SmartWaste

Sistema Intelligente di Gestione Rifiuti Urbani

Studente: Matteo Smerilli
Matricola: 122650
Corso: Progettazione Web
Anno Accademico: 2024/2025
Repository: https://github.com/Smeroo/SmartWaste

---

1. Introduzione

1.1 Contesto e Obiettivi

La gestione dei rifiuti urbani rappresenta una sfida cruciale per le città moderne. **SmartWaste** nasce come soluzione digitale per offrire ai cittadini uno strumento intuitivo per identificare correttamente le modalità di smaltimento dei rifiuti, localizzare i punti di raccolta e segnalare problematiche ambientali in tempo reale.

Obiettivi principali:

- Fornire una guida completa alla raccolta differenziata
- Geolocalizzare punti di raccolta tramite mappe interattive
- Permettere segnalazioni di problematiche ambientali
- Garantire accessibilità su tutti i dispositivi

---

2. Architettura e Stack Tecnologico

2.1 Stack Tecnologico

Frontend:

- **Next.js 14** - Framework React con App Router (SSR/CSR)
- **TypeScript** - Tipizzazione statica
- **TailwindCSS** - Styling responsive
- **Leaflet** - Mappe interattive
- **FontAwesome** - Libreria icone

Backend:

- **Next.js API Routes** - Endpoint REST integrati
- **NextAuth.js** - Autenticazione con OAuth
- **Prisma ORM** - Gestione database type-safe

Database:

- **SQLite** - Sviluppo locale
- **PostgreSQL** - Produzione cloud

2.2 Architettura Applicativa

L'applicazione segue il pattern **MVC (Model-View-Controller)**:

```
Frontend (React Components)
         ↓
API Routes (/api/*)
         ↓
Prisma ORM
         ↓
Database (SQLite/PostgreSQL)
```

2.3 Modello Dati

Entità principali:

- **User** - Utenti con ruoli (USER, OPERATOR, ADMIN)
- **Operator** - Gestori dei punti raccolta
- **CollectionPoint** - Centri di raccolta
- **Address** - Indirizzi georeferenziati
- **WasteType** - Tipologie rifiuti
- **CollectionSchedule** - Orari apertura
- **Report** - Segnalazioni utenti

Relazioni:

- User → Report (1:N)
- Operator → CollectionPoint (1:N)
- CollectionPoint → Address (1:1)
- CollectionPoint → Schedule (1:1)
- CollectionPoint ↔ WasteType (N:M)

---

3. Funzionalità Implementate

3.1 Sistema di Autenticazione

- Registrazione con validazione email
- Login OAuth (Google, GitHub)
- Login tradizionale (email/password con bcrypt)
- Gestione sessioni JWT
- Reset password con token temporanei
- Role-based access control (RBAC)

3.2 Guida ai Rifiuti

- Classificazione completa (Plastica, Carta, Vetro, Organico, Metallo, RAEE)
- Informazioni dettagliate per ogni tipologia
- Esempi pratici di smaltimento

- Codifica colori per riconoscibilità immediata
- Ricerca testuale

3.3 Mappa Interattiva

- Visualizzazione georeferenziata dei punti raccolta
- Marker cliccabili con popup informativi
- Filtri per tipo di rifiuto
- Integrazione Geolocation API
- OpenStreetMap come provider

3.4 Gestione Punti Raccolta

- Elenco con filtri avanzati e ricerca
- Dettaglio completo: indirizzo, orari, contatti, accessibilità
- Informazioni su capacità e carico attuale
- API REST per operazioni CRUD

3.5 Sistema di Segnalazioni

Tipologie:

- Cassonetto pieno/danneggiato/mancante
- Abbandono illecito
- Necessità di pulizia

Stati: PENDING, IN_PROGRESS, RESOLVED, REJECTED

Features:

- Upload immagini
- Tracciamento storico
- Dashboard operatore

---

4. Design e User Experience

4.1 Principi di Design

- **Mobile-First**: Ottimizzazione per dispositivi mobili
- **Accessibilità**: Standard WCAG AA
- **Responsive**: Layout adattivi (breakpoint 640px, 1024px)
- **Feedback Visivo**: Loading states, animazioni fluide

4.2 Colori Distintivi

Ogni tipo di rifiuto ha colore univoco per rapida identificazione: Plastica (giallo), Carta (blu), Vetro (verde), Organico (ambra), Metallo (grigio), RAEE (rosso).

---

5. Sicurezza

Misure implementate:

- Password hashing bcrypt (10 rounds)
- Token JWT firmati con scadenza
- Role-based authorization
- Sanitizzazione input utente
- Protezione SQL Injection (Prisma)
- Variabili d'ambiente isolate (.env)
- HTTPS obbligatorio in produzione
- Cookie HttpOnly e Secure

---

6. Testing e Qualità

Strumenti:

- **ESLint** - Linting codice
- **TypeScript** - Type checking
- **Prettier** - Code formatting
- **Git** - Version control con commit semantici (feat, fix, style, refactor, docs)

---

7. Deployment

7.1 Ambiente Sviluppo

```
npm install
npx prisma migrate dev
npx prisma db seed
npm run dev
```

7.2 Produzione (Pianificato)

- **Hosting**: Vercel
- **Database**: Supabase (PostgreSQL)
- **Storage**: Cloudinary/AWS S3
- **CI/CD**: Deploy automatico su push

---

8. Sviluppi Futuri

Funzionalità pianificate:

- Gamification (punti, badge, classifica)
- Notifiche push e email alerts
- Dashboard analytics per operatori
- Supporto multilingua (i18n)
- PWA con offline support

Ottimizzazioni tecniche:

- React Query per caching
- Lazy loading immagini
- Testing (Jest + React Testing Library)
- Code splitting ottimizzato

---

9. Conclusioni

SmartWaste rappresenta una soluzione full-stack moderna per la gestione intelligente dei rifiuti urbani, dimostrando competenze in:

- [x] Sviluppo full-stack con Next.js 14
- [x] Database relazionali con Prisma
- [x] Autenticazione OAuth sicura
- [x] Design responsive e accessibile
- [x] Integrazione API di terze parti
- [x] Architettura scalabile e manutenibile

L'applicazione combina performance, sicurezza e user experience in un'architettura modulare pronta per crescere, offrendo ai cittadini uno strumento concreto per contribuire alla sostenibilità ambientale.