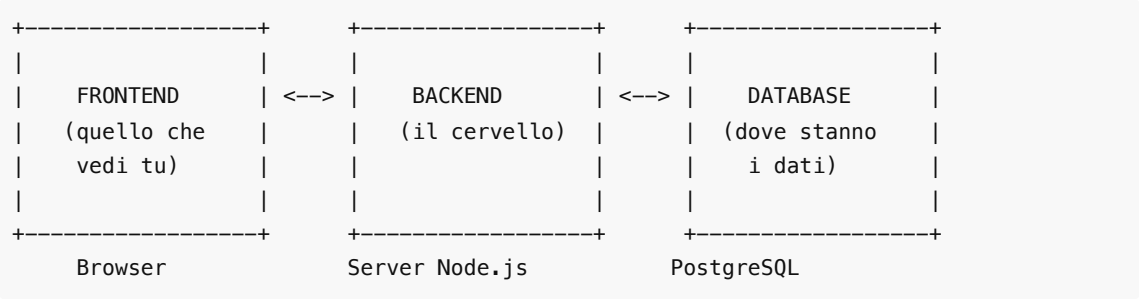
Come Funziona il Gestionale - Guida Semplice

Panoramica

Il gestionale è composto da 3 parti principali che lavorano insieme:



---

1. FRONTEND (Interfaccia Utente)

Cos'è: La parte visibile dell'applicazione - i pulsanti, le tabelle, i form che usi ogni giorno.

Tecnologia: React (una libreria JavaScript moderna)

Dove si trova: Cartella `src/`

- `src/pages/` - Le varie pagine (Dashboard, Prenotazioni, Ospiti, ecc.)
- `src/components/` - I componenti riutilizzabili (pulsanti, card, tabelle)

Come funziona:

1. Apri il browser e vai all'indirizzo del gestionale
2. Il browser scarica l'applicazione React
3. Quando clicchi su qualcosa (es. "Nuova Prenotazione"), React chiede i dati al Backend

---

2. BACKEND (Server)

Cos'è: Il "cervello" dell'applicazione. Gestisce la logica, la sicurezza, e comunica con il database.

Tecnologia: Node.js con Express

Dove si trova: Cartella `server/`

- `server/controllers/` - La logica (cosa fare quando arriva una richiesta)
- `server/routes/` - Gli "indirizzi" delle API (es. `/api/bookings`)
- `server/middleware/` - Controlli di sicurezza (autenticazione, permessi)

Come funziona:

1. Riceve richieste dal Frontend (es. "dammi tutte le prenotazioni")
2. Verifica che l'utente sia autorizzato
3. Chiede i dati al Database
4. Risponde al Frontend con i dati

Esempio di flusso:

```
Frontend: "Voglio vedere le prenotazioni di gennaio"
    ↓
Backend: "Ok, sei autorizzato. Chiedo al database..."
    ↓
Database: "Ecco le 15 prenotazioni di gennaio"
    ↓
Backend: "Ecco i dati al frontend"
    ↓
Frontend: "Li mostro nella tabella"
```

---

3. DATABASE (Dove Stanno i Dati)

Cos'è: Il "magazzino" dove vengono salvati tutti i dati in modo permanente.

Tecnologia: PostgreSQL (un database relazionale molto affidabile)

Dove si trova:

- **Sviluppo locale:** Neon.tech (database cloud gratuito)
- **Produzione (Render):** PostgreSQL di Render

Cosa contiene:

- Utenti e credenziali
- Proprietà (case vacanza)
- Prenotazioni
- Ospiti e documenti
- Costi fissi
- Prodotti e inventario
- Impostazioni

Schema semplificato:

```
UTENTE
  ├── Proprietà (case)
  │     └── Prenotazioni
  │           └── Ospiti
  ├── Costi Fissi
  ├── Prodotti
  └── Impostazioni
```

---

4. AMBIENTI: Sviluppo vs Produzione

Sviluppo (quando lavori sul codice)

```
Tu (localhost:3000) → Server locale (localhost:5001) → Database Neon.tech
```

- Usi `npm run dev` per avviare tutto
- I dati vanno nel database Neon (cloud)
- Modifichi il codice e vedi le modifiche subito

Produzione (quando il cliente usa l'app)

```
Cliente (gestionale-case-vacanza.onrender.com) → Server Render → Database Render
```

- Render ospita sia il server che il database
- I dati di produzione sono separati da quelli di sviluppo
- Ogni push su GitHub fa partire un nuovo deploy

---

5. FLUSSO COMPLETO: Esempio Pratico

Scenario: Un utente crea una nuova prenotazione

```
1. UTENTE clicca "Nuova Prenotazione" nel browser

2. FRONTEND (React)
   - Mostra il form
   - Utente compila i campi
   - Utente clicca "Salva"
   - React invia i dati al backend

3. BACKEND (Node.js)
   - Riceve: POST /api/bookings con i dati
   - Verifica: "L'utente è loggato? Ha i permessi?"
   - Valida: "I dati sono corretti? Date valide?"
   - Salva: Crea la prenotazione nel database

4. DATABASE (PostgreSQL)
   - Riceve il comando di inserimento
   - Salva i dati nella tabella "Booking"
   - Conferma il salvataggio

5. RITORNO
   - Database → Backend: "Fatto, ecco l'ID della prenotazione"
   - Backend → Frontend: "Prenotazione creata con successo"
   - Frontend → Utente: Mostra messaggio di conferma
```

---

6. SICUREZZA

Autenticazione (Chi sei?)

- Quando fai login, il backend crea un "token" (JWT)
- Ogni richiesta include questo token
- Il backend verifica il token prima di rispondere

Autorizzazione (Cosa puoi fare?)

- Ogni utente vede solo i SUOI dati
- Le query al database filtrano sempre per `userId`

---

7. DOVE SONO I FILE IMPORTANTI

```
gestionale-react/
├── src/                    # FRONTEND
│   ├── pages/              # Pagine dell'app
│   ├── components/         # Componenti riutilizzabili
│   ├── api/                # Chiamate al backend
│   └── contexts/           # Stato globale (utente loggato, ecc.)
│
├── server/                 # BACKEND
│   ├── controllers/        # Logica delle operazioni
│   ├── routes/             # Definizione delle API
│   ├── middleware/         # Autenticazione, validazione
│   └── config/             # Configurazione database
│
├── prisma/                 # DATABASE
│   ├── schema.prisma       # Struttura delle tabelle
│   └── seed.js             # Dati iniziali
│
├── .env                    # Configurazione (credenziali, URL database)
└── render.yaml             # Configurazione per il deploy su Render
```

---

8. COMANDI UTILI

Comando	Cosa fa
`npm run dev`	Avvia frontend + backend in sviluppo
`npm run build`	Crea la versione di produzione
`npx prisma studio`	Apre un'interfaccia per vedere il database
`npx prisma db push`	Applica modifiche allo schema del database
`npx prisma db seed`	Inserisce i dati iniziali (categorie, ecc.)
`git push origin main`	Pubblica le modifiche (avvia deploy su Render)

---

9. RIEPILOGO

Componente	Tecnologia	Dove
Frontend	React	Browser dell'utente
Backend	Node.js + Express	Server (Render in prod)
Database	PostgreSQL	Neon (dev) / Render (prod)
Hosting	Render.com	Cloud
Codice	GitHub	Repository online

Domande Frequenti

D: Se modifico qualcosa, quando lo vede il cliente? R: Devi fare `git push`. Render rileva il push e fa un nuovo deploy (circa 5 minuti).

D: I dati di sviluppo e produzione sono gli stessi? R: No, sono database separati. Sviluppo usa Neon, produzione usa il database di Render.

D: Cosa succede se il server si spegne? R: I dati nel database rimangono. Quando il server riparte, tutto funziona di nuovo.

D: Posso vedere i dati direttamente nel database? R: Sì, con `npx prisma studio` in sviluppo, oppure dalla dashboard di Neon/Render.