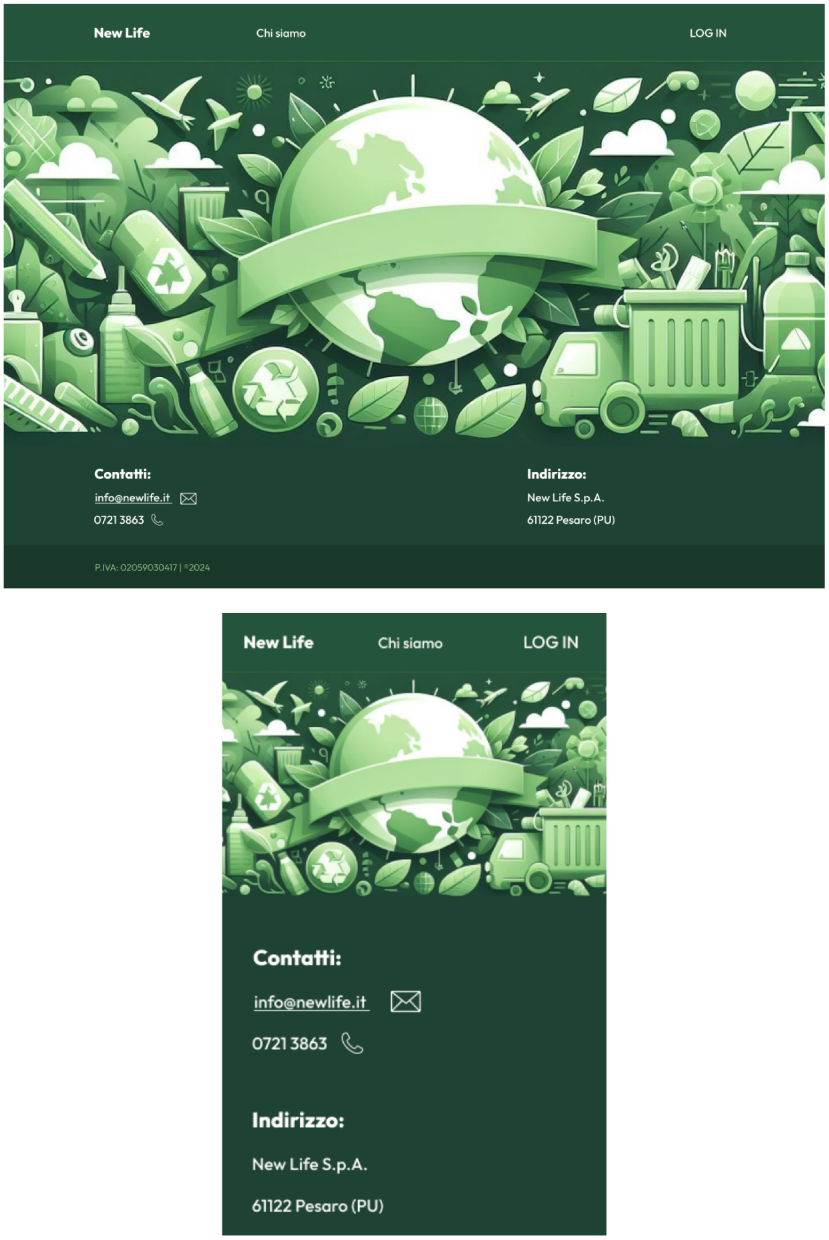

Figura 4.4: Mockup - Home page

funzionalità richieste dai diversi tipi di utenti, assicurando che ogni interfaccia sia ottimizzata per il proprio contesto d'uso.

Prima di analizzare le pagine specifiche per ciascun ruolo, presentiamo le viste comuni a tutti i tipi di utenti. Nelle Figure 4.4 e 4.5 è possibile vedere, rispettivamente, la homepage dell'applicazione e la sezione di login, che rappresenta il punto di accesso iniziale per gli utenti non ancora autenticati. Le figure mostrano sia le versioni desktop che quelle mobile delle pagine, per garantire una fruizione ottimale dell'applicazione anche da parte degli operatori che potrebbero accedervi da dispositivi mobili.

4.2.4 Mockup relativi alla sezione per l'amministratore

Effettuato il login con le apposite credenziali, verranno mostrati, ora, i mockup relativi alle pagine dedicate alle funzioni dell'amministratore.

Figura 4.5: Mockup - Login

In particolare, in Figura 4.6 vengono mostrati i seguenti mockup:

- *Prima schermata*: esso rappresenta la prima pagina mostrata dopo l'autenticazione come amministratore. In questa schermata viene visualizzata una panoramica di tutti i giri di raccolta attualmente attivi, con informazioni dettagliate, incluso lo stato attuale di ogni giro, evidenziato attraverso un banner colorato.
- *Seconda schermata*: esso è accessibile quando l'amministratore clicca sul pulsante "Gestisci". In questa sezione, l'utente ha un controllo maggiore e una visualizzazione più schematica di tutti i giri di raccolta, inclusi quelli non ancora attivati (indicati con uno sfondo grigio). La fruizione è semplificata grazie alla possibilità di filtrare i vari giri secondo diversi criteri. Inoltre, questa sezione consente di passare alla fase di creazione di un nuovo giro di raccolta.

In Figura 4.7 vengono mostrati i seguenti mockup:

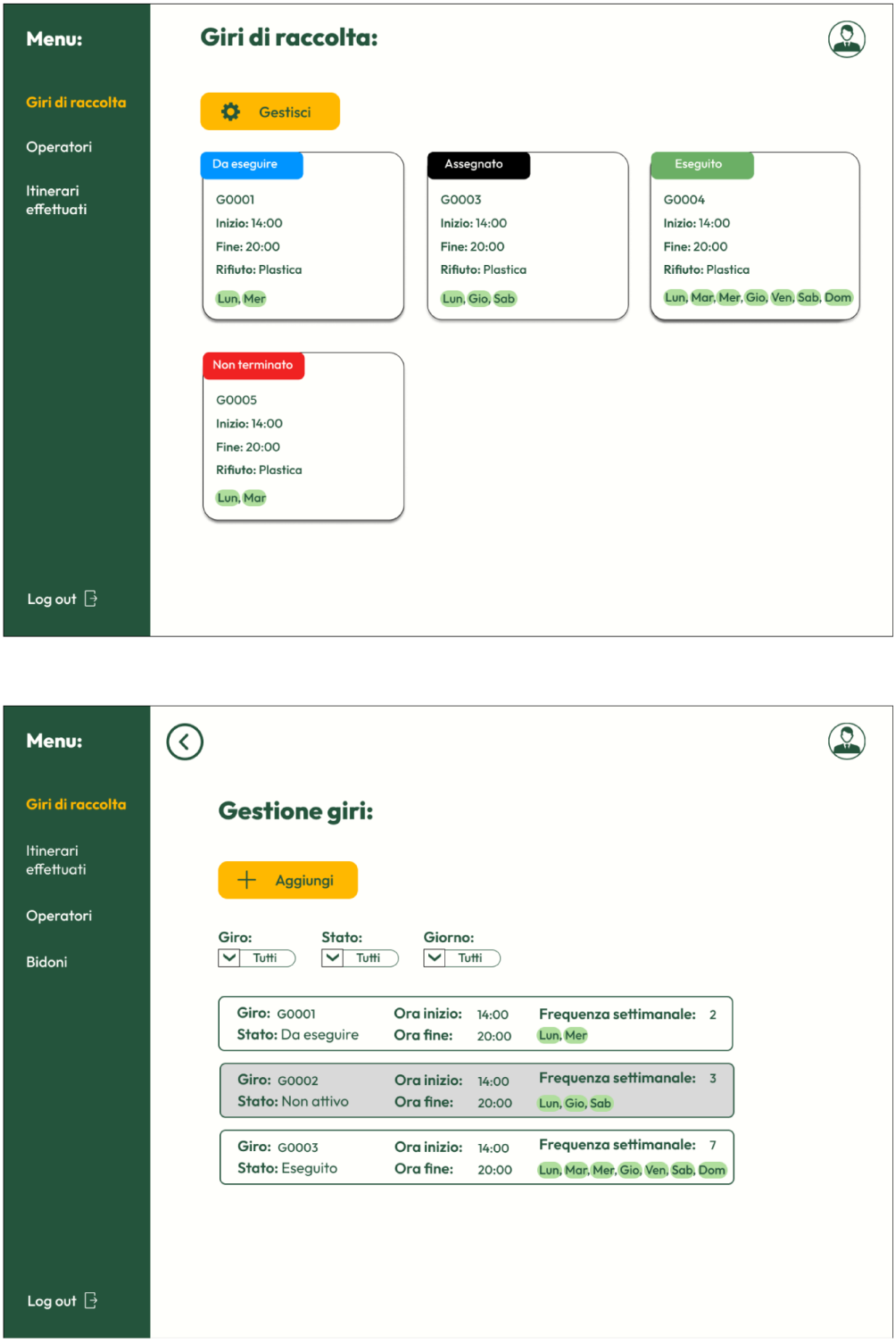

Figura 4.6: Mockup - Home e gestione giri di raccolta

- *Prima schermata*: esso mostra il prototipo della pagina visualizzata dopo aver interagito con il pulsante "Aggiungi", visto nella seconda schermata di Figura 4.6. Essa rappresenta una delle pagine chiave dell'intera applicazione web, poiché contiene il modulo per la creazione di un nuovo giro di raccolta. In questa form, è necessario inserire tutti i dati richiesti, inclusi i giorni della settimana in cui avverrà la raccolta; sarà, inoltre, presente una sezione dedicata alla definizione del percorso dei bidoni.

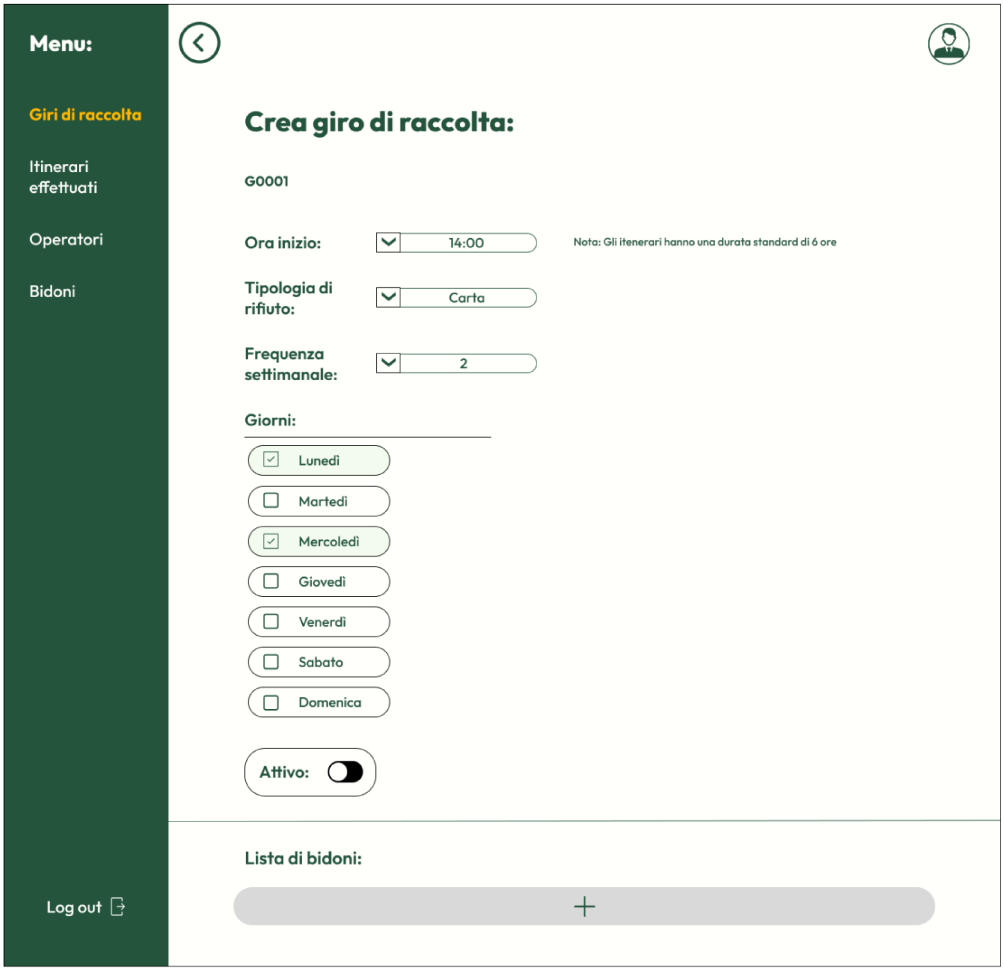

Figura 4.7: Mockup - Creazione giro di raccolta

- *Seconda schermata*: esso appare quando si clicca sul pulsante raffigurante un "+", situato sotto la voce "Lista di bidoni". Questo modale consente di selezionare, tramite una lista di check box, i bidoni che si desidera includere nel percorso del giro di raccolta.

In Figura 4.8 viene mostrata la pagina successiva alla compilazione di tutti i dati necessari per la creazione di un nuovo giro di raccolta, come illustrato nella Figura 4.7. In questa vista

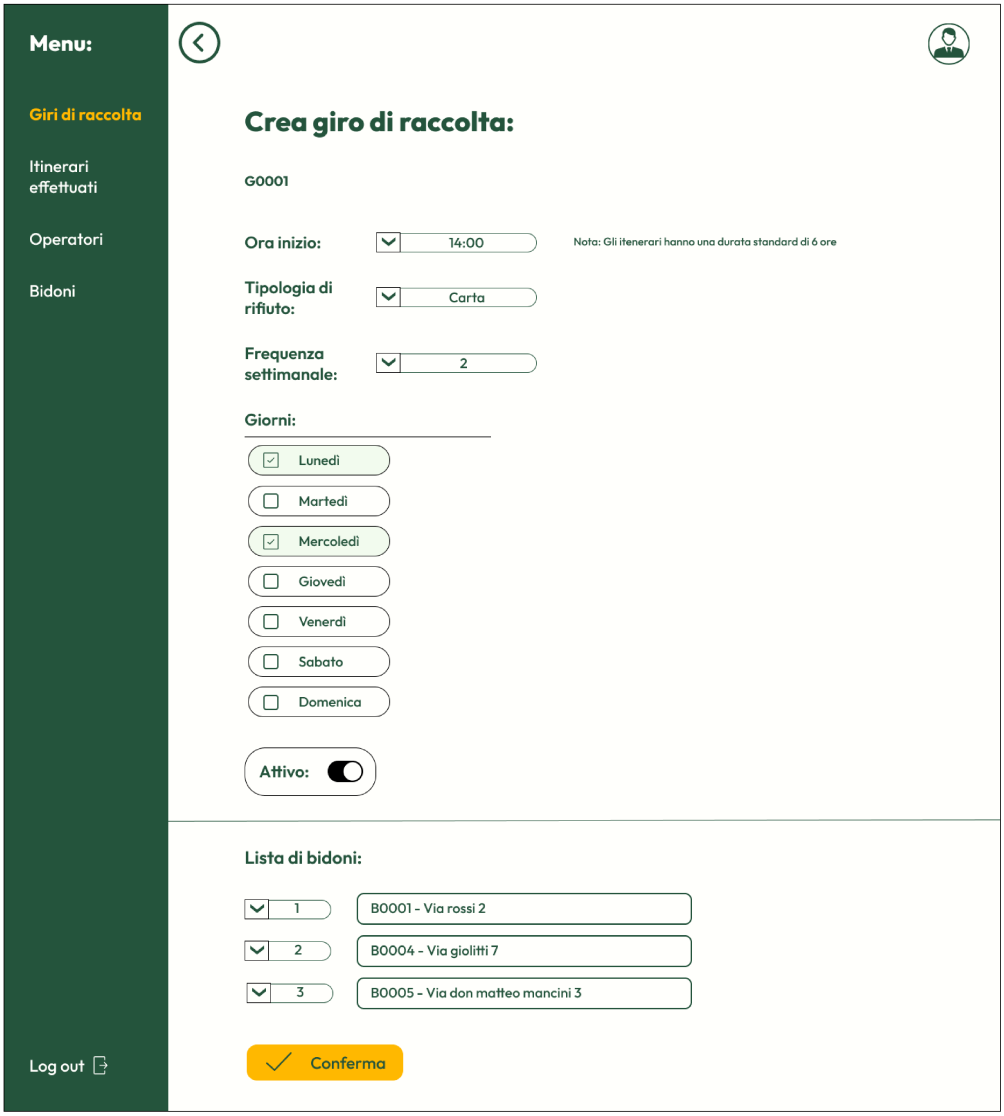

Figura 4.8: Mockup - Creazione giro di raccolta completata

compare la lista dei bidoni selezionati precedentemente nel modale, con una select accanto a ciascuno di essi per permettere all'amministratore di definire l'ordine di raccolta. Infine, è presente un pulsante di conferma per completare il processo di creazione del giro.

In Figura 4.9 vengono visualizzati i seguenti mockup:

- *Prima schermata*: interagendo con uno dei giri di raccolta dalle schermate mostrate nei mockup di Figura 4.6, si accede alla pagina relativa ai dettagli di quel giro. Un banner colorato comunica rapidamente all'utente lo stato del giro. Sono presenti i dati rilevanti e i pulsanti per modificare o eliminare il giro dal database. Nel caso in cui il giro sia nello stato "Da eseguire", è disponibile un pulsante specifico per assegnare il giro a un operatore disponibile.
- *Seconda schermata*: questa sezione appare cliccando sul pulsante "Assegna" accanto alla voce "Operatore". Il modale mostrato consente di selezionare un operatore tramite una lista di radio button, facilitando l'assegnazione del giro di raccolta.

In Figura 4.10 vengono visualizzati i seguenti mockup:

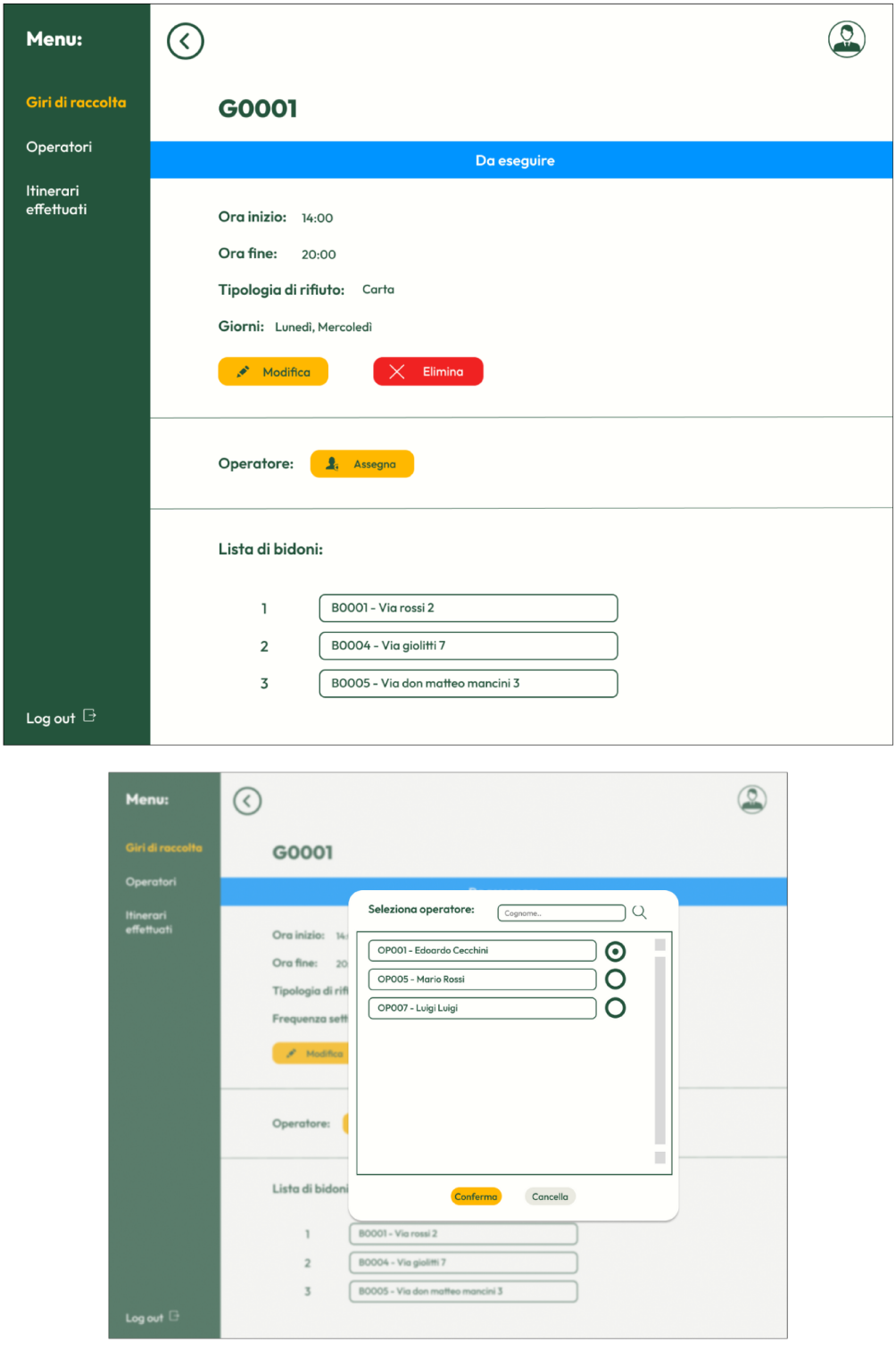

Figura 4.9: Mockup - Dettaglio di un giro e sua assegnazione a un operatore

- *Prima schermata*: questa schermata rappresenta la stessa pagina di Figura 4.9, ma con il giro di raccolta già assegnato a un operatore. È possibile notare il cambiamento di colore e di label del banner, la disabilitazione dei pulsanti per la modifica e l'eliminazione, nonché l'informazione relativa all'operatore assegnato. È presente, anche, il pulsante "Dissocia", che consente di dissociare l'operatore e di riportare il giro allo stato "Da eseguire".

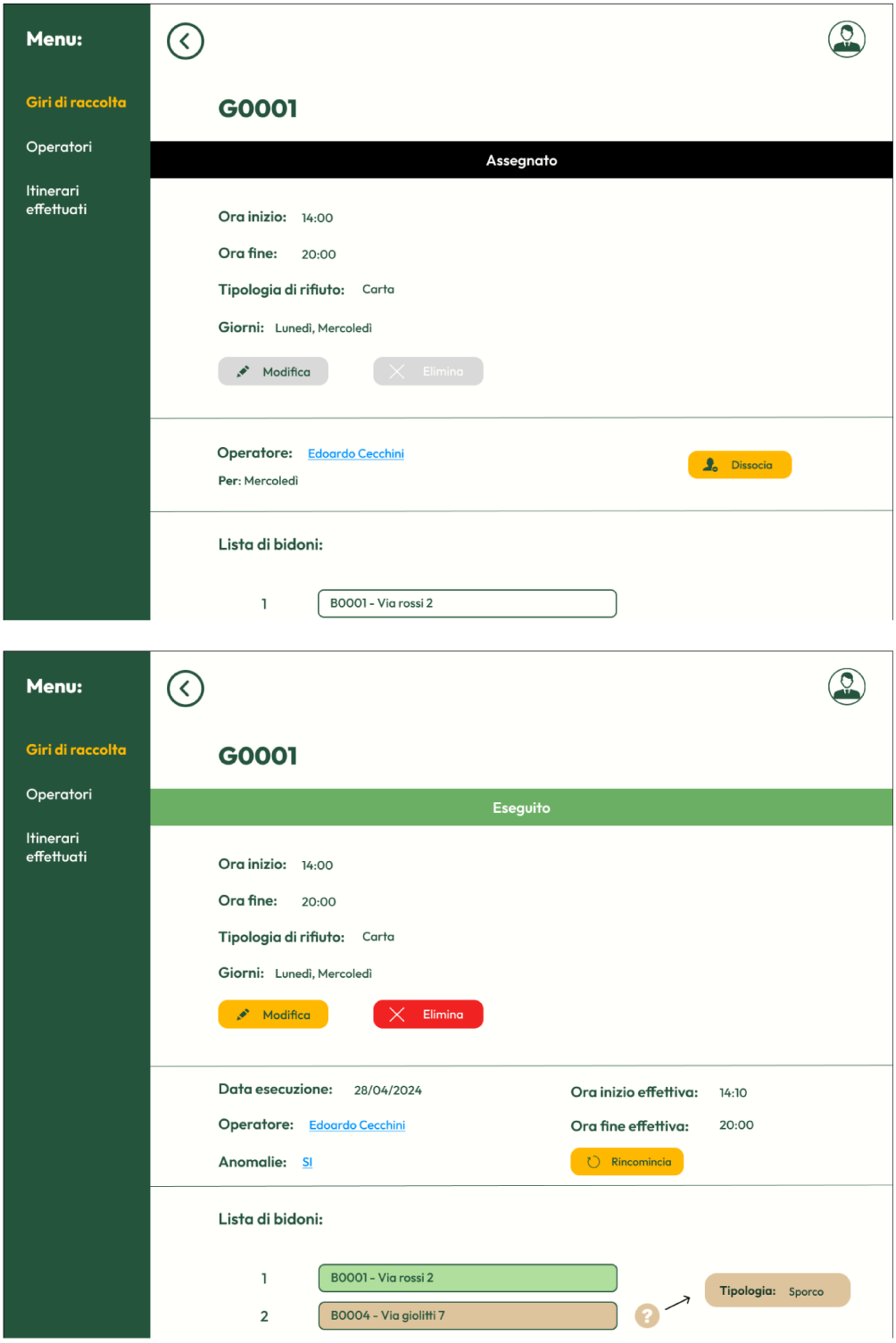

Figura 4.10: Mockup - Dettaglio relativo a un giro assegnato ed eseguito

- *Seconda schermata*: questa vista mostra una variante del dettaglio del giro di raccolta, rispecchiando un giro già completato e per il quale non è ancora scattato il giorno in cui deve essere svolto nuovamente. Oltre alle informazioni sull'itinerario effettuato, è presente un pulsante "Ricomincia", che permette di forzare il cambio di stato del giro passando da "Eseguito" a "Da eseguire". I bidoni con anomalie sono evidenziati per attirare l'attenzione dell'utente.

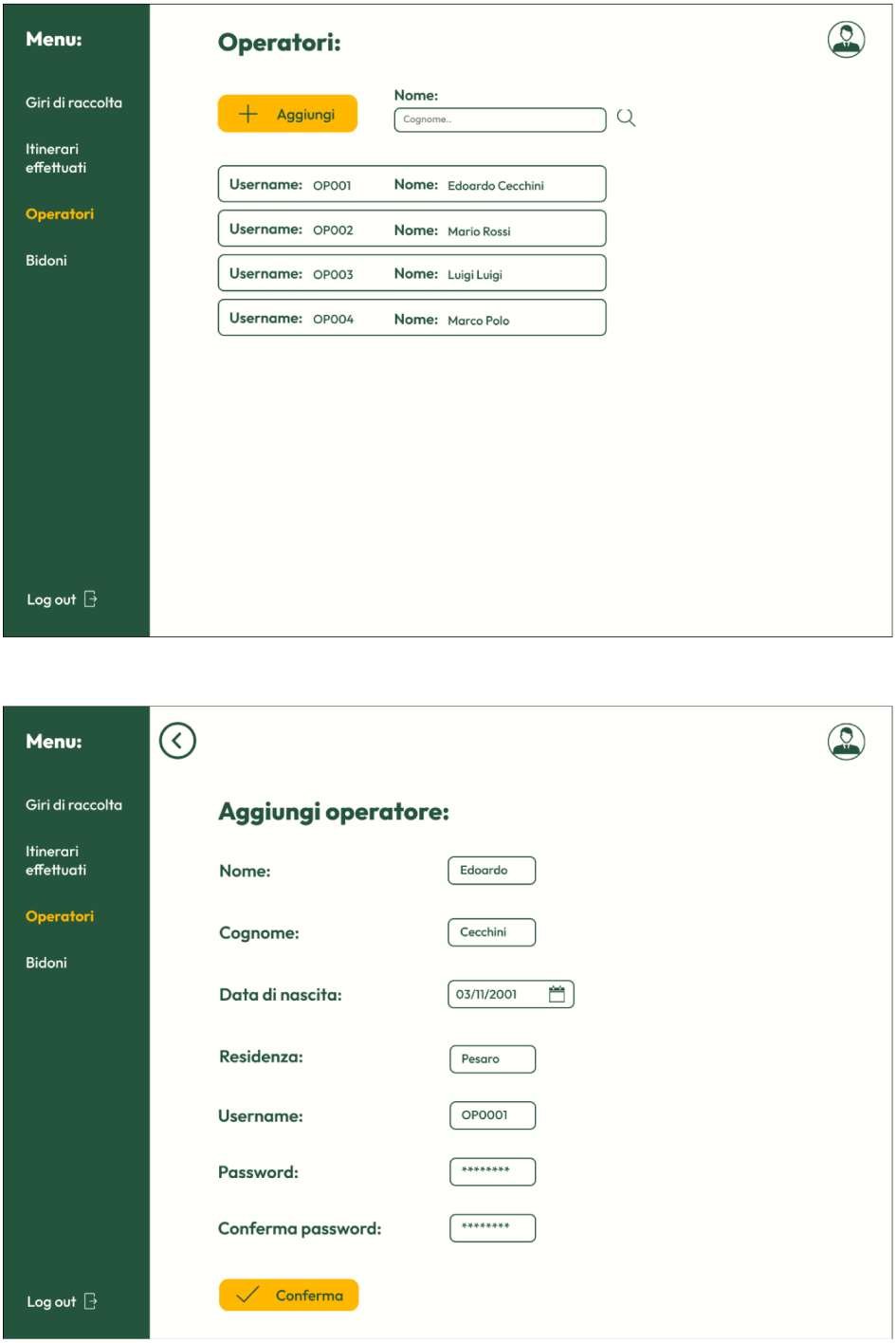

Figura 4.11: Mockup - Lista degli operatori e aggiunta di un nuovo operatore

In Figura 4.11 vengono illustrati i seguenti mockup:

- *Prima schermata*: questo mockup rappresenta la pagina che si raggiunge una volta cliccato sulla sezione "Operatori" dal menù dell'amministratore. Qui viene visualizzata una lista di tutti gli operatori presenti nel database, con alcune informazioni su di essi. Inoltre, in esso, è presente un input di testo utile per effettuare una ricerca per nome e cognome.
- *Seconda schermata*: questa vista mostra il modulo per l'aggiunta di un nuovo operatore

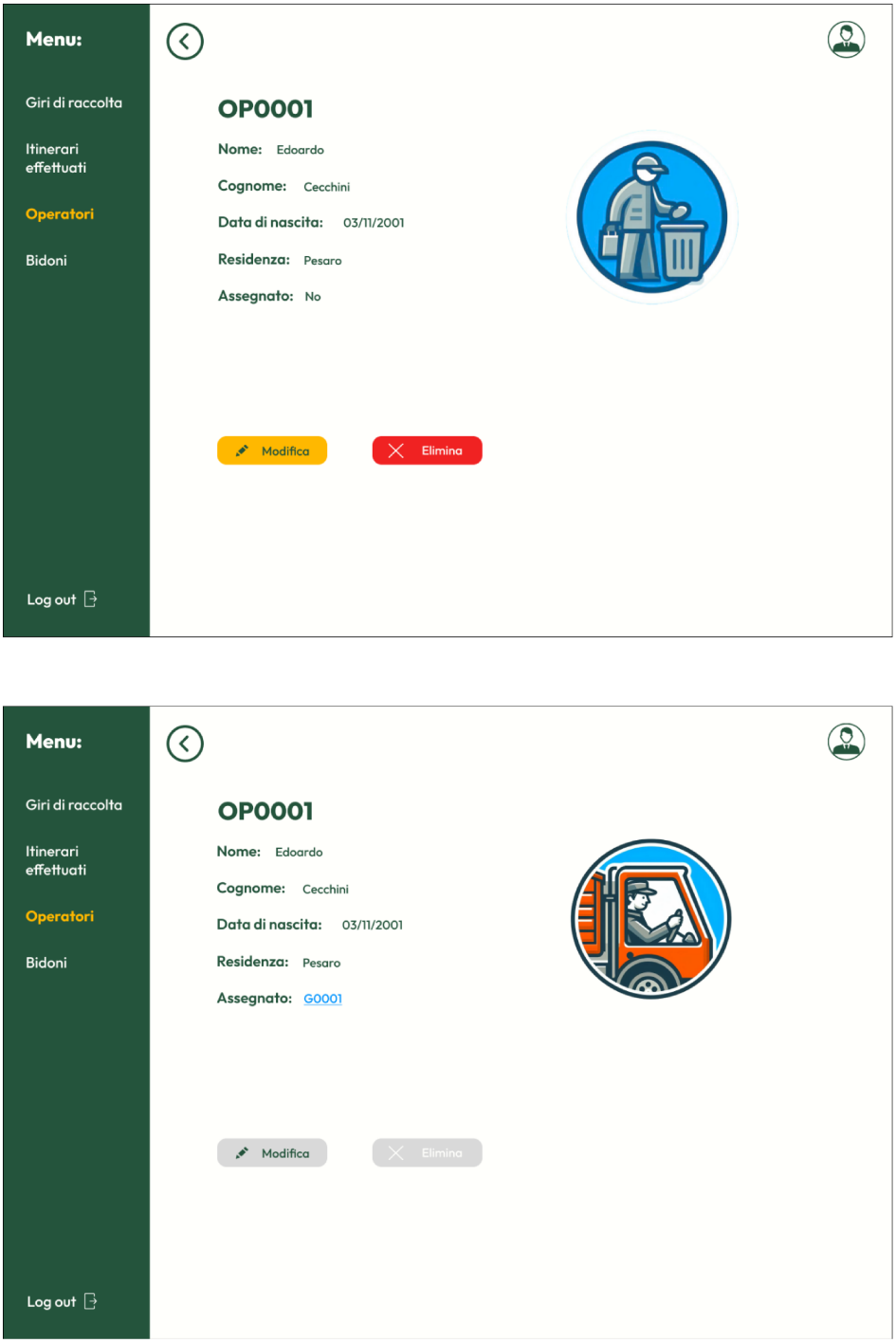

Figura 4.12: Mockup - Dettaglio di un operatore, non assegnato e assegnato

nel sistema. In questa form, oltre ai dati anagrafici, è necessario inserire anche un username e una password, che costituiranno le credenziali di accesso per l'Operatore all'interno dell'applicazione web.

In Figura 4.12 vengono mostrati i seguenti mockup:

- *Prima schermata*: questa schermata mostra il dettaglio di un operatore, accessibile cliccando su uno di essi nella sezione "Operatori" (in Figura 4.11). La schermata indica un

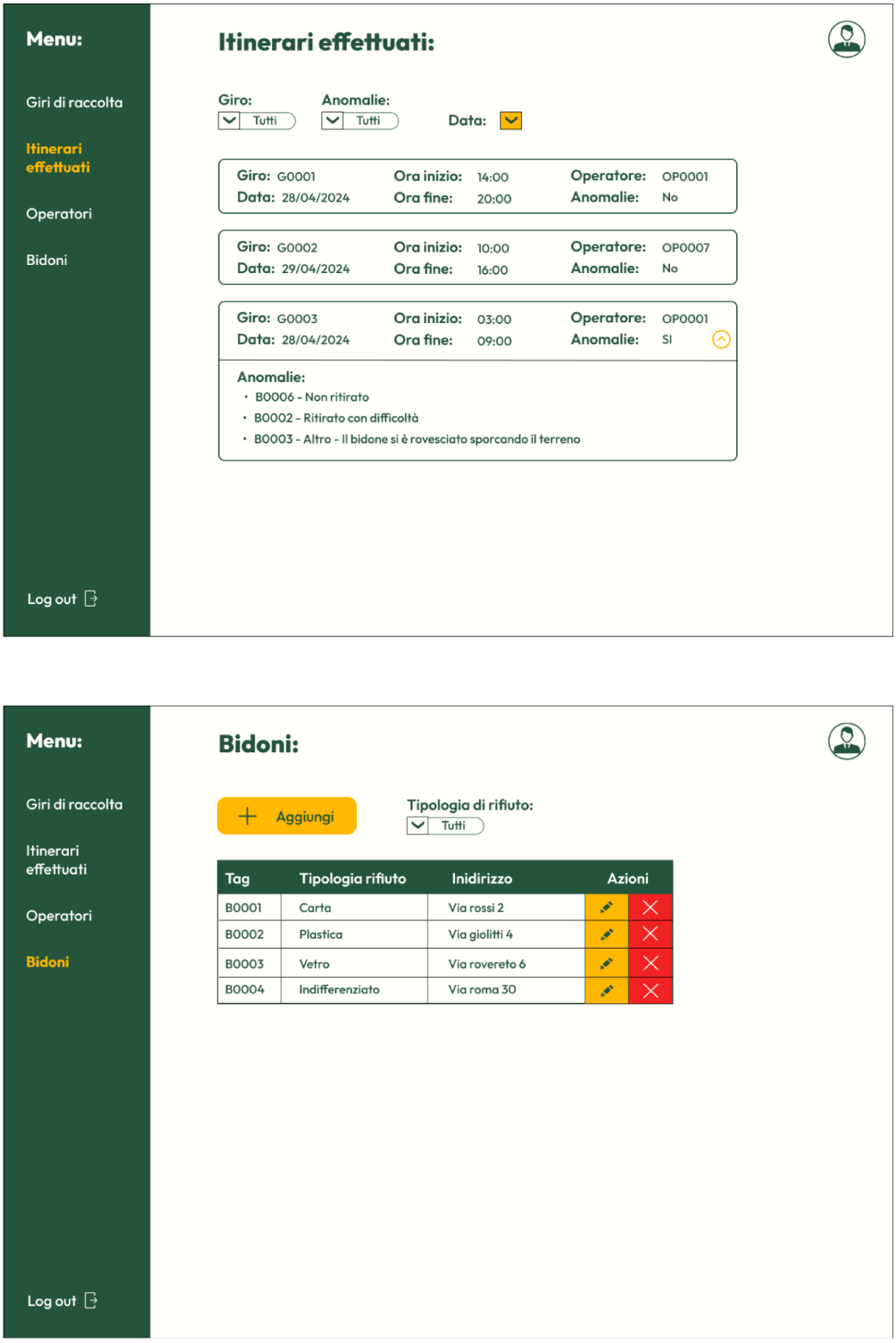

Figura 4.13: Mockup - Sezioni itinerari effettuati e bidoni

operatore non assegnato; ciò è chiaramente indicato sia da una dicitura esplicita, sia dall'immagine presente, che varia in base a questo dato.

- *Seconda schermata*: questa vista rappresenta lo stesso dettaglio di un operatore descritto in precedenza, ma per un operatore attualmente impegnato in un giro di raccolta. Viene mostrato il giro di raccolta assegnato, con un cambiamento nell'immagine associata.

In Figura 4.13 vengono mostrati i seguenti mockup:

- *Prima schermata*: questa schermata mostra lo storico di tutti gli itinerari effettuati, con informazioni dettagliate e opzioni di filtro e ordinamento. Gli itinerari con anomalie presentano un pulsante per espandere la scheda e mostrare i dettagli delle anomalie e i bidoni interessati.
- *Seconda schermata*: questa schermata mostra la sezione "Bidoni", accessibile dal menù. Viene visualizzata una tabella con tutti i bidoni presenti nel database, con opzioni per filtrarli, modificarli o eliminarli. È presente, anche, un pulsante "Aggiungi" per la creazione di nuovi bidoni.

Per motivi di spazio, non è stata mostrata la totalità dei mockup (lato amministratore) realizzati. Tuttavia, quanto esposto finora consente di ottenere una visione concreta di come verranno implementati i vari requisiti per gli utenti con ruolo di amministratore. Procediamo ora con i mockup progettati per la sezione relativa al supporto delle attività che devono svolgere gli operatori.

4.2.5 Mockup relativi alla sezione per l'operatore

Il punto focale della sezione dedicata agli operatori è l'ottimizzazione dell'applicativo web per dispositivi mobili, utilizzati dagli operatori durante le loro attività quotidiane.

Accedendo con le credenziali specifiche fornite dall'amministrazione al momento dell'assunzione, gli operatori avranno a disposizione un'interfaccia progettata per supportare efficacemente lo svolgimento dei giri di raccolta. I mockup presentati di seguito sono mostrati nella loro versione mobile, anche se sono accessibili e utilizzabili da qualsiasi dispositivo.

In Figura 4.14 vengono mostrati i seguenti mockup:

- *Prima schermata*: questa schermata rappresenta la pagina iniziale visualizzata dopo l'autenticazione come operatore. In essa viene presentata una panoramica di tutti i giri di raccolta assegnati all'operatore. Oltre a fornire informazioni essenziali per l'esecuzione dei giri, la pagina include un pulsante per dichiarare rapidamente l'inizio del giro di raccolta.
- *Seconda schermata*: in questa schermata viene, invece, mostrato il menù laterale, accessibile cliccando sull'icona con le tre linee (in alto a sinistra). A causa delle dimensioni ridotte degli schermi mobili, è stato scelto un menù a comparsa che permette di accedere facilmente alle diverse aree utili, senza intaccare la visibilità degli elementi nella pagina.

In Figura 4.15 vengono mostrati i seguenti mockup:

- *Prima schermata*: selezionando uno dei giri di raccolta dalla schermata precedente, si accede alla pagina con i dettagli necessari per lo svolgimento del giro. Qui è presente un pulsante per dichiarare l'inizio del giro, una mappa che mostra la posizione dei bidoni lungo il percorso, e una lista a comparsa con gli indirizzi dei bidoni, utile in caso di problemi di geolocalizzazione (qui la lista risulta nascosta).
- *Seconda schermata*: questa vista mostra una variante della schermata precedente. Una volta iniziato il giro di raccolta, l'ora di inizio viene registrata e il pulsante si trasforma in un'opzione per dichiarare la fine del giro. La lista a comparsa degli indirizzi dei bidoni è visibile in questa schermata. Inoltre, è presente un pulsante, nell'angolo in basso a destra, che permette di tornare rapidamente in cima alla pagina.

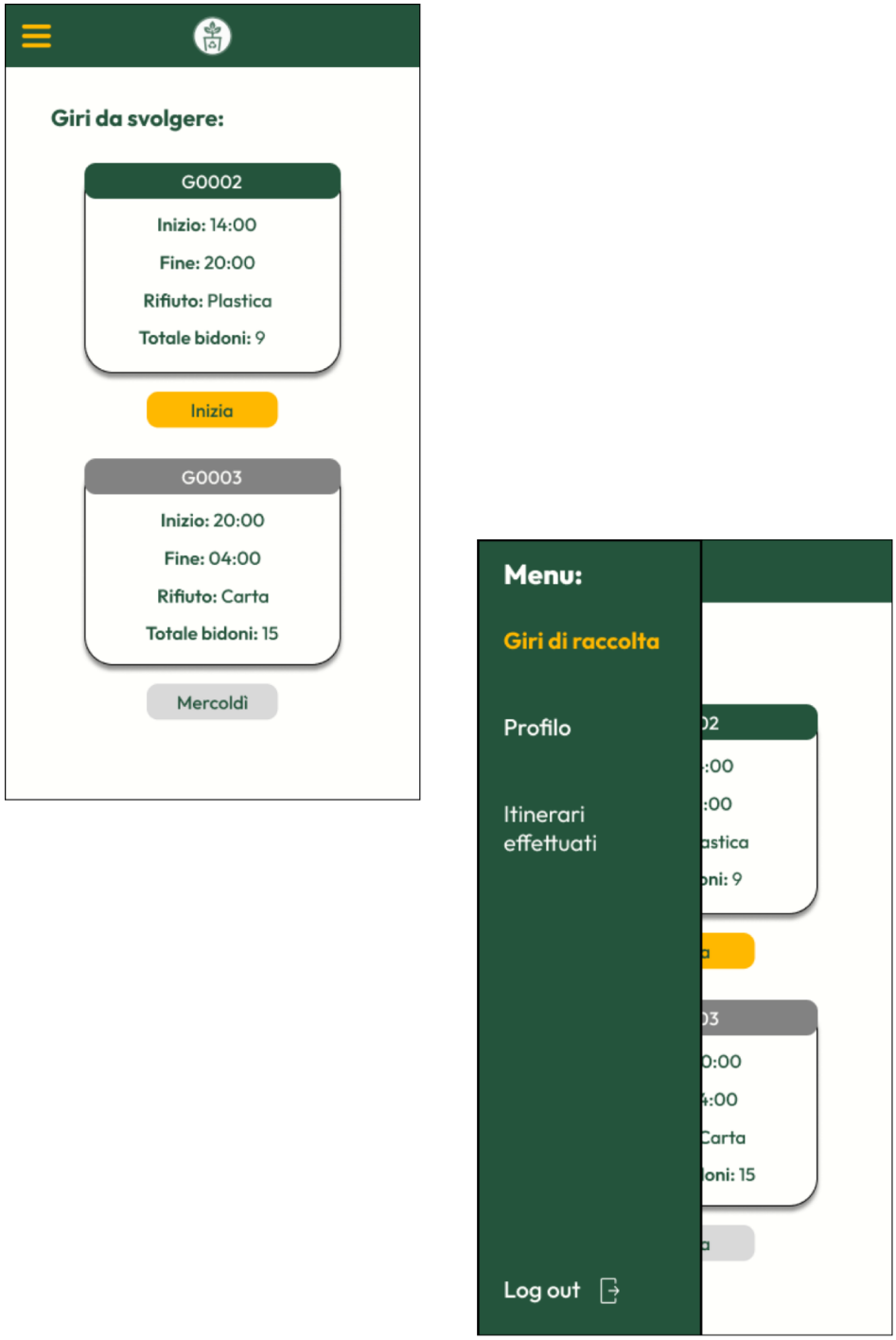

Figura 4.14: Mockup - Home dei giri di raccolta assegnati all'operatore e menù

In Figura 4.16 vengono mostrati i seguenti mockup:

- *Prima schermata*: cliccando sul pulsante "Termina" nella schermata precedente (Figura 4.15), si accede alla pagina di resoconto dell'itinerario appena completato. L'operatore può confermare l'itinerario oppure segnalare eventuali anomalie cliccando su un apposito switch.
- *Seconda schermata*: questa vista mostra la stessa pagina di resoconto, ma con la sezione aggiuntiva per la dichiarazione delle anomalie. In questa sezione vengono elencati tutti

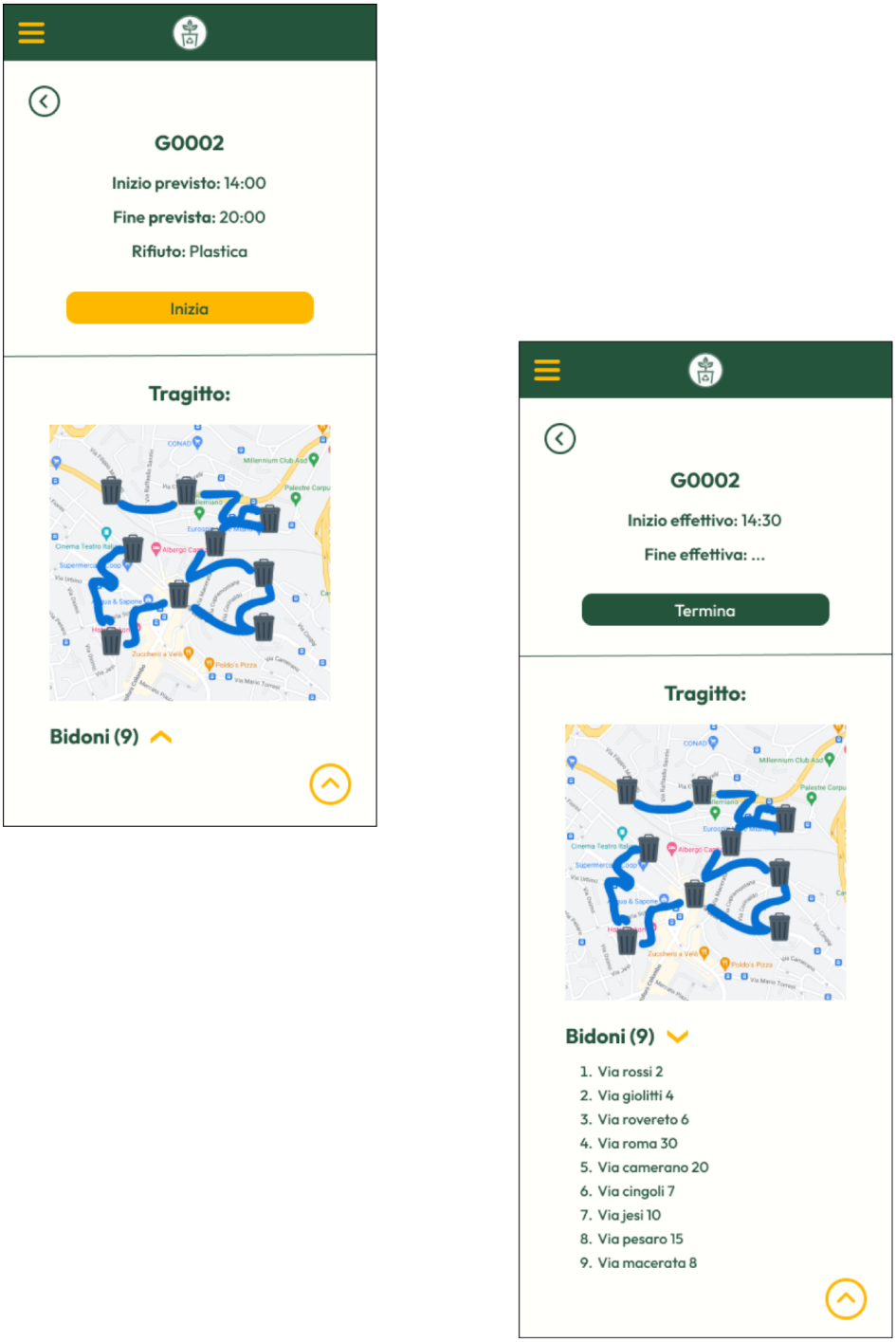

Figura 4.15: Mockup - Dettaglio di un giro non iniziato e iniziato

i bidoni raccolti durante l'itinerario, permettendo all'operatore di specificare eventuali problemi riscontrati.

In Figura 4.17 vengono mostrati i seguenti mockup:

- *Prima schermata*: questa schermata rappresenta il modale che si apre quando l'utente clicca su uno dei bidoni nella lista delle anomalie. Qui l'operatore può selezionare il tipo di anomalia da un menù a tendina e aggiungere una breve descrizione, se viene scelta l'opzione "Altro".

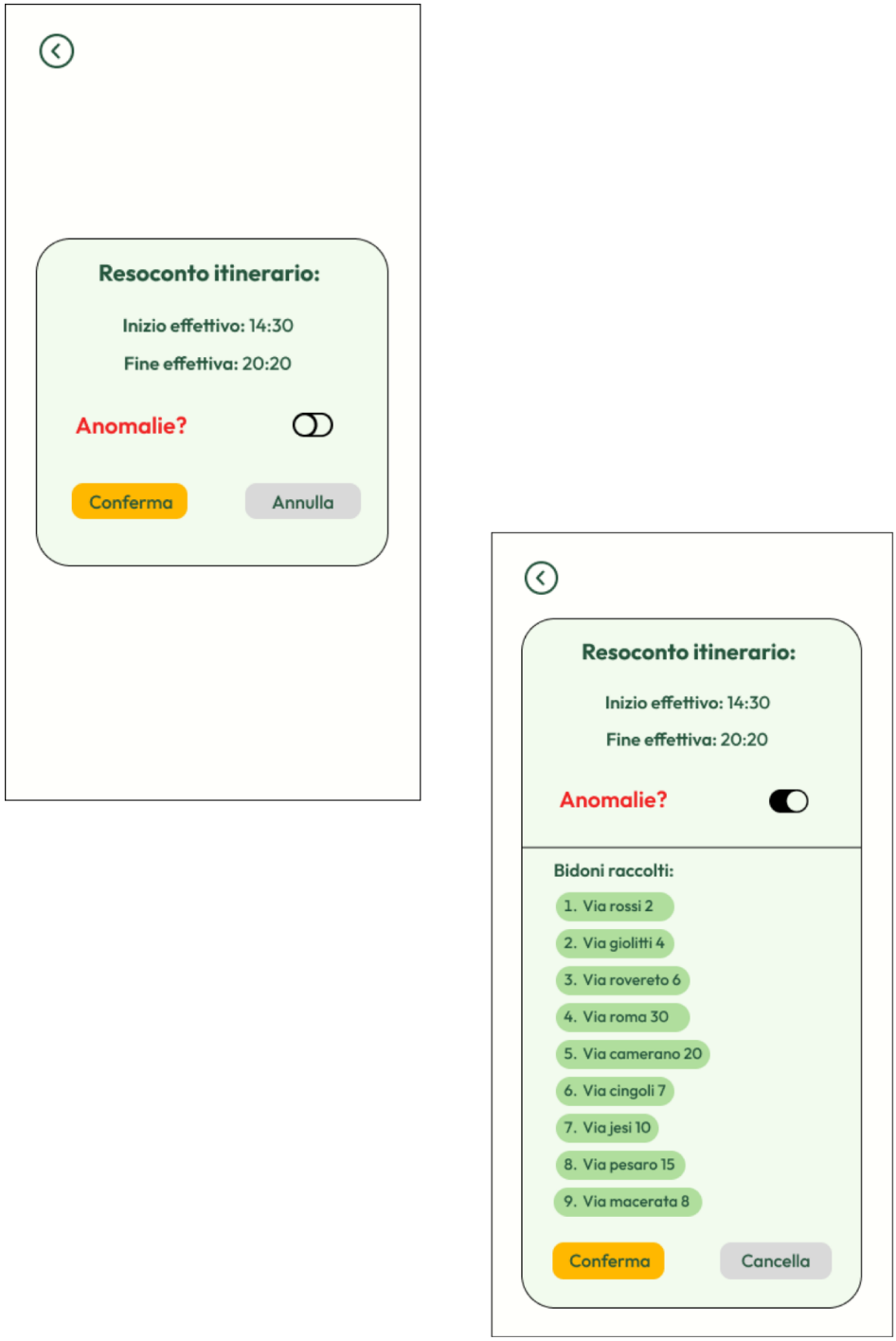

Figura 4.16: Mockup - Resoconto relativo all'itinerario effettuato

- *Seconda schermata*: questa vista mostra la pagina di resoconto con una chiara indicazione visiva dei bidoni per i quali sono state segnalate anomalie. All'interno della pagina sono presenti due pulsanti, uno per confermare l'operazione salvando l'itinerario nel database e un altro per annullare l'operazione.

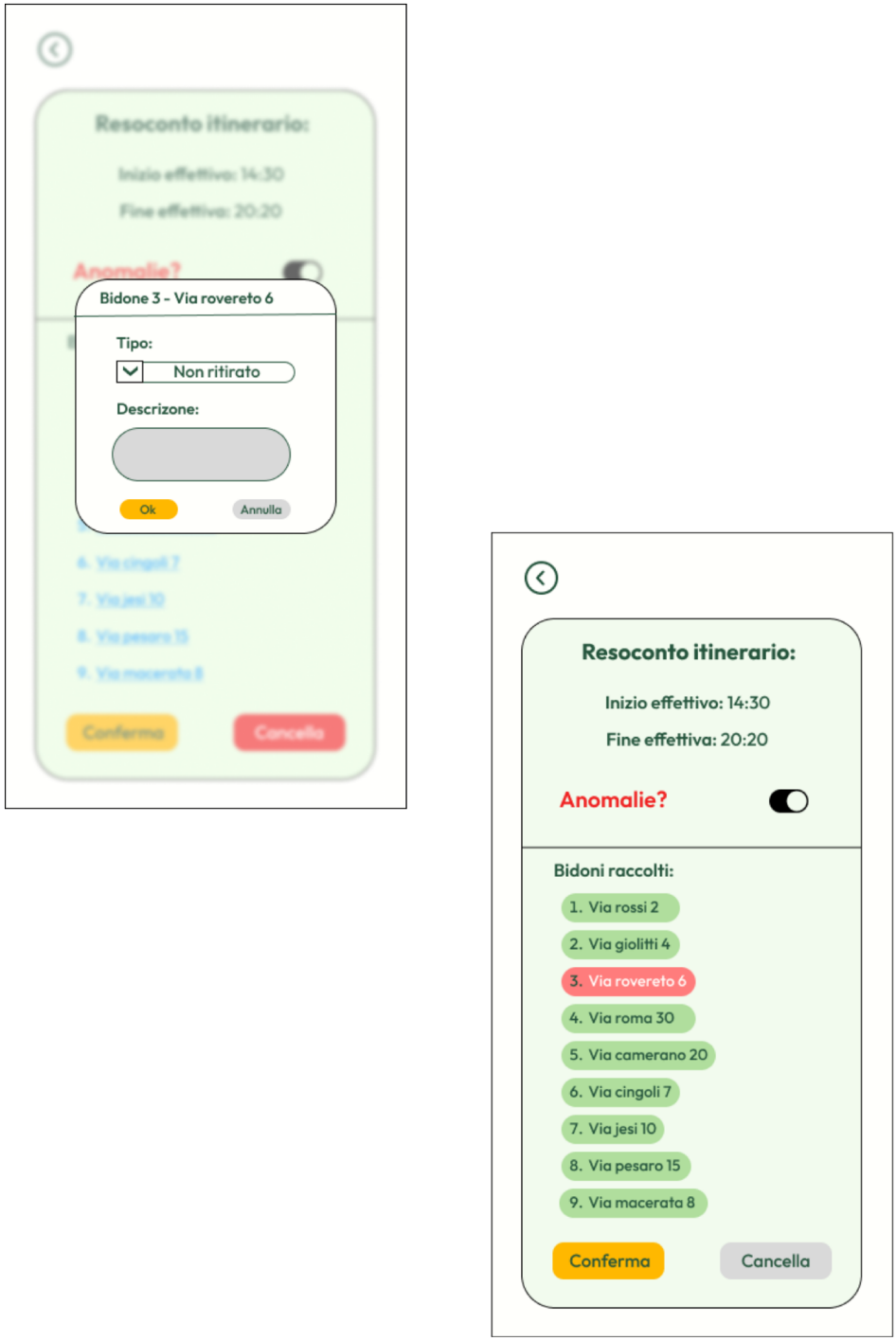

Figura 4.17: Mockup - Definizione di un'anomalia su un bidone

4.3 Progettazione del backend

Il backend costituisce la parte dell'applicazione web che gestisce la logica di business, l'accesso ai dati e le operazioni server-side. È una componente fondamentale che supporta e alimenta il frontend, assicurando che tutte le operazioni e le richieste degli utenti siano gestite correttamente e in modo efficiente.

Nel contesto dell'architettura MVC (Model-View-Controller), il backend è principalmente responsabile della gestione dei Model e dei Controller. I Model, come abbiamo già visto,

rappresentano la struttura dei dati e le interazioni con il database, mentre i Controller gestiscono la logica applicativa, orchestrando le richieste tra il frontend e i Model.

4.3.1 Tecnologie utilizzate

Per la realizzazione del backend dell'applicazione, verranno utilizzate le seguenti tecnologie, molte delle quali già impostate da Laravel:

- *PHP*: linguaggio di scripting server-side utilizzato per lo sviluppo web. Laravel, essendo un framework PHP, sfrutta le potenzialità di questo linguaggio per la gestione della logica di business e per l'interazione con il database.
- *Funzionalità integrate di Laravel*: Laravel mette a disposizione un vasto numero di funzionalità integrate, tra cui:
  - *Routing*: sistema che mappa gli URL richiesti dagli utenti alle corrispondenti funzioni del controller. Questo permette di definire in modo chiaro e conciso come rispondere alle diverse richieste HTTP (GET, POST, PUT, DELETE).
  - *Gestione delle sessioni*: sistema che permette di mantenere lo stato dell'utente tra diverse richieste. Laravel gestisce le sessioni in modo sicuro, permettendo di memorizzare dati di sessione in vari backend, come file, cookie, database etc.
  - *Autenticazione e autorizzazione*: sistema integrato per gestire l'accesso degli utenti alle varie parti dell'applicazione. Laravel fornisce strumenti per l'autenticazione (login, registrazione, reset password) e per l'autorizzazione, controllando chi ha il permesso di eseguire determinate azioni.
- *Artisan*: interfaccia a riga di comando inclusa in Laravel che fornisce una serie di comandi utili per svolgere varie attività di sviluppo, come la creazione di controller, model, migration e seeding del database.
- *MySQL*: sistema di gestione di database relazionale (RDBMS) utilizzato per memorizzare e gestire i dati dell'applicazione.
- *Migration*: sistema di gestione delle versioni del database fornito da Laravel. Le migration permettono di definire e modificare la struttura del database in modo controllato e replicabile.

Vediamo, ora, nel dettaglio il funzionamento di alcune delle tecnologie utilizzate, che risulteranno molto utili nella fase di sviluppo dell'applicazione vera e propria.

4.3.2 Database e ORM

Come riportato nelle tecnologie utilizzate per il backend, l'implementazione del database progettato nel Capitolo 3 viene realizzata utilizzando MySQL. Tuttavia, Laravel offre un ulteriore potente strumento per la gestione dei dati, ovvero l'ORM (Object-Relational Mapping) *Eloquent*.

Eloquent è il sistema ORM integrato in Laravel che fornisce un modo semplice e intuitivo per interagire con il database utilizzando modelli PHP. Esso consente agli sviluppatori di lavorare con il database in modo orientato agli oggetti, evitando la scrittura diretta di query SQL. Tale approccio presenta numerosi vantaggi:

- *Astrazione del Database*: eloquent permette di manipolare i record del database come se fossero oggetti PHP. Ogni tabella del database viene mappata in un corrispondente "modello" Eloquent che viene utilizzato per interagire con quella tabella. Ad esempio, il modello *User* corrisponderà alla tabella *users*.
- *Facilità d'Uso*: eloquent semplifica l'esecuzione delle operazioni CRUD, grazie alla sintassi espressiva e leggibile che fornisce. Le operazioni che normalmente richiederebbero diverse righe di codice SQL possono essere eseguite con una semplice chiamata di metodo. Ad esempio se vogliamo ottenere il primo utente della nostra tabella users, ci basterà richiamare soltanto questo metodo:

```
$user = User::find(1);
```

- *Query Builder*: eloquent consente anche di costruire query SQL più complesse in maniera intuitiva e semplice, attraverso un potente query builder. Ad esempio, se vogliamo interrogare il database per trovare tutti gli utenti con ruolo operatore, con nome "Mario" e ordinati in ordine alfabetico, basterà eseguire il seguente codice:

```
$users = User::where('ruolo', 'operatore')
         ->where('nome', 'Mario')
         ->orderBy('nome', 'asc')
         ->get();
```

Esistono molti altri benefici legati all'utilizzo di questa interfaccia potente e intuitiva per interagire con il database, che spaziano anche verso la sicurezza e l'integrità dei dati, rendendola una scelta molto conveniente per la gestione del database.

Altri strumenti molto utili, che abbiamo già menzionato in precedenza, sono: le *migration* e i *seeder*.

- le *migration*: esse sono un sistema per il versioning del database, simile al controllo di versione del codice sorgente. Esse permettono di definire la struttura delle tabelle del database utilizzando il codice PHP. Ciò consente di creare, modificare e gestire il database in modo programmatico e riproducibile. Tutto ciò rende semplice anche il processo di deployment e rollback delle modifiche al database, migliorando la gestione delle versioni e la continuità del progetto.
- i *seeder*: essi sono utilizzati per popolare il database con dati iniziali o di test. Essi permettono di definire e inserire automaticamente record predefiniti nelle tabelle del database.

Questi strumenti, combinati con l'ORM Eloquent, forniscono un robusto ecosistema per la gestione del database, che sappiamo avere un ruolo fondamentale all'interno di un applicativo web.

4.3.3 Sicurezza

La sicurezza è un aspetto non trascurabile nello sviluppo di applicazioni web, specialmente per il backend, che gestisce dati sensibili e interagisce direttamente con il database. Utilizzando Laravel, diverse misure di sicurezza sono integrate per proteggere l'applicazione da vari tipi di attacchi. Ecco una panoramica delle principali funzionalità di sicurezza implementate in Laravel:

- *Middleware*: essi vengono utilizzati per proteggere le rotte, controllando l'accesso basato sui ruoli e sui permessi.
- *Protezione CSRF*: Laravel implementa la protezione CSRF (Cross-Site Request Forgery) attraverso l'utilizzo dei CSRF token. Un token CSRF è un valore segreto, unico e casuale, generato dall'applicazione e incluso in tutte le richieste non GET che modificano i dati (POST, PUT, DELETE). Per ogni richiesta viene verificato se il token CSRF inviato con essa coincide con quello memorizzato nella sessione. Se i token non coincidono, la richiesta viene rifiutata.
- *Protezione XSS*: gli attacchi XSS si verificano quando un aggressore riesce a iniettare codice JavaScript malevolo in una pagina web, che viene poi visualizzata da altri utenti. Per contrastare tale minaccia, Laravel utilizza il motore di template Blade, che offre un sistema di escaping automatico dei dati. Ciò significa che quando i dati devono essere visualizzati, Blade converte automaticamente tutti i caratteri che potrebbero essere interpretati come codice HTML o JavaScript in una forma sicura, che il browser renderà testo normale. Tale processo previene l'esecuzione di codice malevolo, garantendo che i contenuti iniettati non possano attivarsi.
- *Hashing delle Password*: per garantire la sicurezza delle password degli utenti, Laravel implementa un sistema di hashing che utilizza l'algoritmo bcrypt. Prima che le password siano salvate nel database, vengono criptate, trasformandole in una serie di caratteri incomprensibili. In questo modo, anche in caso di accesso non autorizzato al database, le password criptate risulterebbero inutilizzabili per i malintenzionati.

Arrivati a questo punto, la fase di progettazione del sito web gestionale è terminata. Il passaggio successivo è la realizzazione vera e propria del sistema informatico e, quindi, l'implementazione di tutto ciò che è stato definito fino a ora.

CAPITOLO 5

Implementazione

In questo capitolo, l'obiettivo è fornire una visione dettagliata delle metodologie utilizzate per l'implementazione del sistema informativo, evidenziando le fasi chiave e le decisioni tecniche prese. Si inizierà con il setup del progetto, descrivendo le configurazioni iniziali necessarie per avviare l'ambiente di sviluppo.

Successivamente, verranno illustrate le implementazioni di alcune funzionalità principali, seguendo l'intero processo che inizia con la definizione delle rotte, prosegue mostrando le azioni dei controller associati e termina nella visualizzazione delle viste con i dati richiesti. Infine, si discuterà della pianificazione dei test e del deployment, delineando le strategie necessarie per garantire che l'applicazione funzioni correttamente e sia pronta per l'ambiente di produzione.

5.1 Setup del progetto

Per avviare efficacemente la fase di implementazione dell'applicazione web, è indispensabile effettuare prima la fase di configurazione generale del progetto. Ciò consente di iniziare a sviluppare le funzionalità e la logica di business dell'applicazione in maniera organizzata e strutturata.

5.1.1 Strumenti utilizzati

Prima di iniziare la scrittura del codice, è necessario identificare e configurare gli strumenti che verranno utilizzati sia per il setup iniziale sia per lo sviluppo continuo dell'intero sito web. Tali strumenti sono:

- *Git e GitHub*: Git è un sistema di controllo di versione distribuito utilizzato per gestire le modifiche al codice sorgente durante lo sviluppo. GitHub è una piattaforma di hosting per repository Git, che facilita gestione del codice.
- *Visual Studio Code*: VS Code è un editor di codice leggero e potente che offre integrazione diretta con Git, facilitando la gestione delle modifiche al codice sorgente e il controllo di versione. La scelta di questo IDE è stata influenzata dalla presenza di estensioni utili per lo sviluppo con Laravel, come:
  - *PHP Intelephense*: fornisce un supporto avanzato per PHP, inclusa l'auto completamento del codice, il suggerimento dei metodi e la navigazione tra i file.

  - *Laravel Blade Snippets*: offre snippet di codice per Blade, il motore di template di Laravel, accelerando lo sviluppo delle viste.
  - *Laravel Artisan*: integra comandi di Artisan direttamente nell'editor, consentendo di eseguire facilmente task comuni, come la creazione di controller, modelli e migrazioni.

- *XAMPP*: è una distribuzione che include Apache, MySQL e PHP, ideale per creare un ambiente di sviluppo locale. XAMPP semplifica la configurazione di un server web completo su una macchina locale, permettendo di testare e sviluppare l'applicativo in un ambiente PHP controllato e simile a quello di produzione.
- *Composer*: è un gestore di dipendenze per PHP, utilizzato per installare e gestire le librerie e i pacchetti necessari al progetto. In questo caso, Composer è stato utilizzato per integrare il package *Breeze* in Laravel, semplificando l'implementazione di un sistema di autenticazione predefinito e funzionale.

5.1.2 Creazione del progetto Laravel

Una volta installato PHP tramite XAMPP e Composer, è possibile creare un nuovo progetto Laravel utilizzando il comando di Composer:

```
composer create-project laravel/laravel:^10.0 gestione-rifiuti
```

Questo comando crea una nuova installazione di Laravel, specificamente della versione 10, nella cartella corrente del terminale. Il nome del progetto sarà *gestione-rifiuti*. Durante l'installazione, Laravel struttura automaticamente il progetto in una serie di cartelle, seguendo la logica del pattern architetturale MVC.
Il framework, organizza le cartelle principali come segue:

- Cartella *app*: contiene i modelli dei dati, i controller e i file di "request"; questi ultimi sono utili per la validazione dei dati inviati al server.
- Cartella *database*: include le migrazioni del database e i seeder per popolare quest'ultimo con dati di esempio.
- Cartella *public*: rappresenta la cartella pubblica accessibile tramite il web server. Contiene il file *index.php*, che funge da punto di ingresso per l'applicazione.
- Cartella *resources*: include le viste, implementate utilizzando Blade, il motore di template di Laravel.
- Cartella *routes*: contiene i file di definizione delle "route", che determinano come l'applicazione risponde a una richiesta HTTP specifica.

L'implementazione della maggior parte di questi elementi, nel caso specifico della nostra applicazione, verrà mostrata successivamente.

5.1.3 Configurazione del file .env

Tra i file generati durante la creazione di un progetto Laravel è presente il file *.env*. Questo è un file di configurazione per l'applicazione Laravel che contiene variabili d'ambiente. Tali variabili determinano le impostazioni specifiche dell'ambiente dell'applicazione, come le credenziali del database, le impostazioni di connessione e altre configurazioni sensibili.

Per ragioni di sicurezza, il file *.env* non viene incluso nel controllo di versione con git.

Di seguito sono riportate alcune delle impostazioni per la connessione al database generate da XAMPP:

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=gestione_rifiuti
```

5.1.4 Migrazioni database

Ora che il progetto è stato creato e il database è stato connesso, possiamo procedere con la creazione dei file di migrazione per ciascuna delle tabelle del nostro database, descritte nel Capitolo 3.

Le migrazioni in Laravel dispongono di due metodi principali: *up()* e *down()*. Il metodo *up()* è utilizzato per applicare le modifiche al database, come la creazione di nuove tabelle, mentre il metodo *down()* serve per annullare tali modifiche, eliminando le tabelle create. Per semplicità e chiarezza, mostreremo solo i metodi *up()*, in quanto sono quelli che implementano le modifiche necessarie al database.

Di seguito presentiamo alcune delle migrazioni più significative.

La prima migrazione crea la tabella *users* nel nostro database. Questa migrazione include alcune parti preimpostate dal package *Breeze* (relative alla sessione), che gestisce l'autenticazione degli utenti. Il metodo *up()* di questa migrazione è il seguente:

```
public function up(): void
    {
        Schema::create('users', function (Blueprint $table) {
            $table->increments('id')->primary();
            $table->string('nome',30);
            $table->string('cognome',30);
            $table->date('dataNascita')->nullable();
            $table->string('residenza',50)->nullable();
            $table->string('email',50);
            $table->string('username',30);
            $table->string('password');
            $table->string('ruolo',30);
            $table->rememberToken();
            $table->timestamps();
        });

        Schema::create('sessions', function (Blueprint $table) {
            $table->string('id')->primary();
            $table->foreignId('user_id')->nullable()->index();
            $table->string('ip_address', 45)->nullable();
            $table->text('user_agent')->nullable();
            $table->longText('payload');
            $table->integer('last_activity')->index();
        });
    }
```

La migrazione relativa alla tabella *giro_raccolta* è la seguente:

```
public function up()
{
    Schema::create('giro_raccolta', function (Blueprint $table) {
        $table->increments('id')->primary();
        $table->time('ora_inizio');
        $table->time('ora_fine');
        $table->string('tipologia_rifiuto', 30);
        $table->integer('frequenza')->unsigned()->nullable(false);
        $table->string('stato', 30);
    });
}
```

Per quanto riguarda la migrazione che crea la tabella *itinerario_effettuato*, il metodo *up()* è il seguente:

```
public function up()
{
    Schema::create('itinerario_effettuato',
    function (Blueprint $table) {
        $table->increments('id')->primary();
        $table->date('data_itinerario');
        $table->unsignedInteger('giro_raccolta');
        $table->foreign('giro_raccolta')
        ->references('id')->on('giro_raccolta');
        $table->time('ora_inizio');
        $table->time('ora_fine')->nullable();
        $table->boolean('anomalia')->default(false);
        $table->unsignedInteger('operatore');
        $table->foreign('operatore')
        ->references('id')->on('users');
    });
}
```

La migrazione per la creazione della tabella *bidone* è la seguente:

```
public function up()
{
    Schema::create('bidone', function (Blueprint $table) {
        $table->increments('id')->primary();
        $table->string('tipologia_rifiuto', 30)->nullable(false);
        $table->string('indirizzo', 50)->nullable(false);
    });
}
```

La migrazione per la tabella *operazione_raccolta*, invece, è la seguente:

```
public function up()
{
    Schema::create('operazione_raccolta',
    function (Blueprint $table) {
        $table->increments('id')->primary();
        $table->integer('ordinale')->nullable(false);
```

```
        $table->unsignedInteger('bidone');
        $table->unsignedInteger('giro_raccolta');
        $table->foreign('bidone')
        ->references('id')->on('bidone');
        $table->foreign('giro_raccolta')
        ->references('id')->on('giro_raccolta')
        ->onDelete('cascade');
        });
}
```

In questa migrazione, l'uso del comando:

```
->onDelete('cascade');
```

indica che se un record della tabella *giro_raccolta* viene eliminato, i record corrispondenti nella tabella *operazione_raccolta* saranno eliminati a cascata, prevenendo problemi di integrità referenziale.

Una volta completate tutte le migrazioni, è sufficiente eseguire il comando Artisan seguente per applicare le migrazioni e creare le tabelle nel database:

```
php artisan migrate
```

È importante notare che il codice PHP all'interno di queste migrazioni rappresenta l'equivalente delle query SQL per la creazione delle tabelle, descritte nel Capitolo 3.

5.1.5 Configurazione Model

Seguendo il principio imposto dal pattern MVC, questa sezione illustra i principali modelli di dati utilizzati nella nostra applicazione. Come spiegato in precedenza, queste classi sono mappate una a una con le tabelle del database, e ogni record di queste tabelle sarà manipolabile tramite un'istanza del modello corrispondente, grazie all'ORM di Laravel, ovvero Eloquent.

Il modello relativo ai giri di raccolta, *GiroRaccolta*, è il seguente:

```
class GiroRaccolta extends Model
{
    protected $table = 'giro_raccolta';
    protected $fillable =
    ['ora_inizio', 'ora_fine', 'tipologia_rifiuto',
    'frequenza', 'stato'];

}
```

La riga:

```
protected $table = 'giro_raccolta';
```

lega il modello alla tabella *giro_raccolta* del nostro database.

Il modello relativo agli itinerari effettuati, *ItinerarioEffettuato*, è il seguente:

```
class ItinerarioEffettuato extends Model
{
    protected $table = 'itinerario_effettuato';
    protected $fillable = ['data_itinerario',
    'ora_inizio', 'ora_fine', 'anomalia'];
}
```

Il modello relativo ai bidoni, *Bidone*, è il seguente:

```
class Bidone extends Model
{
    protected $table = 'bidone';
    protected $fillable = ['tipologia_rifiuto', 'indirizzo'];
}
```

Il modello relativo alle operazioni di raccolta, *OperazioneRaccolta*, è il seguente:

```
class OperazioneRaccolta extends Model
{
    protected $table = 'operazione_raccolta';
    protected $fillable = ['ordinale'];

}
```

Per ragioni di utilità, è stato creato un ulteriore modello, *Gestione*, che contiene tutti i metodi per il recupero dei dati dal database. Questo risulterà molto utile all'interno dei controller, poiché definendo al loro interno un'istanza di questo modello, sarà possibile utilizzare questi metodi per interrogare il database e recuperare i dati necessari allo svolgimento delle funzionalità richieste.

Un esempio di metodo contenuto in questa classe, che permette di recuperare dal database tutti i giri di raccolta presenti, è il seguente:

```
public function getAllGiriRaccolta()
{
    $giri = GiroRaccolta::get();
    return $giri;
}
```

Questi modelli rappresentano l'intero schema di gestione dei dati della nostra applicazione. Ogni modello è strettamente legato alla rispettiva tabella del database, facilitando l'interazione con i dati e garantendo un'implementazione coerente e sicura. Una volta completata la configurazione dei modelli, possiamo procedere al secondo pilastro del pattern MVC: i Controller.

5.1.6 Configurazione Controller

I controller sono le classi che integrano la logica di business dell'applicazione, essi contengono metodi che vengono richiamati attraverso le rotte. Tali metodi gestiscono le richieste, elaborano i dati e restituiscono le viste coerenti con esse, eventualmente iniettate con i dati necessari.

Poiché la nostra applicazione comprende due grandi sezioni con esigenze e richieste differenti, definite dai ruoli "Amministratore" e "Operatore", sono stati creati due controller

distinti: *AdminController* e *OperatoreController*. Questi gestiscono, rispettivamente, le richieste dell'amministratore e quelle degli operatori durante lo svolgimento dei giri di raccolta.

All'interno di entrambi i controller creiamo un'istanza del nostro modello *Gestione*, così da poterne sfruttare i metodi per il recupero dei dati dal database:

```
class AdminController extends Controller
{
    protected $_gestioneModel;

    public function __construct()
    {
        $this->_gestioneModel = new Gestione;
    }

    // Metodi per la gestione delle richieste dell'amministratore
    ...
}

class OperatoreController extends Controller
{
    protected $_gestioneModel;

    public function __construct()
    {
        $this->_gestioneModel = new Gestione;
    }

    //Metodi per la gestione delle richieste degli operatori
    ...
}
```

L'ultimo step di configurazione è la creazione del layout per le viste. Creare un layout è utile poiché, utilizzando Blade, possiamo strutturare l'applicazione in modo modulare e riutilizzare i layout in diverse parti dell'applicazione.

5.1.7 Configurazione View

Blade, il motore di templating di Laravel, consente di definire layout di base che possono essere estesi dalle singole viste, favorendo, così, la modularità e il riutilizzo del codice. Di seguito è riportato un esempio del layout utilizzato per tutte le viste relative alle sezioni dell'amministratore:

```
    <html lang="{{ str_replace('_', '-', app()->getLocale()) }}">
    <head>
        <title>Gestione rifiuti</title>
        <meta charset="utf-8">
        <meta name="viewport"
        content="width=device-width, initial-scale=1">
        @section('link')
        @vite('resources/css/app.css')
        @show
```

```
        @section('scripts')
        @show
    </head>
    <body>
        <div class="flex bg-gray-100 min-h-screen">
            @include('layouts/_sidenavadmin')
            <div class="flex-grow">
                @yield('content')
            </div>
        </div>

        <footer class="bg-green-900 h-16 flex items-center mt-auto">
            <p class="text-green-400 ml-24">
            P.IVA: 02059030254 | ®2024</p>
        </footer>
    </body>
</html>
```

In Blade, il comando `@section` è usato per definire sezioni di contenuto che possono essere sovra scritte o riempite dalle viste figlie che estendono un layout principale. In particolare, all'interno del layout presentato ne troviamo due:

- Sezione *link*: questa sezione consente di specificare uno spazio nel quale includere link a file CSS necessari per lo stile della pagina HTML. Il comando `@vite('resources/css/app.css')` integra il file CSS compilato da Vite, garantendo che tutti gli stili di base siano applicati.
- Sezione *scripts*: questa sezione è riservata all'inclusione di eventuali file JavaScript. In tal modo, il layout principale rimane flessibile e consente alle viste figlie di aggiungere script specifici per le loro esigenze.

La barra di navigazione verticale, che permette all'amministratore di spostarsi tra le varie sezioni del menù, viene inclusa nel layout principale attraverso il comando:

```
@include('layouts/_sidenavadmin')
```

La barra di navigazione si trova in un file separato (`_sidenavadmin.blade.php`) e viene importata nel layout principale, seguendo sempre questo principio di modularità.

Un elemento fondamentale del layout è il comando:

```
@yield('content')
```

Questo funge da segnaposto nel layout principale e viene sostituito dal contenuto definito nella vista figlia. Ogni vista figlia, estendendo il layout principale, può inserire il proprio contenuto specifico nella posizione definita da `@yield('content')`, garantendo, così, una struttura unificata per tutte le pagine.

Vediamo allora un esempio di una vista figlia per comprendere appieno questo meccanismo di "ereditarietà":

```
@extends('layouts.public')
@section('title', 'Lista bidoni')
@section('scripts')
```

```
@parent

<script>

$(document).ready(function () {
    $('#aggiungiBidoneButton').click(function () {
        window.location.href = "{{ route('crea_bidone') }}";
    });
});

</script>
@endsection

@section('content')

<h2 class="text-3xl font-extrabold text-green-900 my-5 ml-14">
Bidoni: </h2>

\\codice html della singola vista

@endsection
```

In questa vista, possiamo osservare vari aspetti chiave dell'estensione del layout principale; essi sono:

- *Estensione del Layout*: la direttiva `@extends` indica che questa vista figlia estende il layout `public.blade.php`, definito in precedenza.
- *Ereditarietà degli Script*: qui ereditiamo gli script definiti nel layout principale tramite `@parent` e aggiungiamo un nuovo script specifico per questa vista, che pone un listener sul click di un pulsante.
- *Definizione del Contenuto*: la sezione `@section('content')` definisce il contenuto specifico di questa vista, che verrà inserito al posto di `@yield('content')` nel layout principale.

Questo approccio nello sviluppo delle viste dell'applicazione web offre numerosi vantaggi, in particolare per quanto riguarda il riutilizzo del codice. Le sezioni comuni non devono essere riscritte ogni volta, permettendo di mantenere una coerenza in tutte le parti dell'applicazione pur consentendo la personalizzazione del contenuto specifico per ciascuna vista.

5.2 Sviluppo delle funzionalità

Terminata la fase di configurazione del progetto, il passo successivo è implementare i requisiti funzionali e non funzionali definiti durante la fase di progettazione. Questo è lo stadio in cui viene definita la logica di business dell'applicazione.

In particolare, ci si concentra sulla definizione delle rotte, sulla realizzazione delle azioni nei controller chiamate da esse, sulla creazione delle viste da mostrare all'utente e sulla definizione delle regole di validazione dei dati inseriti nei form di queste viste.

Per motivi di praticità, verranno presentate solo alcune sezioni più significative e interessanti dell'intero sito web realizzato.

5.2.1 Giri di raccolta

L'intera gestione dei giri di raccolta da parte dell'amministratore, elemento centrale di tutta l'applicazione, risulta molto interessante in quanto presenta alcune funzioni uniche. In particolare, ci concentreremo sulle rotte per la visualizzazione dei giri di raccolta attivi, ovvero la home dell'amministratore, e sulla parte di creazione dei giri di raccolta, con la possibilità di programmarli in diverse giornate e di definirne un percorso di bidoni.

Le rotte utilizzate in questi processi, sono le seguenti:

```
// Rotta 1: Visualizzazione dei giri di raccolta attivi
Route::get('/home_giro_raccolta',
[AdminController::class, 'allActiveGiriRaccolta'])
    ->name('home_giro_raccolta')
    ->middleware('admin');

// Rotta 2: Creazione di un nuovo giro di raccolta
Route::get('/crea_giro_raccolta',
[AdminController::class, 'addGiroRaccolta'])
    ->name('crea_giro_raccolta')
    ->middleware('admin');

// Rotta 3: Recupero dei bidoni per la crazione di un percorso
Route::get('/crea_giro_raccolta/{tipologia}',
[AdminController::class, 'getBidoniAddGiro'])
    ->name('crea_giro_raccolta.get_bidoni')
    ->middleware('admin');

// Rotta 4: Memorizzazione di un nuovo giro di raccolta
Route::post('/crea_giro_raccolta',
[AdminController::class, 'storeGiroRaccolta'])
    ->name('crea_giro_raccolta.store')
    ->middleware('admin');
```

Queste rotte attivano specifici metodi nell'*AdminController*, che gestisce le richieste e le operazioni necessarie per soddisfarle.

5.2.2 Giri di raccolta - Home giro raccolta

Il metodo `allActiveGiriRaccolta`, attivato dalla rotta 1, recupera tutti i giri di raccolta attivi e i giorni in cui sono programmati, e passa queste informazioni alla vista `admin.home_giro_raccolta`. Segue l'implementazione del metodo:

```
public function allActiveGiriRaccolta() {
    $giri = $this->_gestioneModel->getAllActiveGiriRaccolta();
    $progActive = $this->_gestioneModel->getActiveProg();

    return view('admin.home_giro_raccolta')
                ->with('allActiveGiri', $giri)
                ->with('allProg', $progActive);
}
```

I dati vengono recuperati tramite il modello *Gestione*, che utilizza Eloquent per interagire con il database.

In particolare, il metodo `getAllActiveGiriRaccolta` è il seguente:

```
public function getAllActiveGiriRaccolta() {
    return GiroRaccolta::where('stato', '!=', 'non_attivo')->get();
}
```

Il metodo `getActiveProg` è leggermente più complesso in quanto richiede un join tra tabelle; esso è il seguente:

```
public function getActiveProg() {
    return Programmazione::join('giro_raccolta',
    'programmazione.giro_raccolta', '=', 'giro_raccolta.id')
        ->where('giro_raccolta.stato', '!=', 'non_attivo')
        ->get();
}
```

La vista `home_giro_raccolta` utilizza le direttive Blade per visualizzare i dati passati dal controller. Dato lo spazio limitato che si ha a disposizione, verranno mostrate solo le parti salienti del codice.

```
<div class="grid grid-cols-3 gap-4 mx-10 mb-4">
    @isset($allActiveGiri)
    @isset($allProg)
    @foreach($allActiveGiri as $giro)
        <div class="max-w-sm ..." onclick="window.location.href =
        '{{ route('dettaglio_giro'[$giro->id]) }}';">
            @if($giro->stato == 'da_eseguire')
                <div class="flex ...">
                    <p class=" text-white">Da eseguire</p>
                </div>
            @elseif($giro->stato == 'assegnato')
                <div class="flex ...">
                    <p class=" text-white">Assegnato</p>
                </div>
            ...
            @endif
            <div class="px-6 py-4">
                <div class="font-bold text-xl text-green-800 mb-2">
                G{{ sprintf('%04d', $giro->id) }}</div>
                <p class="text-green-800"><strong>Inizio:</strong>
                {{ $giro->ora_inizio }}</p>
            ...
            </div>
            <div class="px-6 pt-4 pb-2">
                @foreach($allProg as $prog)
                    @if($prog->giro_raccolta == $giro->id)
                        <span>{{ $prog->giorno }}</span>
                    @endif
                @endforeach
```

```
            </div>
        </div>
    @endforeach
    @endisset
    @endisset
</div>
```

In questo contesto, le direttive Blade (`@isset`, `@foreach`, `@if`) permettono di gestire la logica di visualizzazione dei dati iniettati dal Controller nella vista in modo semplice e funzionale.

5.2.3 Giri di raccolta - Crea giro raccolta

Per quanto riguarda il metodo `addGiroRaccolta`, attivato dalla rotta 2, il suo unico compito è quello di restituire la vista `crea_giro_raccolta` contenente il form per la creazione del giro di raccolta.

Più interessante, invece, è la rotta numero 3, che non viene utilizzata per restituire alcuna vista, ma viene sfruttata all'interno di una chiamata AJAX per recuperare tutti i bidoni della stessa tipologia di rifiuto del giro che si sta creando nella vista `crea_giro_raccolta`.

Vediamo ora lo script jQuery che effettua la chiamata:

```
const baseUrl =
"{{ route('crea_giro_raccolta.get_bidoni', ':tipologia') }}";

function fetchBidoni(tipologia) {
   const url = baseUrl.replace(':tipologia', tipologia);
   $.ajax({
      url: url,
      type: 'GET',
      success: function (data) {
         console.log(data);
         if (data.original && Array.isArray(data.original)) {
            updateBidoniList(data.original);
         } else {
            console.error('Formato della risposta valido:', data);
         }
      },
      error: function (xhr, status, error) {
         console.error('Si è verificato un errore:', error);
      }
});
```

Questa chiamata AJAX è necessaria per recuperare i dati in maniera dinamica, senza ricaricare la pagina, poiché un ricaricamento porterebbe a un reset dei dati precedentemente inseriti nel form di creazione del giro di raccolta.

Il codice relativo all'action del controller di questa rotta è il seguente:

```
public function getBidoniAddGiro($tipologia) {
    $bidoni = $this->_gestioneModel
    ->getBidoniByTipologia($tipologia);
    return response()->json($bidoni);
}
```

La risposta inviata dalla funzione `getBidoniAddGiro` è in formato JSON, così da poter utilizzare facilmente i dati inviati nella vista.

Mentre, la funzione del model *Gestione* che recupera i dati dal database è la seguente:

```
public function getBidoniByTipologia($tipologia) {
    $bidoni = Bidone::where('tipologia_rifiuto', $tipologia)->get();
    return response()->json($bidoni);
}
```

Se la chiamata va a buon fine, viene lanciata la funzione `updateBidoniList`, a cui vengono passati tutti i bidoni recuperati dalla richiesta AJAX. Questa funzione cicla su ognuno di essi, creando dinamicamente un checkbox affiancato da una label HTML, per poi mostrarli nella lista dei bidoni da scegliere per il giro, inizialmente vuota.

Vediamo alcune parti di codice che permettono ciò:

```
function updateBidoniList(bidoni) {
    $('#bidoni_list').empty();

    $.each(bidoni, function (index, bidone) {

    const checkbox = $('<input>').attr({
        type: 'checkbox',
        id: 'bidone_' + bidone.id,
        name: 'bidoni[]',
        value: bidone.id
    });
    const label = $('<label>')
    .attr('for', 'bidone_' + bidone.id)
    .text('B' + bidone.id + ' - ' + bidone.indirizzo)
    .addClass('text-green-900 text-base pl-3');

    $('#bidoni_list').append(checkbox)
    .append(label)
    .append('<br>');
    });
}
```

La lista dei bidoni si trova all'interno di un modal. Poiché queste check box faranno parte di un form in cui l'utente seleziona i bidoni, è importante settare gli attributi `name` e `value` per garantire che i dati vengano inviati correttamente al server.

Una volta confermati i bidoni che devono essere presenti nel giro, attraverso un'altra funzione JavaScript, questi appariranno nella pagina iniziale, affiancati da una *select* che permette di impostare l'ordine di ogni bidone nel percorso di raccolta.

5.2.4 Giri di raccolta - Salva giro di raccolta

L'ultima delle rotte, la numero 4, è predisposta per memorizzare il giro di raccolta all'interno del database. Questo è evidente dal fatto che la chiamata è di tipo POST, utilizzata quando deve essere manipolato il database.

Il metodo `storeGiroRaccolta` si occupa del salvataggio di tutti i dati relativi al giro di raccolta creato. Questo non è un compito semplice, poiché coinvolge molte delle tabelle del database.

Come accennato in precedenza, Laravel fornisce un sistema di validazione dei dati inseriti nelle form prima che vengano inviati al server. Nel caso del giro di raccolta, il file *request* in cui vengono definite le regole di validazione è il seguente:

```
public function rules(): array
    {
        return [
            'ora_inizio' => 'required|date_format:H:i',
            'ora_fine' => 'required|date_format:H:i
            |after:ora_inizio',
            'tipologia_rifiuto' => 'required|string|max:30',
            'frequenza' => 'required|integer|min:1',
            'giorni' => [
                'required',
                'array',
                'size:' . $this->input('frequenza'),
            ],
        ];
    }
```

Possiamo notare che sono presenti regole come: l'ora di inizio del giro deve essere antecedente a quella di fine, la frequenza settimanale deve essere almeno 1 e il numero di giorni in cui è programmato lo svolgimento del giro devono coincidere con la frequenza. Grazie a questo meccanismo garantiamo che i dati manipolati nel backend siano corretti, migliorando la sicurezza e la stabilità.

Il metodo `storeGiroRaccolta` dell'*AdminController* ha come argomento un oggetto request di questo tipo, contenente tutti i dati inseriti nella form che hanno superato la validazione. Sinteticamente, i passi da seguire sono i seguenti:

```
public function storeGiroRaccolta(NuovoGiroRaccoltaRequest $request)
{
    $giro = new GiroRaccolta;
    $giro->ora_inizio = $request->input('ora_inizio');
    $giro->tipologia_rifiuto = $request->input('tipologia_rifiuto');
    ...
    if($request->has('attivo')){
        $giro->stato = "da_eseguire";
    }
    else{
        $giro->stato = "non_attivo";
    }
    $giro->save();

    $giorniSelezionati = $request->input('giorni');
    foreach($giorniSelezionati as $giorno)
        {
            $prog = new Programmazione;
            $prog->giro_raccolta = $giro->id;
            switch ($giorno) {
                case "lunedi":
                    $prog->giorno = 1;
```

```
                    break;
                case "martedi":
                    $prog->giorno = 2;
                    break;
            ...
            }
            $prog->save();
        }

    $bidoniData = json_decode($request->input('bidoni_data'), true);

    foreach($bidoniData as $bidone){
        $op = new OperazioneRaccolta;
        $op->ordinale = $bidone['ordinale'];
        $op->bidone = $bidone['id'];
        $op->giro_raccolta = $giro->id;
        $op->save();
    }

    return redirect()
    ->route('crea_giro_raccolta')
    ->with('success', 'Giro creato con successo');
}
```

Attraverso l'oggetto *request* siamo in grado di ottenere tutti i dati inseriti nella form di creazione. Grazie a Eloquent, possiamo trattare i record che saranno salvati nel database come se fossero degli oggetti, assegnando i valori corretti ai relativi attributi dei modelli legati alle tabelle *giro_raccolta*, *programmazione* e *operazione_raccolta*. Ciò viene fatto tramite il metodo *->save()*. L'intero processo permette di creare un giro di raccolta completo, con i giorni in cui deve essere eseguito e il percorso di bidoni che deve essere seguito.

Inoltre, nella form di creazione del giro di raccolta, è presente un toggle per decidere se il giro deve essere inizializzato come attivo. Il valore di questa checkbox viene verificato nel codice e lo stato iniziale del giro di raccolta viene impostato di conseguenza.

La decisione di mostrare questa parte dell'implementazione deriva dall'importanza dei giri di raccolta in tutti gli ambiti dell'applicazione e dalla presenza di funzioni interessanti, come quella per la chiamata AJAX. Tuttavia, per soddisfare la totalità dei requisiti, le funzioni implementate sono in un numero molto maggiore rispetto a quelle viste e risulta impossibile mostrarle tutte in questa sede.

5.3 Pianificazione dei test e del deployment

Una volta completata l'implementazione di tutti i requisiti dell'applicazione, è fondamentale pianificare ed eseguire accuratamente le fasi di test e deployment per garantire un prodotto di qualità. Queste fasi assicurano che l'applicazione funzioni come previsto, sia stabile e sicura, e possa essere rilasciata agli utenti finali senza intoppi.

A causa della limitatezza di risorse derivante dal fine didattico dell'intero progetto, la parte di test e di deployment non è stata implementata. Tuttavia, è importante delineare i passi necessari per completare queste fasi in futuro, affinché l'applicazione possa considerarsi completa.

Pianificazione dei Test

La pianificazione dei test prevede diversi passaggi importanti, volti a garantire che l'applicazione sia robusta e affidabile. Tali passaggi sono:

1. *Definizione degli scenari di test*: identificare tutti gli scenari d'uso dell'applicazione, includendo sia i casi d'uso comuni che quelli più complessi o meno frequenti, ma di uguale importanza.
2. *Test Unitari*: scrivere test per verificare singolarmente le funzionalità dei metodi e delle classi dell'applicazione. Questi test aiutano a identificare errori nelle funzioni specifiche senza influenze esterne.
3. *Test di Integrazione*: verificare che i diversi componenti dell'applicazione funzionino correttamente insieme. Questo tipo di test assicura che i moduli interagiscano come previsto.
4. *Test Funzionali*: simulare scenari reali di utilizzo dell'applicazione per garantire che tutte le funzionalità operino correttamente in un ambiente che simula quello reale.
5. *Test di Sicurezza*: identificare e correggere vulnerabilità di sicurezza nell'applicazione, assicurando che i dati degli utenti siano protetti e che l'applicazione non sia suscettibile a minacce di attacco.

Solo attraverso numerosi e accurati test, si può garantire che l'applicazione soddisfi al meglio le esigenze del cliente.

Pianificazione del Deployment

Una volta che l'applicazione ha superato tutti i test, si può procedere con il deployment. Questo processo include diverse fasi per assicurare un rilascio ottimale. Tali fasi sono:

1. *Preparazione dell'Ambiente di Produzione*: configurare il server di produzione, installare le dipendenze necessarie e preparare l'infrastruttura per ospitare l'applicazione. Nel caso studiato, questo ambiente sarà accessibile solo internamente.
    - *Accesso Limitato*: assicurarsi che l'accesso al server sia limitato agli amministratori e agli operatori autorizzati. Utilizzare firewall e regole di accesso per restringere l'accesso IP e garantire che solo il personale autorizzato possa accedere all'applicazione.
2. *Backup e Rollback*: implementare una strategia di backup per salvare lo stato corrente dell'applicazione e dei dati. Inoltre, pianificare una procedura di rollback per tornare a una versione precedente in caso di problemi durante il deployment.
3. *Monitoraggio Post-Deployment*: monitorare l'applicazione in produzione per rilevare e risolvere rapidamente eventuali problemi. Durante questa fase è essenziale ascoltare il feedback degli utenti per identificare e correggere possibili anomalie non previste.
4. *Documentazione*: mantenere una documentazione accurata delle modifiche apportate, delle configurazioni e delle procedure seguite durante il deployment. Questo passaggio non va sottovalutato in ottica di manutenzioni future, poiché facilita la comprensione del sistema e l'intervento su di esso.

Questi sono i passaggi principali che andranno seguiti per garantire che l'applicazione gestionale sia pronta all'uso interno. Pianificare accuratamente e seguire queste fasi contribuirà a garantire il successo del progetto.

CAPITOLO 6

Manuale utente

In questo capitolo, viene presentato il manuale utente della nostra applicazione, fornendo una guida dettagliata per una navigazione efficace tra le diverse pagine e un utilizzo ottimale delle funzionalità disponibili. Si inizierà con un sommario, che offre una panoramica delle sezioni principali del manuale e una descrizione dei contenuti trattati. Successivamente, verranno illustrate nel dettaglio le varie sezioni del manuale. In particolare, la sezione dedicata alla navigazione e all'utilizzo delle funzionalità spiegherà come accedere a tutte le pagine dell'applicazione e come utilizzare, passo passo, le principali funzionalità disponibili. Infine, verrà presentata la sezione delle FAQ, dove saranno fornite risposte a possibili domande che gli utenti potrebbero porsi durante l'utilizzo del sito web.

L'obiettivo di questo capitolo è garantire che ogni utente, indipendentemente dal suo livello di familiarità con l'applicazione, possa sfruttare al meglio tutte le funzionalità offerte e risolvere eventuali problemi in modo autonomo.

6.1 Sommario

Il manuale utente è un documento essenziale che accompagna il software, il cui scopo principale è guidare gli utenti nell'uso corretto e ottimale delle funzionalità offerte. Spesso, per motivi di tempo o budget, questo deliverable viene trascurato. Tuttavia, creare una documentazione efficace porta molti più benefici di quanti si possa pensare.

6.1.1 Scopo del manuale

Il manuale utente del software ha i seguenti obiettivi:

- fornire una panoramica chiara e dettagliata delle funzionalità dell'applicazione;
- guidare gli utenti passo passo nelle operazioni principali, riducendo il rischio di errori;
- assicurare che tutte le funzionalità siano accessibili e comprensibili per gli utenti, indipendentemente dal loro livello di competenza tecnica;
- rispondere alle domande frequenti, migliorando, così, l'esperienza utente.

6.1.2 Struttura del manuale

Il manuale utente relativo alla nostra applicazione è organizzato nelle seguenti sezioni:

1. *Navigazione e utilizzo delle funzionalità*: essa fornisce una panoramica su come raggiungere tutte le pagine disponibili all'interno dell'applicazione e illustra le istruzioni dettagliate per lo svolgimento delle principali attività presenti in ciascuna pagina.
2. *FAQ*: essa fornisce risposte alle domande più frequenti che potrebbero porsi gli utenti.

6.1.3 Vantaggi della redazione di un manuale utente

La realizzazione di un manuale utente, come già accennato, porta numerosi vantaggi; in particolare, menzioniamo i seguenti vantaggi:

- *Facilitazione dell'adozione*: un manuale ben strutturato e chiaro facilita l'adozione del software da parte di nuovi utenti, riducendo il tempo necessario per apprendere il funzionamento dell'applicazione.
- *Riduzione degli errori*: guidando gli utenti attraverso i processi corretti, il manuale contribuisce a ridurre gli errori e le inefficienze nell'uso quotidiano del software.
- *Onere di supporto ridotto*: offrendo agli utenti la possibilità di risolvere i propri problemi, è possibile ridurre l'onere sui canali di supporto umano.

Per questi motivi, è essenziale considerare il manuale utente non come un complemento del software, ma come un componente del sistema stesso.

6.2 Navigazione e utilizzo delle funzionalità

Dopo aver compreso l'importanza della realizzazione di un manuale utente, iniziamo con la stesura di esso, con la struttura definita precedentemente.

In questa sezione del manuale, verranno mostrati i passaggi necessari per raggiungere qualsiasi tipo di pagina all'interno del sistema, così da facilitare agli utenti l'apprendimento delle sezioni nelle quali possono svolgere le loro attività. In ognuna delle pagine, poi, verranno mostrare le istruzioni passo passo per svolgere qualsiasi attività disponibile all'interno di esse.

6.2.1 Accesso e autenticazione

L'accesso al sito web aziendale avviene tramite un URL privato fornito dall'azienda. Per autenticarsi e accedere al sistema, è necessario seguire i seguenti passaggi:

1. *Accesso alla pagina di login*: dalla home page del sito web, fare clic sul collegamento "LOG IN", situato nell'angolo in alto a destra dello schermo (Figura 4.4).
2. *Reindirizzamento alla pagina di autenticazione*: successivamente, si verrà reindirizzati alla pagina dedicata al login, dove sarà possibile procedere con l'autenticazione (Figura 4.5).
3. *Inserimento delle credenziali*: nella pagina di login, è possibile inserire le proprie credenziali di accesso. Questo procedimento è valido per gli utenti di entrambi i ruoli.
4. *Accesso alle aree dedicate*: dopo aver inserito le credenziali, fare click sul pulsante "Log in". Sulla base del ruolo dell'utente appena autenticato, si navigherà verso la propria area dedicata all'interno del sistema.

Una volta effettuato l'accesso, vediamo come navigare nelle proprie aree personali, analizzando, come abbiamo fatto spesso, prima le pagine raggiungibili dall'amministratore e poi quelle degli operatori.

6.2.2 Sezioni dell'amministratore

Se l'utente autenticato ha il ruolo di amministratore, verrà indirizzato alla pagina iniziale mostrata nella prima schermata in Figura 4.6.

Alla sinistra della pagina è presente una barra di navigazione, che è il cuore del sistema di navigazione. Questa barra consente di spostarsi rapidamente tra le quattro sezioni principali dedicate all'amministratore: "Giri di raccolta", "Itinerari effettuati", "Operatori" e "Bidoni". In questo menù è presente, in basso, anche la voce per effettuare il log out dal proprio account.

6.2.3 Admin - Sezione Giri di raccolta

Di default, come detto, una volta effettuato l'accesso come amministratore ci si trova già nella sezione dei giri di raccolta, in particolare nella home page. Vediamo da qui come raggiungere le altre pagine inerenti ai giri di raccolta e quali funzionalità esse permettono di svolgere.

- *Gestione dei giri di raccolta*: per raggiungere la pagina dedicata alla gestione dei giri (Figura 4.6), fare click sul pulsante "Gestisci" presente in cima alla home page. All'interno di questa pagina, l'utente avrà le seguenti funzionalità:

  1. *Filtrare i giri di raccolta*: in alto nella pagina sono presenti tre input di selezione, ciascuno dei quali permette di filtrare i giri di raccolta secondo diversi criteri. Per attivare i filtri, fare clic sul filtro relativo e selezionare una delle voci presenti nel menù a tendina che verrà aperto.
     NOTA: è possibile anche combinare i filtri tra loro.

- *Dettaglio di un giro di raccolta*: la pagina che mostra i dettagli di un giro di raccolta (Figure 4.9 e 4.10) può essere raggiunta tramite due percorsi:

  – dalla home page dei giri, facendo click sul giro di cui si desidera visualizzare il dettaglio;
  – dalla pagina di gestione dei giri, facendo click sul giro di raccolta di riferimento.

  NOTA: i giri di raccolta con stato "non attivo" sono presenti solo nella sezione di gestione dei giri; quindi per visualizzare il dettaglio di uno di questi, l'unico metodo disponibile è dalla pagina di gestione.

  Le funzionalità disponibili in questa pagina variano in base allo stato attuale del giro di raccolta. In particolare, possono esserci i seguenti giri:

  1. Giro con stato *non attivo*: da qui è possibile attivare il giro cambiando il suo stato da "non attivo" a "da eseguire", facendo click sul pulsante "Attiva".
  2. Giro con stato *da eseguire*: da qui è possibile assegnare questo giro di raccolta a un operatore; per farlo, è necessario seguire questi passaggi:
     (a) fare click sul pulsante "Assegna", che mostrerà una sezione con tutti gli operatori disponibili;
     (b) selezionare l'operatore desiderato facendo click sul pulsante di opzione relativo;
     (c) confermare l'operazione facendo click sul pulsante "Conferma".

     NOTA: è possibile selezionare un solo operatore per volta; tentare di selezionare più di un operatore porterà al deselezionamento dei precedenti.

3. Giro con stato *assegnato*: da qui è possibile visualizzare i dettagli dell'operatore assegnato facendo click sul nome dell'operatore. In essa, è anche possibile dissociare l'operatore dal giro riportando lo stato del giro a "da eseguire". Ecco i passaggi per farlo:
   (a) fare click sul pulsante "Dissocia", posizionato alla destra del nome dell'operatore;
   (b) fare click sul pulsante rosso "Si, conferma", nella scheda di conferma mostrata.
4. Giro con stato *eseguito*: quando un operatore termina un giro di raccolta, fino alla scadenza del giorno in cui deve essere nuovamente eseguito, lo stato del giro risulterà "eseguito". In particolare, in questa pagina è possibile:
   (a) visualizzare i dettagli dell'operatore che ha eseguito il giro, facendo click sul link relativo al suo nome;
   (b) visualizzare le anomalie, se presenti, sia in maniera grafica, scorrendo nella lista dei bidoni del giro, sia facendo click sulla parola "Si" a destra della dicitura "Anomalie";
   (c) forzare il cambio di stato del giro a "da eseguire", facendo click sull'apposito pulsante "Rincomincia".
5. Giro con stato *non terminato*: quando un operatore termina un giro di raccolta, ma in esso si presentano anomalie non trascurabili, lo stato del giro risulterà "non terminato". In particolare, in questa pagina sono presenti le seguenti funzionalità:
   (a) visualizzare i dettagli dell'operatore che ha eseguito il giro, facendo click sul link relativo al suo nome;
   (b) visualizzare i bidoni non ritirati, i quali saranno evidenziati di rosso;
   (c) assegnare questo giro di raccolta speciale, contenente solo i bidoni non ritirati, a un nuovo operatore con lo stesso meccanismo di assegnazione visto per un giro di raccolta con stato "da eseguire".
6. *Modifica ed eliminazione*: per ogni giro di raccolta, a eccezione di quelli con stato "assegnato", è possibile modificare le informazioni o eliminarlo tramite i pulsanti "Modifica" ed "Elimina".

- *Creazione del giro di raccolta*: per raggiungere la pagina contenente la form di creazione di un nuovo giro di raccolta (Figure 4.7 e 4.8), navigare verso la pagina di gestione dei giri e fare click sul pulsante "Aggiungi". In questa pagina, è possibile:

  1. *Creare un nuovo giro di raccolta*: per aggiungere un nuovo giro di raccolta al sistema, seguire le seguenti istruzioni:
     (a) Definire l'ora di inizio, la tipologia di rifiuto da raccogliere e la frequenza settimanale attraverso i menù di selezione.
     (b) Selezionare i giorni della settimana in cui il giro di raccolta verrà svolto, facendo clic sui selettori relativi.
     NOTA: la frequenza settimanale inserita precedentemente deve coincidere con il numero di giorni selezionati.
     (c) Scegliere se il giro di raccolta deve inizializzarsi con stato "non attivo" o "da eseguire". Di default, il selettore è posizionato su "non attivo"; basterà fare click su di esso per cambiare questa impostazione.
     (d) Definire il percorso dei bidoni facendo click sul pulsante "+" al di sotto della voce "Lista di bidoni" e selezionando i bidoni desiderati, attraverso le check box relative.

(e) Confermare la selezione, facendo click sul pulsante "Conferma" presente in fondo a sinistra. Ciò riporterà il focus alla pagina di creazione del giro, con sostituita al pulsante "+" la lista dei bidoni selezionati. A ogni bidone della lista, sarà associato un menù a tendina, il cui valore indicherà l'ordine di esso nella lista.

(f) Ordinare i bidoni selezionati attraverso il menù a tendina presente a sinistra di ciascun bidone.

(g) Confermare l'operazione facendo click sul pulsante "Conferma".

Con questo, terminiamo la descrizione delle pagine e delle funzionalità della sezione principale "Giri di raccolta".

6.2.4 Admin - Sezione Itinerari effettuati

Facendo click sulla voce "Itinerari effettuati" nel menù laterale dell'amministratore, si accede alla sezione che mostra lo storico di tutti gli itinerari effettuati dagli operatori (vedi la prima schermata di Figura 4.13). Questa sezione comprende una sola pagina, ma offre diverse funzionalità:

1. *Filtrare gli itinerari effettuati*: in alto nella pagina sono presenti due input di selezione, ciascuno dei quali permette di filtrare gli itinerari secondo diversi criteri. Per attivare i filtri, è sufficiente fare click sull'input corrispondente e selezionare una delle voci presenti nel menù a tendina.

2. *Ordinare cronologicamente gli itinerari effettuati*: a destra dei due filtri è presente un pulsante raffigurante una freccia. Facendo click su di esso, è possibile ordinare gli itinerari dal più recente al più datato, e viceversa.

   NOTA: lo stato attuale dell'ordinamento è intuibile dalla direzione della freccia: se quest'ultima è rivolta verso il basso, l'ordinamento sarà dal più recente al meno recente; se essa è rivolta verso l'alto, viceversa.

3. *Visualizzare i dettagli delle anomalie*: se in uno degli itinerari sono state riscontrate anomalie, un pulsante raffigurante una freccia sarà presente in basso a destra. Facendo click su questo pulsante, si aprirà una sezione dell'itinerario, precedentemente nascosta, che mostrerà i dettagli relativi ai bidoni su cui sono avvenute le anomalie e la loro tipologia.

6.2.5 Admin - Sezione Operatori

Facendo click sulla voce "Operatori" nel menù laterale dell'amministratore, si accede alla sezione relativa alla gestione del personale operativo. La pagina di ingresso è quella mostrata nella prima schermata di Figura 4.11. All'interno della homepage degli operatori, si possono svolgere le seguenti attività:

1. *Ricercare gli operatori*: in alto a destra nella pagina è presente un campo di testo che permette, una volta inserito il nome e/o il cognome di un operatore, di cercarlo e visualizzare solo esso nella lista.

Vediamo, ora, come raggiungere le altre pagine inerenti agli operatori nonché le funzionalità che esse permettono di svolgere.

- *Creazione di un operatore*: per raggiungere la pagina dedicata alla creazione di un nuovo operatore (Figura 4.11), fare click sul pulsante "Aggiungi" presente in cima alla home page. All'interno di questa pagina, è possibile svolgere i seguenti compiti:

1. *Aggiungere un nuovo operatore e creare il suo account*: per aggiungere un nuovo operatore al sistema e creargli un account per l'accesso al sito, seguire le seguenti istruzioni:
   (a) inserire il nome, il cognome, la data di nascita e la residenza negli appositi campi;
   (b) inserire uno username e una password, che diventeranno le credenziali di accesso dell'operatore;
   (c) confermare l'operazione facendo click sul pulsante "Conferma".

- *Dettaglio di un operatore*: la pagina che mostra i dettagli di un operatore (Figura 4.12) può essere raggiunta tramite diversi percorsi:
  – dalla home page degli operatori, facendo click sull'operatore di cui si desidera visualizzare il dettaglio;
  – dalla pagina di dettaglio di un giro di raccolta, se tale giro è assegnato o è stato completato da quell'operatore.

  Le funzionalità disponibili in questa pagina sono le seguenti, con una piccola variazione se l'operatore è attualmente assegnato a un giro di raccolta:

  1. *Modifica dei dati dell'operatore*: facendo click sul tasto "Modifica" raffigurante una matita, si accederà al modulo di modifica dei dati dell'operatore.
  2. *Eliminazione dell'operatore*: facendo click sul tasto "Elimina" raffigurante una X, dopo aver confermato la richiesta, l'operatore verrà eliminato dal database e le sue credenziali di accesso non saranno più valide.
  3. *Visualizzazione giro assegnato*: se l'operatore è attualmente assegnato a un giro di raccolta, facendo click sul codice del giro di raccolta, sarà possibile passare alla pagina di dettaglio di quel giro (vedi la seconda schermata in Figura 4.12).
     NOTA: prima di poter eliminare un operatore è necessario dissociarlo da un eventuale giro di raccolta.

6.2.6 Admin - Sezione Bidoni

L'ultima delle sezioni disponibili per l'amministratore, accessibile facendo click sulla voce "Bidoni" nel menù laterale, è mostrata nella seconda schermata di Figura 4.13. All'interno di questa pagina, sono disponibili le seguenti funzionalità:

1. *Filtrare i bidoni*: in alto nella pagina è presente un campo di selezione che permette di filtrare i bidoni per tipologia di rifiuto. Per attivare il filtro, fare click sull'input e selezionare una delle voci presenti nel menù a tendina.

2. *Modifica delle informazioni del bidone*: facendo click sul pulsante raffigurante una matita, situato sotto la voce "Azioni" nella riga corrispondente al bidone desiderato, si accederà al modulo di modifica dei dati di quel bidone.

3. *Eliminazione del bidone*: facendo click sul tasto raffigurante una X, situato sotto la voce "Azioni" nella riga corrispondente al bidone desiderato, e confermando la richiesta, il bidone verrà eliminato dal database.

Da questa pagina è possibile navigare verso un'altra pagina dedicata alla creazione di un nuovo bidone. Per accedervi, fare click sul pulsante "Aggiungi" presente in cima alla home page dei bidoni.

- *Creazione di un Bidone*: per aggiungere un nuovo bidone al sistema, seguire le seguenti istruzioni:

    1. inserire la tipologia di rifiuto contenuto nel bidone tramite il campo di selezione e inserire l'indirizzo per la geolocalizzazione tramite il campo di testo;
    2. confermare l'operazione facendo click sul pulsante "Conferma".

Con questo si conclude la parte del manuale dedicata agli utenti con ruolo di amministratore. Passiamo, ora, a illustrare come gli operatori possono muoversi tra le varie sezioni a loro disposizione e come possono svolgere le proprie attività operative.

6.2.7 Sezioni dell'operatore

Quando un utente autenticato ha il ruolo di operatore, viene indirizzato alla pagina iniziale mostrata in Figura 4.14.

Facendo click sull'icona raffigurante tre linee, situata in alto a sinistra, comparirà il menù di navigazione a sinistra della pagina. Questa barra di navigazione consente di spostarsi rapidamente tra le tre principali sezioni con le quali l'operatore dovrà interagire: "Giri di raccolta", "Profilo" e "Itinerari effettuati". In questo menù è presente, in basso, anche la voce per effettuare il logout dal proprio account.

6.2.8 Operatore - Sezione Giri di raccolta

Effettuato l'accesso come operatore, ci si trova già nella sezione dei giri di raccolta, che funge da home page. All'interno di questa pagina, all'operatore vengono mostrati tutti i giri di raccolta assegnatigli, con alcune informazioni per lo svolgimento di essi. Senza muoversi da questa pagina, l'operatore può:

1. *Dichiarare l'inizio di un giro di raccolta*: sotto ogni giro di raccolta mostrato, è presente un pulsante "Inizia". Facendo click su di esso, l'operatore dichiara l'inizio del giro di raccolta.

2. *Dichiarare la fine di un giro di raccolta*: se un giro di raccolta è stato iniziato, il pulsante sotto quel giro mostrerà, ora, la voce "Termina". Facendo click su di esso, l'operatore dichiara la fine del giro di raccolta.

Da questo punto di partenza vediamo come raggiungere le altre pagine, di interesse per operatore, inerenti ai giri di raccolta. Per ognuna di esse verranno illustrate, poi, le attività che consentono di svolgere. Tali attività sono:

- *Dettaglio di un giro di raccolta*: la pagina di dettaglio di un giro di raccolta (Figura 4.15) può essere raggiunta facendo click su uno dei giri di raccolta presenti nella home page. All'interno di questa pagina, si ha una visione più chiara dei dettagli per lo svolgimento del giro di raccolta e, inoltre, è possibile accedere alle seguenti funzionalità:

    1. *Dichiarare l'inizio o la fine del giro di raccolta*: con le stesse modalità viste in precedenza, anche all'interno della pagina di dettaglio, è possibile dichiarare l'inizio e la fine del giro di raccolta del quale si sta visualizzando il dettaglio.
    2. *Visualizzare la mappa del percorso di bidoni*: all'interno della pagina sarà possibile visualizzare una mappa, nella quale sarà evidenziato il percorso da seguire per raggiungere tutti i bidoni. Facendo click sulla mappa sarà possibile spostarsi all'interno di essa.

3. *Visualizzare manualmente il percorso di bidoni*: in fondo alla pagina, è presente una voce "Bidoni" con il numero di bidoni corrispondente. Facendo click sul pulsante adiacente, raffigurante una freccia, sarà possibile visualizzare la lista completa e ordinata di tutti i bidoni da raccogliere nel giro. Facendo nuovamente click sulla freccia, la lista tornerà nascosta.
   NOTA: in basso a destra è presente un pulsante raffigurante una freccia circondata da un cerchio, che permette di tornare immediatamente in cima alla pagina. Essa risulta utile nel caso in cui si sia scesi molto all'interno della pagina, visualizzando la lista aperta dei bidoni.

- *Resoconto dell'itinerario*: la pagina di resoconto dell'itinerario (Figura 4.16) può essere raggiunta da due pagine differenti, ma con le stesse modalità. In particolare, essa si può raggiungere:
  - dalla home page dei giri, facendo click sul pulsante "Termina" di un giro di raccolta precedentemente iniziato;
  - dal dettaglio di un giro, facendo click sul pulsante "Termina", sempre dopo aver iniziato quel giro.

  Qui l'utente può visualizzare l'orario di inizio e di fine effettivo del giro di raccolta appena svolto. Le funzioni disponibili all'operatore in questa pagina sono le seguenti:

  1. *Confermare il termine di un giro di raccolta senza anomalie*: facendo click sul pulsante "Conferma" e lasciando invariate tutte le altre impostazioni, sarà possibile confermare il termine del giro di raccolta effettuato, dichiarando di non aver riscontrato alcuna anomalia.
  2. *Confermare il termine di un giro di raccolta con anomalie*: per dichiarare delle anomalie riscontrate durante il giro di raccolta, seguire le seguenti istruzioni:
     (a) Fare click sul selettore posto a destra della voce "Anomalie?" scritta in rosso. Ciò permetterà di aprire una nuova sezione della scheda (Figura 4.16), contenente la lista di tutti i bidoni che dovevano essere raccolti in quel giro.
     (b) Di default, ogni bidone mostrato nella nuova lista sarà colorato di verde, indicando che non presenta anomalie. Per dichiarare un'anomalia su uno dei bidoni, fare click sul bidone corrispondente nella lista. Ciò aprirà un modulo per l'inserimento dell'anomalia (Figura 4.17).
     (c) Inserire la tipologia di anomalia dal menù a tendina presente e fare click sul pulsante "Ok".
     NOTA: per tutte le tipologie di anomalie, eccetto "Altro", il campo di testo per inserire la descrizione sarà disabilitato.
     (d) Verrà visualizzata nuovamente la pagina precedente, ma il bidone su cui è stata dichiarata l'anomalia sarà evidenziato in rosso. Il processo di segnalazione delle anomalie può essere iterato per tutti i bidoni necessari.
     (e) Confermare l'operazione facendo click sul pulsante "Conferma".

6.2.9 Operatore - Sezione Profilo

Facendo click sulla voce "Profilo" nel menù laterale dell'operatore, si accede alla sezione che mostra lo username e tutti i dati anagrafici dell'operatore attualmente autenticato.

6.2.10 Operatore - Sezione Itinerari effettuati

Facendo click sulla voce "Itinerari effettuati" nel menù laterale dell'operatore, si accede alla sezione che mostra lo storico di tutti gli itinerari effettuati dall'operatore attualmente autenticato. All'interno di questa pagina l'operatore può:

1. *Ordinare cronologicamente gli itinerari effettuati*: in alto è presente un pulsante raffigurante una freccia. Facendo click su di esso, è possibile ordinare gli itinerari dal più recente al più datato, e viceversa.

2. *Visualizzare i dettagli delle anomalie*: se in uno degli itinerari sono state riscontrate anomalie, un pulsante raffigurante una freccia sarà presente in basso a destra. Facendo click su questo pulsante, si aprirà una sezione dell'itinerario, precedentemente nascosta, che mostrerà i dettagli relativi ai bidoni su cui sono avvenute le anomalie e la loro tipologia.

   NOTA: la logica di queste due funzionalità è la stessa dello storico visualizzabile dall'utente con ruolo "Amministratore".

Nella parte del manuale appena mostrata, sono state riportate tutte le funzionalità che offre l'applicazione, con una spiegazione dettagliata su come portare a compimento ognuna di esse. Ora, terminiamo la stesura del manuale con alcune risposte alle domande che gli utenti potrebbero porsi durante l'utilizzo dell'applicativo.

6.3 FAQ

Le FAQ (Frequently Asked Questions) rappresentano una raccolta di domande comuni e delle relative risposte, che gli utenti possono consultare per ottenere rapidamente informazioni su problemi o dubbi ricorrenti riguardanti l'uso dell'applicazione. Tipicamente, questa sezione viene aggiornata man mano che l'applicazione viene utilizzata dagli utenti e si raccolgono i loro feedback.

Siccome l'applicazione non è ancora stata rilasciata, in questa sezione verranno mostrate delle domande, con le relative risposte, sugli argomenti ritenuti meno espliciti e di immediata intuizione. Anche qui, come di consueto, la descrizione verrà suddivisa per ruolo dell'utente.

6.3.1 Domande e risposte - Ruolo amministratore

Domanda 1: Cosa succede se, mentre sto creando un giro di raccolta in cui ho già definito un percorso di bidoni, cambio la tipologia di rifiuto del giro?

- *Risposta*: Poiché i bidoni che possono essere inseriti nel percorso di un giro di raccolta devono essere della stessa tipologia di rifiuto di quel giro, se viene cambiata la tipologia di rifiuto, la lista dei bidoni creata in precedenza verrà cancellata. Prima di eliminare la lista, il sistema avviserà l'utente e richiederà una conferma dell'operazione.

Domanda 2: Quando creo un nuovo giro di raccolta, come faccio a inserire l'ora in cui deve terminare?

- *Risposta*: Non è necessario inserire l'ora di fine del giro di raccolta, poiché essa verrà generata automaticamente a partire da quella di inizio. In particolare viene fissata a sei ore dopo l'orario di inizio.

Domanda 3: Non riesco a creare un giro di raccolta che inizi dopo le 18:00, perché?

- *Risposta*: Dato che la durata standard di un giro di raccolta è di sei ore, creare un giro che inizi dopo le 18:00 comporterebbe una fine oltre la mezzanotte, facendo scattare il nuovo giorno. Per mantenere chiara la definizione dei giorni in cui deve essere svolto un giro di raccolta, ciò non è possibile. Per ottenere un risultato simile, basterà inserire la giornata successiva nei giorni di svolgimento del giro e inserire l'ora di inizio dopo la mezzanotte.

Domanda 4: Cosa succede se, quando creo il percorso di un giro di raccolta, metto lo stesso ordinale a due bidoni differenti?

- *Risposta*: All'interno dell'applicazione è implementato un meccanismo per evitare questo tipo di errore. Quando si proverà a inserire nel campo relativo all'ordine di un bidone, un numero già presente per un altro bidone della lista, apparirà un avviso che segnalerà l'errore.

Domanda 5: Cosa succede all'ordine dei miei bidoni quando ne aggiungo di nuovi o ne rimuovo alcuni già esistenti nella lista?

- *Risposta*: L'applicazione implementa un sistema che gestisce efficacemente questo processo. Nel caso in cui venga aggiunto un nuovo bidone, esso prenderà l'ordine subito successivo all'ultimo bidone della lista precedente. Se viene rimosso un bidone e sono presenti bidoni con un ordine superiore al numero totale dei bidoni, essi verranno automaticamente scalati a valori di ordine validi.

Domanda 6: Nella fase di aggiunta di un nuovo operatore, in particolare nella parte di creazione dell'account corrispondente, i dati che posso inserire devono rispettare dei vincoli?

- *Risposta*: Sì, è importante che ogni operatore abbia uno username unico. Per quanto riguarda la password, è richiesto che essa sia sicura e, quindi, vanno rispettate le seguenti regole:
  - presenza di almeno una lettera maiuscola;
  - presenza di almeno un carattere speciale;
  - lunghezza maggiore di otto caratteri.

Domanda 7: Perché, nella gestione dei giri di raccolta, alcuni di essi vengono mostrati con uno sfondo grigio?

- *Risposta*: Lo sfondo grigio è uno strumento grafico implementato per mostrare immediatamente all'utente quali, tra tutti i giri di raccolta, sono quelli con stato "non attivo".

6.3.2 Domande e risposte - Ruolo operatore

Domanda 1: Posso dichiarare l'inizio di un giro di raccolta mentre ne ho già iniziato un altro?

- *Risposta*: No, una volta iniziato un giro di raccolta, fino al suo termine, non è possibile interagire con i pulsanti di qualsiasi altro giro di raccolta assegnato.

Domanda 2: Perché quando definisco un'anomalia in uno dei bidoni, non mi è possibile scrivere nulla nella descrizione?

- *Risposta*: Di default, il campo per l'inserimento della descrizione è disabilitato, poiché risulta più efficiente gestire le anomalie utilizzando le tipologie preimpostate. Se la casistica lo richiede, selezionando nel menù a tendina la voce "Altro", sarà possibile scrivere una breve descrizione dell'accaduto.

Domanda 3: Per errore ho cliccato il pulsante "Termina" prima di aver effettivamente terminato il giro di raccolta. Posso tornare indietro?

- *Risposta*: Sì, fino a quando non viene premuto il tasto "Conferma" nella pagina di resoconto dell'itinerario, il giro di raccolta non viene considerato terminato ed è sempre possibile tornare indietro senza alcun tipo di problema.

Si conclude, così, il manuale utente, con la speranza che possa risultare utile a tutti gli utenti dell'applicazione durante lo svolgimento delle loro attività quotidiane.

CAPITOLO 7

Confronto con sistemi correlati

In questo capitolo finale, l'intento è fornire un'analisi critica delle pratiche di gestione dei rifiuti adottate nel contesto da noi studiato, confrontandole con quelle di altre località rinomate per la loro efficacia in questo settore. Dopo aver esaminato alcuni dei punti chiave degli approcci implementati da queste località selezionate, procederemo a un confronto dettagliato basato su queste informazioni. Verranno individuati i punti di contatto con il nostro sistema e, successivamente, saranno identificate le principali possibilità di miglioramento, sia a livello nazionale che locale, prendendo spunto proprio da questi contesti studiati.

Infine, si discuterà di come l'adozione di alcune di queste pratiche potrebbe portare allo sviluppo di nuove interessanti funzionalità nel sistema informatico realizzato. L'obiettivo è di fornire una visione completa che permetta di comprendere le potenzialità di crescita del nostro sistema attraverso l'integrazione di alcune delle migliori pratiche internazionali.

7.1 Introduzione

Il percorso seguito finora è iniziato dall'analisi di un sistema di gestione della raccolta rifiuti ed è terminato nella realizzazione di un'applicazione per supportare le attività quotidiane di tale realtà.

La sfida legata al cambiamento climatico, però, impone l'adozione di sistemi di gestione rifiuti sempre più efficienti. A tal proposito, il lavoro svolto non può essere considerato definitivo e legato a un evento isolato e statico. Per rimanere sostenibili nel tempo, è cruciale adattarsi continuamente, poiché le variabili in gioco evolvono rapidamente. In una situazione simile, la possibilità di apprendere dagli altri diventa una risorsa preziosissima.

Seguendo la filosofia appena descritta, al fine di mantenere la competitività nel settore, risulterà importante confrontare il sistema sviluppato con altri sistemi analoghi. Tale confronto offre numerosi vantaggi:

- *Individuazione dei punti di contatto*: analizzare le similitudini tra le metodologie provenienti da diverse realtà aiuta a identificare le pratiche efficaci da mantenere e quelle che possono essere migliorate o gestite diversamente, basandosi sull'esperienza altrui.
- *Adozione di metodologie efficaci*: studiare nuovi approcci, che si sono dimostrati efficaci in contesti simili, permette di considerare l'adozione di tali metodologie per migliorare il proprio sistema.
- *Identificazione delle aree di miglioramento*: il confronto evidenzia i settori con il maggiore potenziale di miglioramento, permettendo di concentrare gli sforzi per aumentare l'efficienza complessiva.

L'obiettivo di questo capitolo è, quindi, quello di confrontare il sistema sviluppato e le metodologie di gestione dei rifiuti implementate con quelle di altri sistemi simili, al fine di trarre degli spunti di riflessione utili.

7.1.1 Precisazioni

Nonostante l'analisi alla base del progetto sia accurata e fondata sulle informazioni di un'azienda reale, è stato necessario un significativo processo di semplificazione nella fase di informatizzazione. Pertanto, le considerazioni che seguiranno si riferiscono principalmente all'intera organizzazione studiata, mentre verranno in seguito presentati spunti interessanti riguardanti il sistema informatico realizzato.

7.2 Approccio adottato per la raccolta differenziata da altre località

Prima di procedere con il confronto dettagliato del sistema da noi proposto, esamineremo le località selezionate e i principali elementi che caratterizzano il loro sistema di gestione dei rifiuti. La scelta di queste località non è del tutto casuale, ma è stata guidata dai seguenti criteri:

- realtà nazionali di rilievo nel settore, per il loro impegno significativo nella raccolta differenziata dei rifiuti;
- contesti nazionali che mostrano alcune criticità e inefficienze nel sistema di gestione dei rifiuti;
- realtà europee e mondiali rinomate per l'efficacia nella gestione dei rifiuti.

Le località italiane selezionate consentono un confronto equo, poiché presentano condizioni iniziali simili al nostro sistema, per esempio nelle normative vigenti o nelle abitudini dei cittadini.

Lo studio delle località europee e mondiali, leader nel settore, offre l'opportunità di esplorare nuovi approcci e comprendere le ragioni del loro successo.

7.2.1 Spiegazione dei termini che verranno utilizzati

Prima di addentrarci nell'analisi, è essenziale comprendere il significato di alcuni termini e concetti che verranno utilizzati durante questo processo.

1. *Raccolta Porta a Porta*: a differenza della raccolta differenziata tradizionale, la raccolta porta a porta sostituisce i cassonetti pubblici con la raccolta domiciliare secondo un calendario stabilito dall'amministrazione comunale. I cittadini devono separare i rifiuti secondo le tabelle comunali prima di consegnarli agli operatori addetti. Questo approccio ha mostrato risultati statistici molto positivi, con un aumento delle percentuali di raccolta differenziata di oltre il 20%, secondo dati ISTAT del 2016.

   Tuttavia, questo sistema è particolarmente efficiente in piccoli comuni o quartieri meno popolati, mentre diventa complesso in grandi città metropolitane e comuni ad elevata densità. Pertanto, spesso è necessario adottare una soluzione combinata, nella quale il successo dipende dalla sinergia tra cittadini e amministrazioni locali.

2. *Tariffazione Puntuale (PAYT)*: la tariffazione puntuale prevede che i cittadini paghino una tassa proporzionale alla quantità di rifiuti indifferenziati prodotti. Questo sistema risulta equo poiché premia i comportamenti virtuosi, incentivando i cittadini a ridurre

la produzione di rifiuti non differenziabili e a partecipare attivamente alla raccolta differenziata.

3. *Centri di Raccolta*: conosciuti anche come stazioni ecologiche o isole ecologiche, sono luoghi sorvegliati e strutturati dove i cittadini possono portare gratuitamente i rifiuti urbani differenziati che, per dimensione o tipologia, non possono essere conferiti nei cassonetti standard o nella raccolta porta a porta.

   Si tratta di un servizio importante perché ha lo scopo di potenziare la raccolta differenziata e disincentivare l'abbandono abusivo dei rifiuti sul territorio.

Chiariti questi termini, le località che saranno analizzate sono le seguenti:

1. Treviso, Italia.
2. Roma, Italia.
3. Copenaghen, Danimarca.
4. San Francisco, USA.

Concentriamoci, quindi, sui punti chiave che caratterizzano il sistema di raccolta di rifiuti (differenziata) di ciascuna di esse.

7.2.2 Treviso

Treviso è una delle città italiane con la più alta percentuale di raccolta differenziata, attestata all'87,5%. Dunque, risulta interessante analizzare i pilastri del sistema di gestione dei rifiuti di questa città, che presenta:

1. *Raccolta Porta a Porta*: il sistema porta a porta è molto efficace e copre quasi tutta la città.
2. *Tariffazione Puntuale*: sistema di tariffazione puntuale ben sviluppato, che incentiva i cittadini a ridurre i rifiuti indifferenziati.
3. *Centri di Raccolta*: disponibili per il conferimento di rifiuti ingombranti e speciali.
4. *Educazione Ambientale*: vengono messe in atto campagne di sensibilizzazione per educare i cittadini sulla corretta separazione dei rifiuti.

Treviso rappresenta un'eccellenza nazionale nel settore; analizzare questa organizzazione può risultare molto costruttivo.

7.2.3 Roma

Per quanto riguarda Roma, la capitale italiana, è noto che essa affronta problemi significativi nella gestione dei rifiuti, con frequenti casi di accumulo di rifiuti non raccolti e difficoltà nel mantenere il servizio regolare. La percentuale di raccolta differenziata è del 53,3%, lasciando ampi margini di miglioramento. Esaminiamo, allora, i punti chiave della gestione dei rifiuti a Roma:

1. *Raccolta Mista*: sistema misto con raccolta porta a porta in alcune zone e cassonetti stradali in altre. La copertura del porta a porta è limitata rispetto ad altre città italiane.
2. *Centri di Raccolta*: disponibili per rifiuti ingombranti e speciali, ma con accesso limitato e spesso non conveniente per i cittadini, aumentando il tasso di abbandono dei rifiuti.

3. *Educazione e Sensibilizzazione*: mancanza di campagne efficaci per educare la popolazione sulla corretta separazione dei rifiuti.

4. *Infrastrutture Insufficienti*: carenza di impianti adeguati per il trattamento e il riciclaggio, costringendo al trasporto dei rifiuti all'estero con costi aggiuntivi e impatto ambientale.

Analizzare, anche, contesti più problematici come quello di Roma, è importante per capire le origini di questi problemi e come evitarli.

7.2.4 Copenaghen

La Danimarca, con il suo primo posto nell'Indice di Performance Ambientale, è innegabilmente un leader globale ed europeo nella sostenibilità. Secondo il Global Waste Index, la Danimarca è anche uno dei paesi migliori per quanto riguarda la gestione dei rifiuti. Analizziamo i punti di forza del sistema di gestione dei rifiuti di Copenaghen, capitale della Danimarca:

1. *Raccolta Mista*: in molte aree, la raccolta dei rifiuti avviene tramite il sistema porta a porta. Nelle zone urbane o con alta densità abitativa, è comune trovare bidoni per la raccolta dei rifiuti.

2. *Cassonetti Intelligenti*: presenza di bidoni smart con sensori di livello per monitorare il riempimento in tempo reale e ottimizzare i percorsi dei camion, riducendo le emissioni di CO2. Alcuni di essi, inoltre, sono dotati di interfacce per fornire feedback ai cittadini sulle pratiche di riciclaggio.

3. *Tecnologie e Infrastrutture*: La Danimarca è leader nella conversione dei rifiuti in energia (WtE). Questo risulta possibile grazie:

    - *Impianti di Incenerimento Efficienti*: bruciano i rifiuti per generare elettricità e calore, riducendo l'uso delle discariche.
    - *Sistemi di Teleriscaldamento*: integrati con impianti di conversione dei rifiuti in energia, fornendo calore a migliaia di abitazioni.
    - *Controllo delle Emissioni*: tecnologie avanzate per minimizzare le emissioni dall'incenerimento.

4. *Gestione dei Rifiuti Organici*: i rifiuti organici sono indirizzati a efficienti impianti di compostaggio o digestione anaerobica per produrre biogas, che può essere utilizzato per l'energia.

5. *Educazione e Sensibilizzazione*: programmi educativi e campagne di sensibilizzazione per coinvolgere la popolazione nella raccolta differenziata, con incentivi per promuovere il corretto riciclaggio.

Nonostante il successo della Danimarca nella gestione dei rifiuti, essa è uno dei paesi con la maggiore produzione di rifiuti pro capite nell'UE, circa 845 kg per persona ogni anno, secondo Eurostat. Una gestione ottimale dei rifiuti dovrebbe partire da una solida prevenzione di essi, piuttosto che da ottime modalità per smaltirli.

L'obiettivo delle analisi comparative condotte, infatti, non è quello di replicare pedissequamente le metodologie di gestione dei rifiuti adottate dalle località che, secondo le statistiche, raggiungono i migliori risultati in questo ambito. Piuttosto, l'intento è quello di trarre ispirazione dalle procedure virtuose e dagli approcci innovativi di tali realtà, garantendo al contempo la continuità delle buone abitudini e delle strategie efficaci già adottate.

7.2.5 San Francisco

San Francisco è una delle prime grandi città a stabilire l'ambizioso obiettivo di "zero waste" entro il 2020. Anche se non ha raggiunto il 100%, ha fatto notevoli progressi. Analizziamo i punti chiave del sistema di gestione dei rifiuti di questa città:

1. *Sistema di Raccolta Trifase*: il sistema è suddiviso in tre flussi principali:
    - *Materiali riciclabili*: carta, plastica, vetro, metalli, ecc. (bidone blu in Figura 7.1).
    - *Rifiuti compostabili*: inclusi scarti alimentari e rifiuti da giardino (bidone verde in Figura 7.1).
    - *Rifiuti della discarica*: tutto ciò che non può essere compostato o riciclato (bidone nero in Figura 7.1).

Figura 7.1: Raccolta a 3 flussi di San Francisco

2. *Leggi e Regolamenti Avanzati*: sono presenti regolamentazioni rigide, come:
    - *Divieto di Sacchetti di Plastica*: sono vietati i sacchetti di plastica monouso e sono promosse alternative riutilizzabili.
    - *Ordinanza sul Compostaggio e Riciclaggio*: San Francisco è stata una delle prime città a rendere obbligatorio il compostaggio e il riciclaggio per residenti e aziende.
    - *Riduzione degli Imballaggi*: regolamenti per ridurre gli imballaggi non riciclabili.
3. *Monitoraggio e Reporting*: analisi dettagliate dei flussi di rifiuti e tassi di riciclaggio, con pubblicazione di rapporti periodici sui progressi verso l'obiettivo "zero waste". Questo monitoraggio continuo consente di valutare l'efficacia delle politiche e applicare miglioramenti necessari nel tempo.

Come molte delle località viste in questo studio, anche San Francisco utilizza infrastrutture e tecnologie avanzate, promuove la raccolta porta a porta per tutti i flussi di rifiuti e incentiva la partecipazione attiva della comunità.

7.3 Confronto con il contesto studiato e il sistema proposto

Come accennato a inizio capitolo, il sistema informatico proposto è stato limitato a solo un'area delle molteplici gestite dal contesto locale analizzato. Pertanto, prima di discutere del sistema sviluppato, vediamo dove si collocano le Marche in questa lista, i punti di contatto con gli approcci adottati da altre località e le aree in cui c'è possibilità di crescita.

Le Marche si posizionano positivamente rispetto alle località analizzate. Infatti, nel 2022, il tasso di riciclo dei principali materiali raccolti è arrivato al 66%, superando di 12 anni l'obiettivo del 65% fissato da Bruxelles per il 2035.

7.3.1 Punti di contatto

Gli obiettivi sopra citati sono stati raggiunti anche grazie all'implementazione di alcune caratteristiche che si sono rivelate particolarmente efficienti nelle località analizzate. In particolare, i punti comuni con queste realtà sono:

1. *Raccolta Mista*: gestione estesa e capillare per diverse tipologie di rifiuti (umido, carta, plastica, vetro, indifferenziato), attuata sia attraverso la metodologia di raccolta porta a porta, dove più opportuno, sia mediante la collocazione di bidoni nelle zone urbane con maggiore densità popolosa.
2. *Tariffazione Puntuale*: adozione di un sistema di tariffazione basato sulla quantità di rifiuti indifferenziati prodotti, incentivando una maggiore responsabilizzazione dei cittadini.
3. *Centri di raccolta differenziata*: presenza di centri di raccolta differenziata in tutte le principali zone necessarie, facilitando lo smaltimento corretto dei rifiuti.
4. *Tentativi di sensibilizzare i cittadini*: implementazione di campagne volte a incentivare la corretta raccolta differenziata, come l'applicazione mobile "Il Rifiutologo", che fornisce informazioni dettagliate sul riciclo e permette di segnalare rifiuti abbandonati.

Questi punti di contatto con le metodologie adottate in altre località permettono di comprendere ciò che si sta facendo bene e di mantenere vive queste pratiche. Un'*unicità* osservata, nei servizi offerti dall'azienda studiata, è il servizio gratuito di ritiro a domicilio di rifiuti domestici ingombranti, come mobili dismessi o grandi apparecchiature elettriche. Tutte pratiche messe in atto per combattere l'abbandono abusivo.

7.3.2 Possibilità di miglioramento nel contesto nazionale

Tuttavia, esempi come quelli di Copenaghen e San Francisco dimostrano quanto sia ancora possibile fare per garantire una sempre migliore sostenibilità ambientale. Nonostante gli sforzi compiuti negli ultimi anni un significativo problema nazionale riguarda le infrastrutture. In Italia, esiste una buona varietà di impianti per la gestione dei rifiuti, ma vi sono notevoli disparità regionali. Alcune regioni dispongono di impianti obsoleti e insufficienti, mentre altre vantano infrastrutture moderne e innovative che, tuttavia, risultano sottoutilizzate. Un primo passo necessario potrebbe essere quello di unificare e dare una direzione chiara ai vari enti che gestiscono i rifiuti a livello nazionale.

Di fatto, prima di ottimizzare i sistemi già ben funzionanti, sarebbe più proficuo per l'ambiente concentrarsi sul portare ogni area d'Italia a un livello di riciclaggio considerabile "buono". Esistono, infatti, zone con un enorme margine di miglioramento, dove sarebbe necessario investire maggiormente in infrastrutture e applicare alcune delle metodologie risultate efficaci dall'analisi effettuata. Altri spunti di miglioramento, tratti da Copenaghen e San Francisco, includono:

- *Infrastrutture nuove*: l'adozione di termovalorizzatori moderni che riducono il volume dei rifiuti e generano energia. Inoltre, rafforzare le infrastrutture per il trattamento dei rifiuti organici potrebbe ridurre significativamente i rifiuti destinati alle discariche, producendo biogas come fonte di energia rinnovabile.
- *Leggi e Regolamenti Avanzati*: San Francisco ha introdotto regolamentazioni stringenti, come il divieto di sacchetti di plastica monouso e l'obbligo di compostaggio e riciclaggio per residenti e aziende, offrendo incentivi finanziari per la riduzione dei rifiuti. In Italia, un rafforzamento delle leggi in questa direzione potrebbe promuovere una maggiore partecipazione di cittadini e imprese.
- *Programmi di Educazione e Sensibilizzazione*: come già menzionato, la collaborazione tra enti e cittadini è fondamentale. Campagne mirate e programmi di formazione nelle scuole e nelle comunità possono incrementare la consapevolezza e l'impegno verso pratiche di smaltimento dei rifiuti corrette.

Logicamente, ogni potenziale miglioramento comporta dei compromessi e deve essere adattato alle specificità locali. Ad esempio, i termovalorizzatori rappresentano una soluzione molto valida per lo smaltimento dei rifiuti non riciclabili, ma, come evidenziato dallo studio condotto in Danimarca, essi potrebbero diventare una giustificazione per ridurre l'attenzione alla riduzione dei rifiuti. È, quindi, fondamentale considerare questi strumenti come un supporto e non come un ostacolo ai progressi fatti verso il concetto di economia circolare.

Sebbene una parte dei miglioramenti possa essere implementata a livello nazionale, ciò non significa che l'impegno dei singoli enti debba essere sottovalutato, poiché sono essi che, uniti, costituiscono il motore di questo meccanismo.

7.3.3 Possibilità di miglioramento nel contesto locale

Relativamente alle analisi e alla situazione attuale del contesto da noi studiato, una componente che potrebbe portare numerosi benefici è quella relativa all'adozione di nuove tecnologie avanzate. Durante l'analisi di Copenaghen, si sono osservati i benefici legati all'implementazione dei cassonetti intelligenti. Questi cassonetti sono dotati di sensori che monitorano il livello di riempimento in tempo reale e offrono la possibilità di fornire feedback visivi agli utenti.

Fino ad ora, si è parlato maggiormente del contesto generale e meno del sistema sviluppato, ma è proprio in questo frangente che possiamo pensare a dei miglioramenti relativi ad esso. In caso di adozione di cassonetti intelligenti, potrebbero essere sfruttate tutte le informazioni che essi forniscono per ottimizzare uno dei punti cardine del sistema software sviluppato: i *giri di raccolta*. Attualmente, la programmazione dei giri di raccolta si basa su stime effettuate con dati pregressi. Tuttavia, trattandosi di stime, non si ha la certezza che l'attuale metodologia sia la più efficiente. Utilizzando, invece, le informazioni in tempo reale relative al riempimento di ogni cassonetto sul territorio, si potrebbe rivedere completamente il sistema di creazione dei giri di raccolta.

Si potrebbe creare un sistema, supportato dall'Intelligenza Artificiale, che in base al livello attuale di riempimento dei cassonetti, generi giri di raccolta con percorsi sempre differenti, concentrandosi esclusivamente sui bidoni che necessitano realmente di essere svuotati. Questo ridurrebbe il numero di viaggi necessari e, di conseguenza, le emissioni di CO2 e i costi operativi. Da ciò, il processo di pianificazione di essi potrebbe essere alleggerito pesantemente e reso più accurato.

Inoltre, la sezione "Bidoni" del sistema potrebbe diventare molto più utile e complessa, con la visualizzazione dei dati di riempimento in tempo reale di ciascun bidone. Cassonetti dotati di sistemi di tracciabilità RFID potrebbero addirittura identificare l'inserimento di

rifiuti errati o contaminati, notificando immediatamente l'amministrazione, che può così intervenire tempestivamente. I feedback visivi, in aggiunta, potrebbero avvisare l'utente in tempo reale dell'errore appena commesso, con messaggi personalizzati volti a migliorarne le abitudini di riciclaggio.

Queste migliorie non devono essere sottovalutate nemmeno per quanto riguarda la parte operativa. Qui, i benefici includerebbero una riduzione del carico di lavoro per gli operatori, una minore esposizione ai rifiuti pericolosi e condizioni di lavoro generali migliorate.

Conclusioni

Durante lo sviluppo della presente tesi, è stata condotta un'analisi approfondita delle pratiche di gestione dei rifiuti di un'azienda locale, esplorando il contesto generale, i processi interni e le varie interazioni tra le entità che ne sono protagoniste. Abbiamo definito i requisiti funzionali e non funzionali del sistema informatico da implementare, abbiamo progettato la componente dati necessaria per la gestione delle informazioni rilevanti, e abbiamo determinato l'architettura applicativa del sistema. La fase di implementazione ha visto la realizzazione pratica delle funzionalità del sistema, con particolare attenzione all'esperienza utente. Successivamente, abbiamo redatto un manuale del software dettagliato per guidare gli utilizzatori finali nella navigazione e nell'utilizzo del sistema. Infine, abbiamo effettuato un confronto critico con altri sistemi di gestione dei rifiuti di località differenti, con l'obiettivo di trarne degli spunti interessanti.

Lungo l'intero percorso, è stata più volte sottolineata l'importanza di adottare un sistema informatico ben progettato e moderno, capace di affrontare le sfide del futuro. In un contesto come quello italiano, dove il margine di miglioramento è ampio, tali sistemi rappresentano un'opportunità da non sottovalutare. Un sistema informatico efficiente può supportare in modo significativo le attività quotidiane delle aziende di gestione dei rifiuti, ottimizzando la gestione dei giri di raccolta, riducendo i costi operativi e le emissioni di CO_2, e migliorando le condizioni di lavoro degli operatori. Non si deve essere scettici di fronte a questa rivoluzione tecnologica, poiché ogni strumento che può contribuire alla lotta contro il cambiamento climatico va preso in considerazione.

Il sistema realizzato durante questo progetto è soltanto un punto di partenza, una fase embrionale di tutto ciò che potrebbe diventare. In futuro, si potrebbe implementare anche la raccolta porta a porta all'interno della piattaforma, con la gestione dei giri che gli operatori devono eseguire presso i domicili dei cittadini, e un sistema di notifiche che aumenti l'interazione con gli utenti, avvisandoli dei giorni e delle modalità con cui avverrà la raccolta. Si potrebbero esplorare nuove tecnologie avanzate, come i sistemi di monitoraggio in tempo reale o altre attrezzature IoT, algoritmi di Intelligenza Artificiale per la pianificazione dinamica delle attività, e piattaforme interattive per il coinvolgimento dei cittadini.

Un altro strumento interessante da considerare nell'ottica di ottimizzazione, potrebbe essere la Ricerca Operativa. Essa, può essere utilizzata per migliorare l'efficienza delle operazioni di raccolta e smaltimento dei rifiuti attraverso tecniche di ottimizzazione avanzate. Ad esempio, può aiutare a determinare i percorsi ottimali per i veicoli di raccolta oppure migliorare la distribuzione dei bidoni per la raccolta differenziata, assicurando che siano posizionati nei punti strategici, massimizzando così, l'efficacia e la comodità per i cittadini.

Le opportunità offerte da queste tecnologie sono molteplici e l'ingegno umano, una

delle qualità più importanti in nostro possesso, è ora più che mai necessario per trovare soluzioni innovative che possono trasformare radicalmente il settore, portandoci verso un futuro sostenibile.

Bibliografia

ACKERMANN, P. (2023), *Full Stack Web Development: The Comprehensive Guide*, Sap Pr America.

FARLEY, D. (2021), *Modern Software Engineering: Doing What Works to Build Better Software Faster*, Addison-Wesley Professional.

FLANAGAN, D. (2020), *Javascript: The Definitive Guide: Master the World's Most-Used Programming Language*, O'Reilly Media.

GANDY, M. (2014), *Recycling and the Politics of Urban Waste*, Routledge.

HERNANDEZ, M. J. (2020), *Database Design for Mere Mortals: 25th Anniversary Edition*, Addison-Wesley Professional.

HOORNWEG, D. e BHADA-TATA, P. (2012), «What a waste: a global review of solid waste management», *openknowledge.worldbank.org*.

LARMAN, C. e CABIBBO, L. (2020), *Applicare UML e i pattern. Analisi e progettazione orientata agli oggetti*, Pearson.

MCDONALD, M. (2020), *Web Security for Developers: Real Threats, Practical Defense*, No Starch Press.

NANDA, S. e BERRUTI, F. (2021), «Municipal solid waste management and landfilling technologies: a review», *Environmental chemistry letters*.

PICHTEL, J. (2014), *Waste Management Practices*, CRC Press.

RICHARDS, M. e FORD, N. (2020), *Fundamentals of Software Architecture: An Engineering Approach*, O'Reilly Media.

SCHOTT, A. B. S., ASPEGREN, H., BISSMONT, M. e LA COUR JANSEN, J. (2013), *Modern Solid Waste Management in Practice: The City of Malmö Experience*, Springer.

SOMMERVILLE, I. e MICUCCI, D. (2017), *Introduzione all'ingegneria del software*, Pearson.

STAUFFER, M. (2019), *Laravel: Up & Running: A Framework for Building Modern Php Apps*, O'Reilly Media.

TCHOBANOGLOUS, G. e KREITH, F. (2002), *Handbook of Solid Waste Management*, McGraw-Hill.

WINDFELD, E. e BROOKS, M. (2015), «Medical waste management–A review», *Journal of Environmental Management*.

Sitografia

- Composer, gestore di dipendenze – getcomposer.org
- Documentazione Laravel – www.laravel.com/docs/11.x/readme
- Documentazione Tailwind CSS – www.tailwindcss.com/docs
- Fasda, raccolta porta a porta – www.fasda.it/raccolta-differenziata-porta-a-porta/
- GitHub, servizio di hosting per il progetto software – www.github.com/home
- Marche multiservizi, il portale dell'azienda – www.gruppomarchemultiservizi.it
- Visual Studio Code, IDE – www.visualstudio.com
- XAMPP, ambiente di sviluppo PHP – www.apachefriends.org/it/index.html

Ringraziamenti

Prima di tutto ci tengo a ringraziare i miei genitori, che da sempre mi sostengono in qualsiasi decisione io prenda; so che non è per nulla scontato e averli al mio fianco mi rende infinitamente grato.

Vorrei, poi, ringraziare i miei due fratelli Giordano e Jacopo. Giordy, per avermi fatto da "mentore" sulle dinamiche dell'università e per aver creduto in me, anche quando io smettevo di farlo. Jaco, per essere stato il mio migliore amico, ancor prima che mio fratello; credo che insieme siamo cresciuti tanto.

Una persona molto importante per me e pilastro di questo viaggio è la mia ragazza: Anna. Da quando sei entrata nella mia vita durante il secondo anno di università, mi hai ricordato che sono umano e che non sono uno di quei computer con cui studio o lavoro; mi hai ricordato che la vita va vissuta, va esplorata a fondo e di quanto sia importante non trascurarsi nel mentre. Grazie per essere stata la mia fan numero uno a ogni esame, indipendentemente dal risultato, e per esserti fatta in quattro in ogni situazione per aiutarmi o, semplicemente, vedermi felice. Non vedo l'ora di festeggiare anche il tuo di traguardo, perché so quanto lo meriti.

Un ringraziamento speciale va ai miei fedeli compagni di corso, in particolare a Laura, che è riuscita a farmi superare un inizio poco convincente a causa del Covid. Sia nei progetti, che nello studio ho sempre potuto contare su di te. Inoltre, ringrazio infinitamente il professore Ursino, una delle persone più disponibili che abbia mai conosciuto e che mi ha accompagnato durante tutto il processo di stesura della tesi… nonostante i tempi stretti.

Ringrazio tutti i miei amici di Pesaro; sono stato poco presente per molti di loro e mi stupisce sempre l'affetto che mi dimostrano quando torno, nonostante tutto. Giovi, la tua leggerezza nell'affrontare le cose un po' la trasmetti e mi fa bene.

L'ultima persona che voglio ringraziare, sono io. Non ho mai avuto le risposte chiare come altri, ho convissuto spesso con la paura di non aver fatto le scelte giuste. Ora, però, riguardando la strada che ho fatto, penso che forse questo sia stato un bene, che mi ha portato a dare sempre il massimo e a non fermarmi mai.