Relazione Tecnica sul Progetto "BeachBites"

MATTEO PIO D'APOLITO
Università degli studi di Modena e Reggio Emilia
Matricola N. 164904

Contents

1 Il progetto

Il progetto "BeachBites" è stato concepito come un servizio SaaS (Software as a Service) per la gestione delle ordinazioni in spiaggia. Di seguito vengono riportate in maniera dettagliata le funzionalità richieste:

1.1 Funzionalità Pubbliche per i Clienti

- **Ordinazione Prodotti**: I clienti possono visualizzare un elenco di prodotti disponibili e ordinare cibo e bevande. Devono inserire i dettagli essenziali come nome, cognome, ora di consegna e numero dell'ombrellone.

1.2 Funzionalità Private per il Gestore dello Stabilimento

- **Gestione Ordini**: Visualizzare in tempo reale gli ordini effettuati dai clienti.
- **Aggiunta Prodotti**: Inserire nuovi prodotti, aggiornare o eliminare quelli esistenti.
- **Gestione Categorie**: Creare e gestire categorie di prodotti per una migliore organizzazione del menu.
- **Gestione Allergenici**: Aggiungere e gestire una lista di allergeni per ciascun prodotto per garantire la sicurezza alimentare dei clienti.
- **Impostazioni**: Configurare le informazioni di contatto, orari di apertura e chiusura dello stabilimento.
- **Geolocalizzazione**: Definire una lista di coordinate GPS che identificano l'area dello stabilimento, impedendo gli ordini al di fuori di questa zona tramite rilevazione GPS.

2 Descrizione del progetto dell'applicazione

2.1 Use Case UML

Descrizione dei vari casi d'uso tramite diagrammi UML per rappresentare le varie funzionalità a seconda della tipologia di utente.

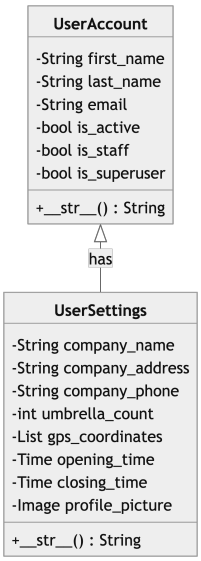

Figure 1: Diagramma UML Autenticazione

2.2 Diagramma delle Classi

Rappresentazione dettagliata delle entità coinvolte e delle loro relazioni tramite un diagramma delle classi.

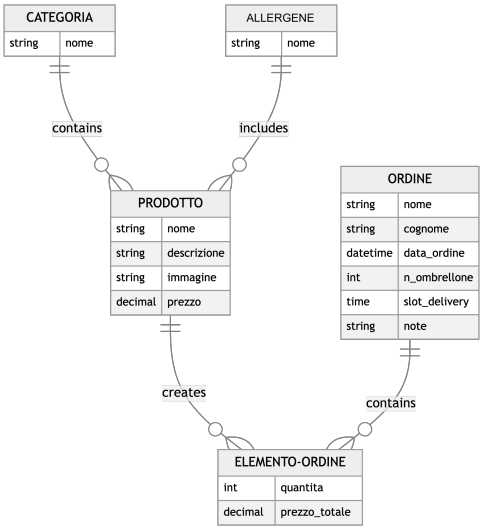

Figure 2: Diagramma ER Ordini

2.3 Activity Diagram

Diagrammi delle attività per spiegare i principali passaggi logici coinvolti.

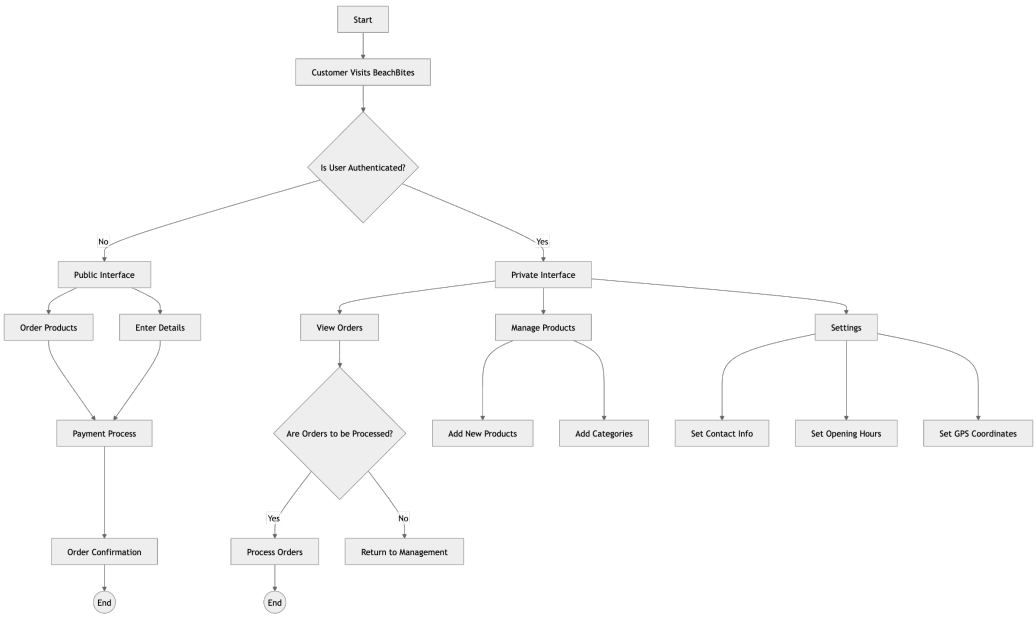

Figure 3: Activity Diagram - percorso utenti

3 Stack Tecnologico

Il progetto **BeachBites** è stato sviluppato utilizzando una combinazione di tecnologie moderne e consolidate per garantire robustezza, scalabilità e un'esperienza utente ottimale. Questa sezione descrive in dettaglio lo stack tecnologico adottato per il backend e il frontend del progetto.

3.1 Backend

Il backend del progetto **BeachBites** è stato realizzato con l'obiettivo di garantire un'architettura solida e facilmente manutenibile. Le tecnologie principali utilizzate sono:

- **Linguaggio di Programmazione**: Python.
  - Python è stato scelto per la sua semplicità, leggibilità del codice e la vasta gamma di librerie disponibili che accelerano lo sviluppo.

- **Framework Web**: Django.
  - Django è un framework ad alto livello per lo sviluppo rapido di applicazioni web, che include una serie di strumenti utili per l'autenticazione, la gestione dei database e la sicurezza.
- **Database**: PostgreSQL.
  - PostgreSQL è un sistema di gestione di database relazionale (RDBMS) conosciuto per la sua affidabilità, integrità dei dati e supporto per transazioni complesse. È stato scelto per l'ambiente di produzione.
  - Durante lo sviluppo, è stato utilizzato il database `sqlite3` integrato in Django per la sua semplicità di configurazione.
- **Autenticazione**: JWT con `djoser`.
  - `djoser` è una libreria che fornisce endpoint per le operazioni comuni di autenticazione quali registrazione, login, recupero password e altro. L'autenticazione JWT garantisce sessioni sicure e senza stato.
- **Servizi Email**: AWS SES.
  - Amazon Simple Email Service (SES) è stato utilizzato per inviare email di conferma registrazione agli utenti, sfruttando le capacità scalabili e affidabili di AWS.

3.2 Frontend

Il frontend del progetto **BeachBites** è stato costruito utilizzando le seguenti tecnologie per garantire un'interfaccia utente moderna e reattiva:

- **Libreria di Frontend**: React.
  - React è una libreria JavaScript popolare per la costruzione di interfacce utente, scelta per la sua capacità di creare componenti riutilizzabili e l'aggiornamento efficiente del DOM.
- **Framework**: Next.js 14.
  - Next.js è un framework per React che offre funzionalità come rendering lato server (SSR), rendering statico (SSG) e un sistema di routing semplice ed efficace. Questo permette di migliorare le performance e l'ottimizzazione SEO dell'applicazione.

- **Gestione dello Stato**: React Hook Form.
  - React Hook Form è utilizzato per la gestione dei form, offrendo una gestione dello stato leggera e performante specialmente per i moduli di input degli utenti.
- **Validazione dei Dati**: Zod.
  - Zod è una libreria per la validazione e parsing degli schemi che assicura che i dati degli input siano corretti e sicuri prima di essere inviati al backend.
- **Stili**: Tailwind CSS e `shadcn/ui`.
  - Tailwind CSS è un framework di utility-first CSS che permette di creare design complessi applicando classi direttamente nei componenti. Tale approccio facilita la manutenzione del CSS e consente una rapida prototipazione.
  - `shadcn/ui` è stata utilizzata per migliorare l'interazione dell'interfaccia utente, fornendo componenti UI pronti all'uso e personalizzabili.

3.3 Geolocalizzazione

Per garantire che gli ordini possano essere effettuati solo all'interno dell'area dello stabilimento balneare, è stata implementata la funzionalità di geolocalizzazione:

- **Servizio di Geolocalizzazione GPS**:
  - Le coordinate GPS vengono utilizzate per determinare se l'utente si trova all'interno dell'area di servizio definita dal gestore. Questa funzionalità aggiunge un ulteriore livello di controllo e sicurezza per il servizio offerto.

4 Organizzazione Logica dell'Applicazione

L'applicazione **BeachBites** è stata progettata con un'architettura modulare e ben strutturata per favorire la manutenibilità, la scalabilità e la chiarezza del codice. Questa sezione descrive la suddivisione logica del codice e le motivazioni alla base delle scelte architetturali.

4.1 Organizzazione logica

Il framework Django permette una suddivisione del progetto in app, ciascuna delle quali incapsula funzionalità specifiche. In **BeachBites**, sono state create due app principali: `authentication` e `ordini`. Ogni app è responsabile di un insieme di funzionalità coerenti e partecipa alla gestione generale del sistema.

4.2 App `authentication`

L'app `authentication` è dedicata alla gestione dell'autenticazione e degli utenti. Essa si avvale della libreria `djoser` per facilitare le operazioni di registrazione, login e gestione delle password e contiene un modello che gestisce le informazioni dello stabilimento balneare. Le principali responsabilità dell'app `authentication` sono:

- Implementare i meccanismi di autenticazione tramite JWT (JSON Web Tokens).
- Gestire la registrazione degli utenti e l'invio di email di conferma tramite AWS SES.
- Mantenere i dati relativi ai parametri dello stabilimento, come le informazioni di contatto, orari di apertura e chiusura, e coordinate GPS.

La scelta di segregare l'autenticazione in una propria app distintiva è motivata dalla necessità di mantenere il codice organizzato e facilmente manutenibile, separando le concern specifiche dell'autenticazione da altre funzionalità del sistema.

4.3 App `ordini`

L'app `ordini` è responsabile della gestione dei prodotti e degli ordini. Questa suddivisione permette di isolare i processi di business associati alla gestione degli ordini e della catalogazione dei prodotti, facilitando le operazioni CRUD (Create, Read, Update, Delete). Le principali responsabilità dell'app `ordini` sono:

- Gestire i modelli che rappresentano i prodotti, le categorie e gli allergeni.
- Implementare le operazioni CRUD per la gestione dei prodotti, inclusa l'aggiunta, modifica, visualizzazione e cancellazione di articoli.

- Gestire gli ordini dei clienti, che includono dettagli quali nome, cognome, ora di consegna e numero dell'ombrellone.
- Fornire un'interfaccia per il gestore dello stabilimento per visualizzare gli ordini e aggiornarli in tempo reale.

La suddivisione in due app (`authentication` e `ordini`) è strategica per mantenere una chiara separazione delle responsabilità. Questa struttura modulare permette anche una maggiore facilità di test, manutenibilità e potenziale riusabilità delle app in altri progetti futuri.

4.4 Scelte Particolari

- **Utilizzo di JWT per l'Autenticazione**:
  - L'uso di JSON Web Tokens (JWT) per l'autenticazione è stato scelto per la sua capacità di creare sessioni senza stato, che si adatta bene a un'architettura scalabile e distribuita. JWT facilita anche l'integrazione con il frontend basato su React.
- **Invio di Email tramite AWS SES**:
  - Amazon Simple Email Service (SES) è stato selezionato per l'invio di email di conferma della registrazione e altre notifiche. Questo servizio cloud scalabile e affidabile garantisce che le email vengano consegnate tempestivamente e senza problemi di deliverability.
- **Geolocalizzazione per Controllo della Zona di Servizio**:
  - L'implementazione della verifica tramite coordinate GPS garantisce che gli ordini possano essere effettuati solo entro i confini dello stabilimento balneare. Questa funzionalità sfrutta la geolocalizzazione dei dispositivi mobili degli utenti, migliorando la precisione del servizio.
- **Frontend con React e Next.js 14**:
  - La scelta di React in combinazione con Next.js 14 permette un rendering efficiente e un miglioramento delle performance grazie al rendering lato server (SSR) e alla generazione di pagine statiche (SSG). Questo approccio garantisce un'esperienza utente fluida e veloce.

In sintesi, l'organizzazione logica dell'applicazione **BeachBites** è stata progettata per garantire modularità, facilità di manutenzione e una separazione chiara delle responsabilità. Le scelte tecnologiche e architetturali sono state fatte per bilanciare la semplicità di sviluppo con la necessità di robustezza e scalabilità.

5 Testing

Per garantire la qualità e la robustezza del software sviluppato, è stata eseguita una serie di test mirati principalmente sul backend del progetto **BeachBites**. L'utilizzo di test automatizzati consente di identificare rapidamente eventuali regressioni o bug e di assicurare che il sistema rispetti i requisiti funzionali e non funzionali.

5.1 Test di Backend

Il backend sviluppato con Django è stato sottoposto a diversi tipi di test, descritti brevemente di seguito:

5.1.1 Test Unitari

I test unitari sono stati utilizzati per verificare il corretto funzionamento delle singole unità di codice, come funzioni e metodi. Per l'implementazione dei test unitari, è stata utilizzata la libreria `unittest`. Alcuni esempi di test unitari eseguiti includono:

- **Test delle Funzioni di Modello**:
  - Test delle funzioni di modello per la corretta creazione, aggiornamento e cancellazione degli oggetti nel database.
- **Test delle Funzioni di Utilità**:
  - Test delle funzioni di utilità per la validazione dei dati e la gestione delle eccezioni.

5.1.2 Test di Integrazione

I test di integrazione sono stati utilizzati per verificare che diversi componenti del sistema funzionino correttamente insieme. Alcuni esempi di test di integrazione eseguiti includono:

- **Test delle API REST**:
  - Test degli endpoint delle API per assicurare che le richieste HTTP (GET, POST, PUT, DELETE) restituiscano le risposte attese. Sono stati utilizzati strumenti come `Postman` per facilitare questi test.
- **Test di Autenticazione e Autorizzazione**:
  - Test dei flussi di autenticazione tramite JWT, inclusi registrazione, login, e accesso a risorse protette.
- **Test delle Interazioni con il Database**:
  - Test delle operazioni di lettura e scrittura nel database PostgreSQL per assicurare l'integrità dei dati.

5.1.3 Test Funzionali

I test funzionali sono stati condotti per verificare che il sistema soddisfi i requisiti funzionali specificati. Alcuni esempi di test funzionali eseguiti includono:

- **Test del Processo di Ordine**:
  - Verifica del corretto funzionamento del processo di ordinazione e pagamento dei prodotti, inclusa la gestione delle informazioni dell'utente (nome, cognome, ora di consegna, numero ombrellone).
- **Test dei Flussi di Email**:
  - Verifica della corretta inviazione delle email di attivazione account tramite AWS SES.

5.1.4 Test di Performance

Nonostante la loro importanza, i test di performance sono stati implementati in modo limitato dato il contesto accademico del progetto. Tuttavia, alcuni test di base sono stati eseguiti per assicurare che l'applicazione rispondesse adeguatamente sotto carico. Esempi di test di performance includono:

- **Test di Carico degli Endpoint**:
  - Verifica del tempo di risposta degli endpoint principali sotto carico simulato.

In conclusione, l'adozione di una strategia di test completa per il backend ha permesso di garantire un elevato livello di qualità del software, minimizzando il rischio di bug e regressioni.

6 Conclusioni

Al termine dello sviluppo, il progetto **BeachBites** è stato pubblicato online per una dimostrazione live. Questo permette agli utenti e agli stakeholder di interagire con l'applicazione in modo diretto e di testare tutte le funzionalità implementate.

6.1 Demo Live

La versione di dimostrazione live del progetto è accessibile tramite il link presente nel README della repository. Questa demo live è stata configurata per mostrare tutte le funzionalità chiave del progetto, sia nella parte pubblica riservata ai clienti che nella parte privata per il gestore dello stabilimento.

6.2 Repository GitHub

Per facilitare ulteriormente l'accesso al codice sorgente e alla documentazione dettagliata del progetto, **BeachBites** è stato pubblicato su GitHub. Nella repository sono presenti tutte le istruzioni necessarie per clonare il progetto, configurare l'ambiente di sviluppo e avviare l'applicazione localmente.

La repository GitHub può essere trovata al seguente link:

`https://github.com/matteodapolito/beachbites`

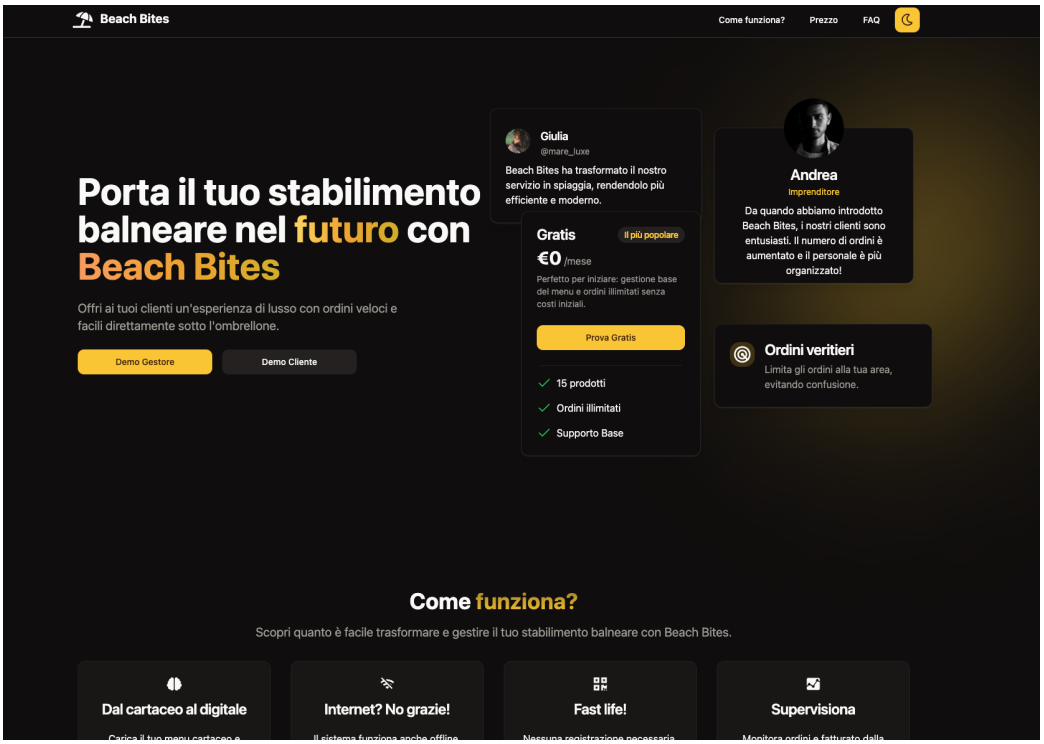

Figure 4: Landing Page del servizio

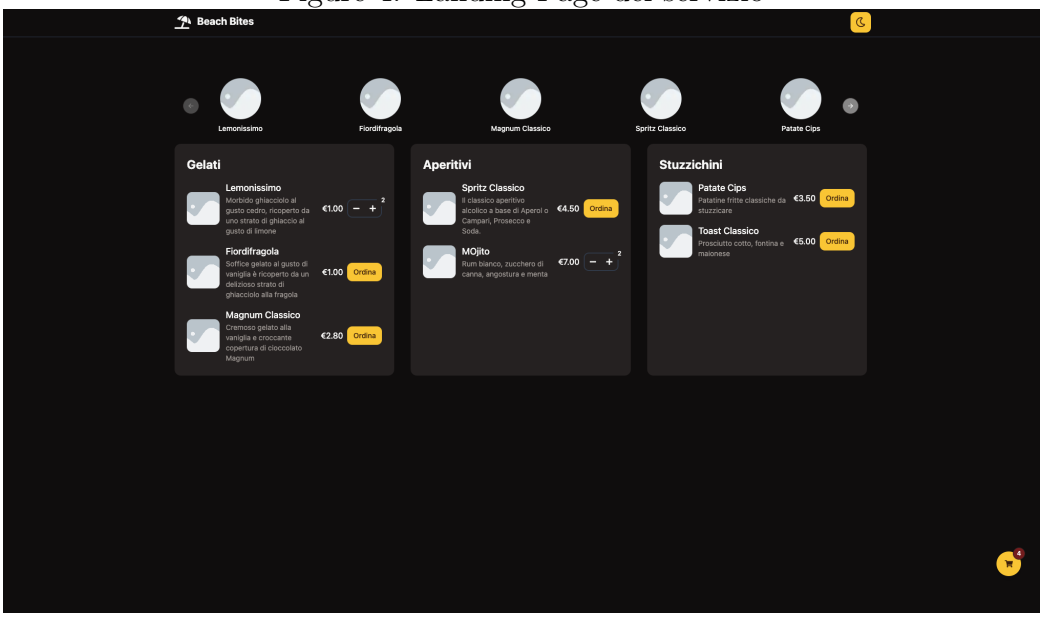

Figure 5: Pagina Ordini per clienti

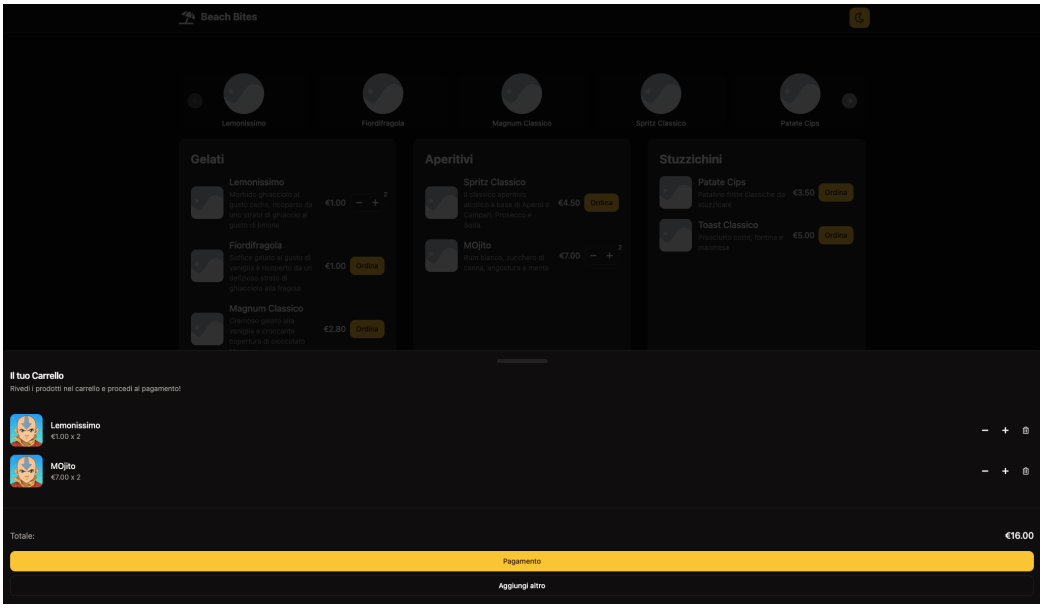

Figure 6: Drawer carrello

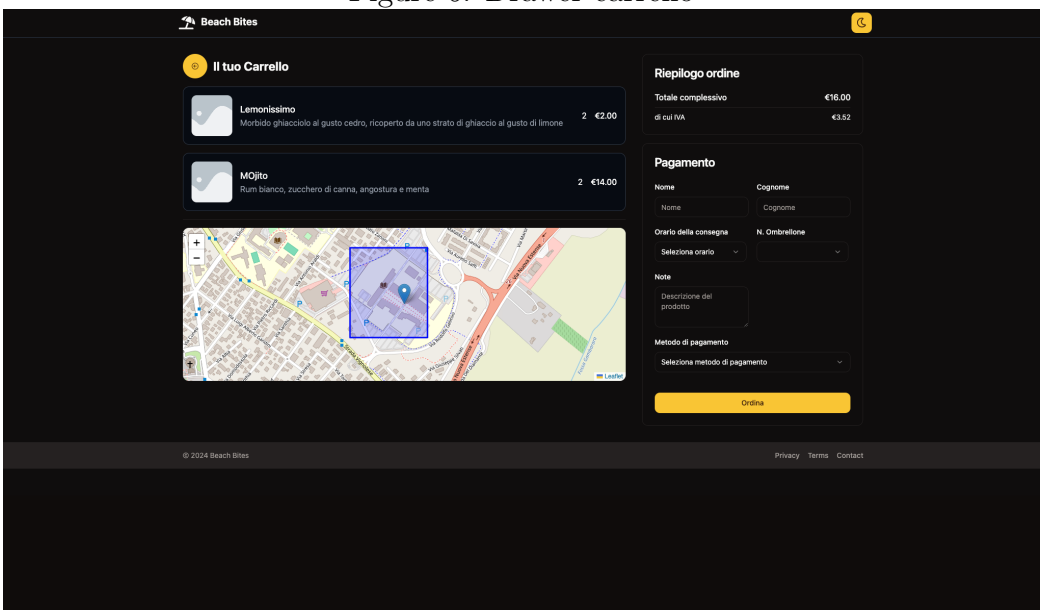

Figure 7: Pagina Checkout

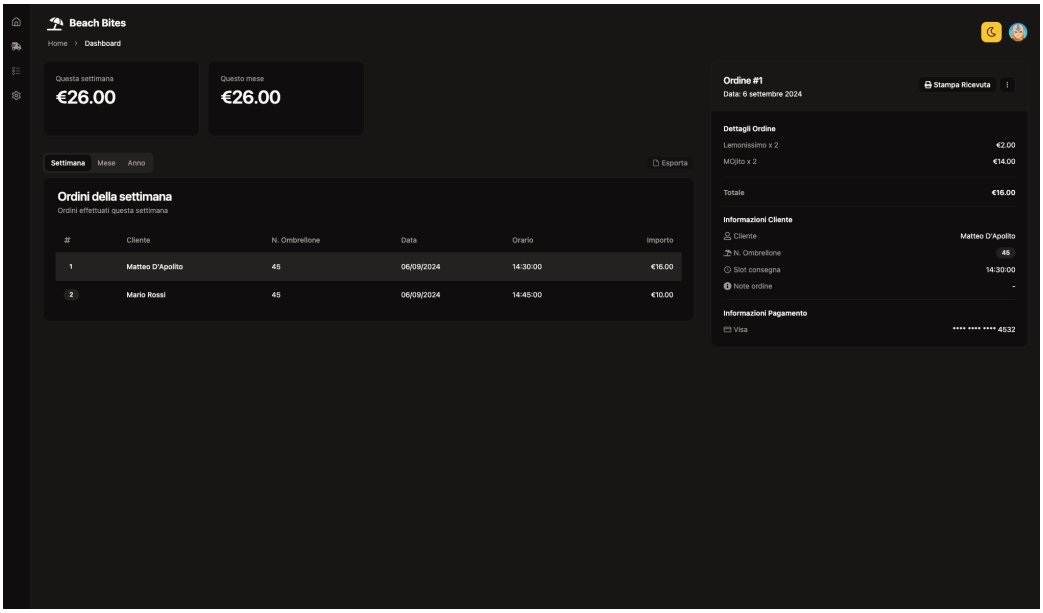

Figure 8: Dashboard ordini ricevuti

Figure 9: Stampa ricevuta

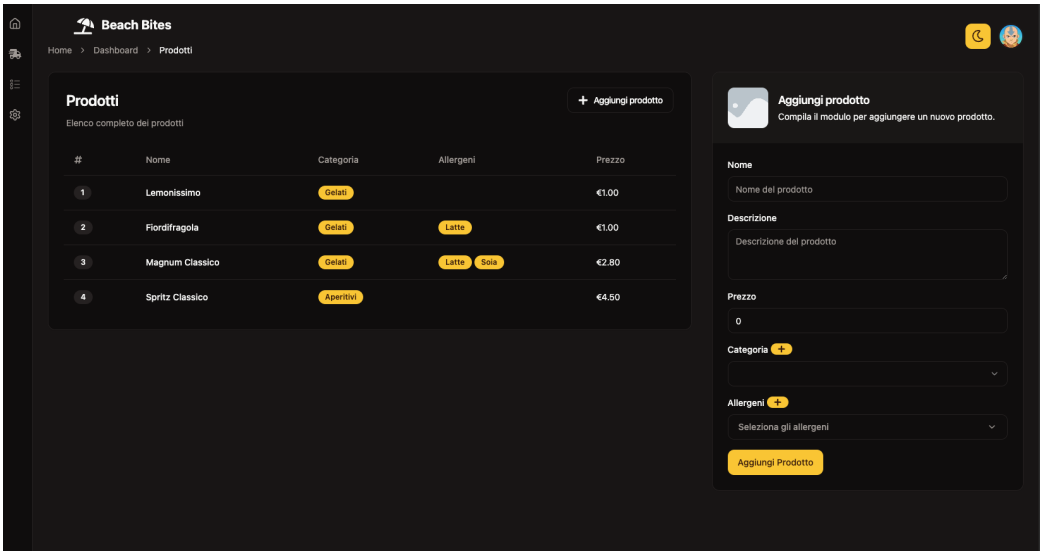

Figure 10: Dashboard prodotti

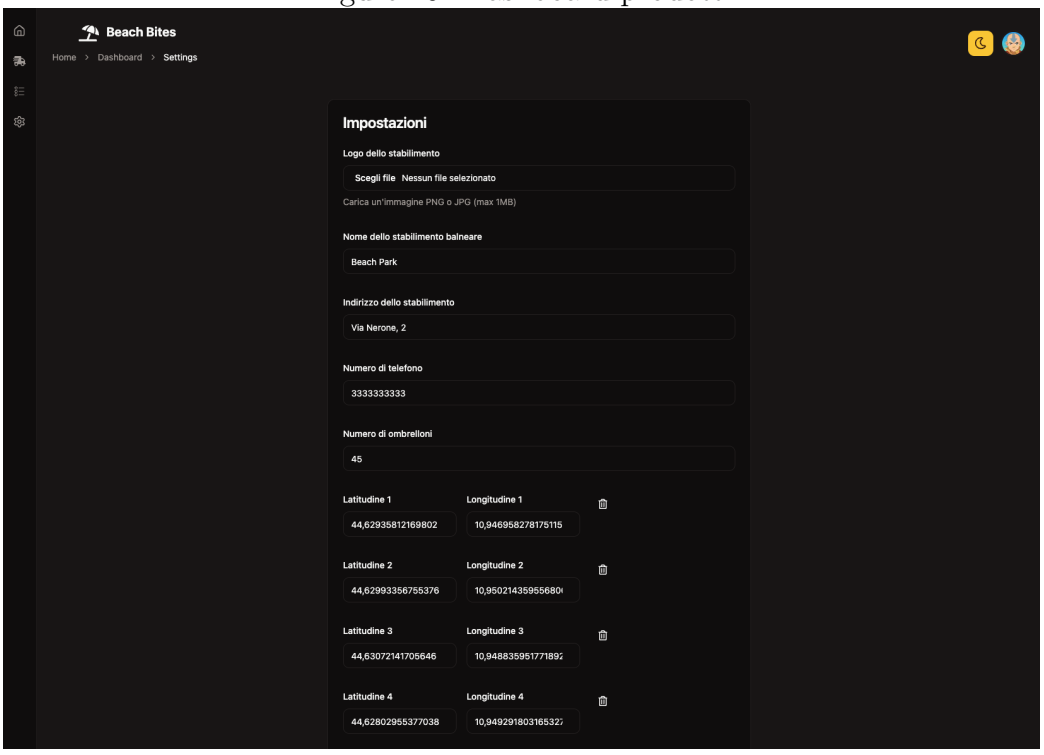

Figure 11: Dashboard impostazioni