UNIVERSITÀ DEGLI STUDI DI SALERNO

DIPARTIMENTO DI INFORMATICA

Impact Analysis Evently

Progetto di Ingegneria, Gestione ed Evoluzione del Software

Giulio De Pascale Matr.0522502049
Nicoletta Mauro Matr.0522502072

Contents

1 Contesto del Progetto

In un contesto in continua evoluzione, la capacità di rispondere in modo rapido ed efficace ai cambiamenti è fondamentale per mantenere la qualità del prodotto e soddisfare le aspettative degli utenti.
In quest'ottica si inserisce **Evently**, una piattaforma web progettata per supportare la creazione, la gestione e la promozione di eventi di diversa natura. L'applicazione offre agli organizzatori strumenti intuitivi per pubblicare eventi, gestire prenotazioni e vendere biglietti, mentre agli utenti finali consente di scoprire facilmente nuove attività, effettuare acquisti e interagire con le organizzazioni di loro interesse. Grazie alla sua architettura modulare e scalabile, Evently mira a fornire un ambiente dinamico e personalizzato che favorisca il coinvolgimento della community e semplifichi i processi legati alla pubblicizzazione di eventi.
Grazie alla sua architettura modulare e scalabile, Evently mira a fornire un ambiente dinamico e personalizzato che favorisca il coinvolgimento della community e semplifichi i processi legati all'organizzazione di eventi. Evently connette organizzatori e partecipanti, offrendo funzionalità quali la creazione di eventi, la gestione di organizzazioni, la vendita di biglietti e la prenotazione di posti. Il sistema si basa su un'architettura **3-Tier** composta da un livello di presentazione, uno di logica applicativa e uno di gestione dei dati, con l'integrazione di servizi esterni per i pagamenti (**Stripe**) e per l'invio di email di autenticazione (**Resend**).
L'idea alla base del progetto è di potenziare Evently introducendo nuove funzionalità che migliorino l'esperienza utente e incentivino il coinvolgimento degli iscritti.

2 Obiettivi del Progetto

Il progetto mira a estendere la piattaforma Evently mediante l'implementazione delle funzionalità di **Organizzazioni Preferite** e **Notifiche**, configurandosi come un intervento di **enhancements**. Questa attività non si limita alla semplice aggiunta di nuove feature, ma si basa su un'analisi approfondita del sistema esistente per garantire che le modifiche siano integrate in modo coerente e sicuro.
Il modello di processo adottato rientra nell'approccio di **enhancements per l'aggiunta di requisiti funzionali**, poiché le modifiche riguardano specifici moduli e funzionalità del sistema. In questo modo è stato possibile concentrare gli sforzi sul miglioramento mirato dell'esperienza utente, minimizzando i rischi e ottimizzando tempi e risorse.

La **Change Request** proposta si concentra su due elementi chiave:

1. La possibilità per gli utenti di seguire le organizzazioni e aggiungerle ai propri preferiti, ricevendo aggiornamenti mirati sulle attività di loro interesse.

2. L'integrazione di un sistema di notifiche interne, che consenta di informare l'utente in tempo reale su eventi rilevanti come pubblicazioni di nuovi eventi, acquisti di biglietti e conferme di prenotazioni.

L'aggiunta di queste funzionalità rappresenta una **scelta strategica** volta a migliorare l'interazione tra utenti e organizzatori, creando un ecosistema dinamico e orientato alla fidelizzazione. La possibilità di seguire organizzazioni consente agli utenti di personalizzare la propria esperienza, rafforzando il legame con le realtà di maggiore interesse, mentre il sistema di notifiche centralizzato offre un canale di comunicazione diretto e immediato, riducendo la dipendenza da strumenti esterni.

Queste modifiche rientrano in una **visione strategica di crescita della piattaforma**, che mira a:

- Offrire agli utenti un'esperienza più ricca, personalizzata e interattiva, migliorandone la soddisfazione e la probabilità di ritorno.
- Fornire agli organizzatori strumenti per mantenere un rapporto più stretto e costante con il proprio pubblico, con effetti positivi sull'engagement e sulla promozione degli eventi.
- Consolidare la posizione di Evently come piattaforma competitiva e moderna, capace di rispondere alle esigenze di un mercato in continua evoluzione.

In sintesi, l'estensione proposta costituisce un **investimento strategico** nella crescita a lungo termine di Evently, combinando l'analisi del sistema esistente con l'introduzione di nuove funzionalità in modo mirato, sicuro e coerente.

3 Analisi del sistema

Prima di procedere con l'implementazione delle nuove funzionalità, è stato necessario condurre un'analisi dettagliata del sistema Evently. L'attività ha previsto una fase di **reverse engineering**, volta a ricostruire a partire dal codice esistente la struttura logica ed architetturale della piattaforma. Questo approccio ha permesso di ottenere una rappresentazione ad un livello di astrazione più alto, individuando i principali moduli, le interazioni tra essi e le logiche di business sottostanti. Tale passaggio si è rivelato fondamentale per garantire una piena comprensione del funzionamento complessivo del sistema.

3.1 Diagramma Entità Relazione

A partire dal codice e dallo schema del database definito in `prisma/schema.prisma`, è stato ricavato il diagramma Entità-Relazione del sistema. Questo passaggio ha permesso di rappresentare in maniera chiara le entità principali e le loro relazioni, dando una visione complessiva della struttura dei dati. Il diagramma ER ha permesso di comprendere come le informazioni vengono archiviate e collegate tra loro, e risulta particolarmente utile per valutare l'impatto delle modifiche richieste dalle Change Requests.

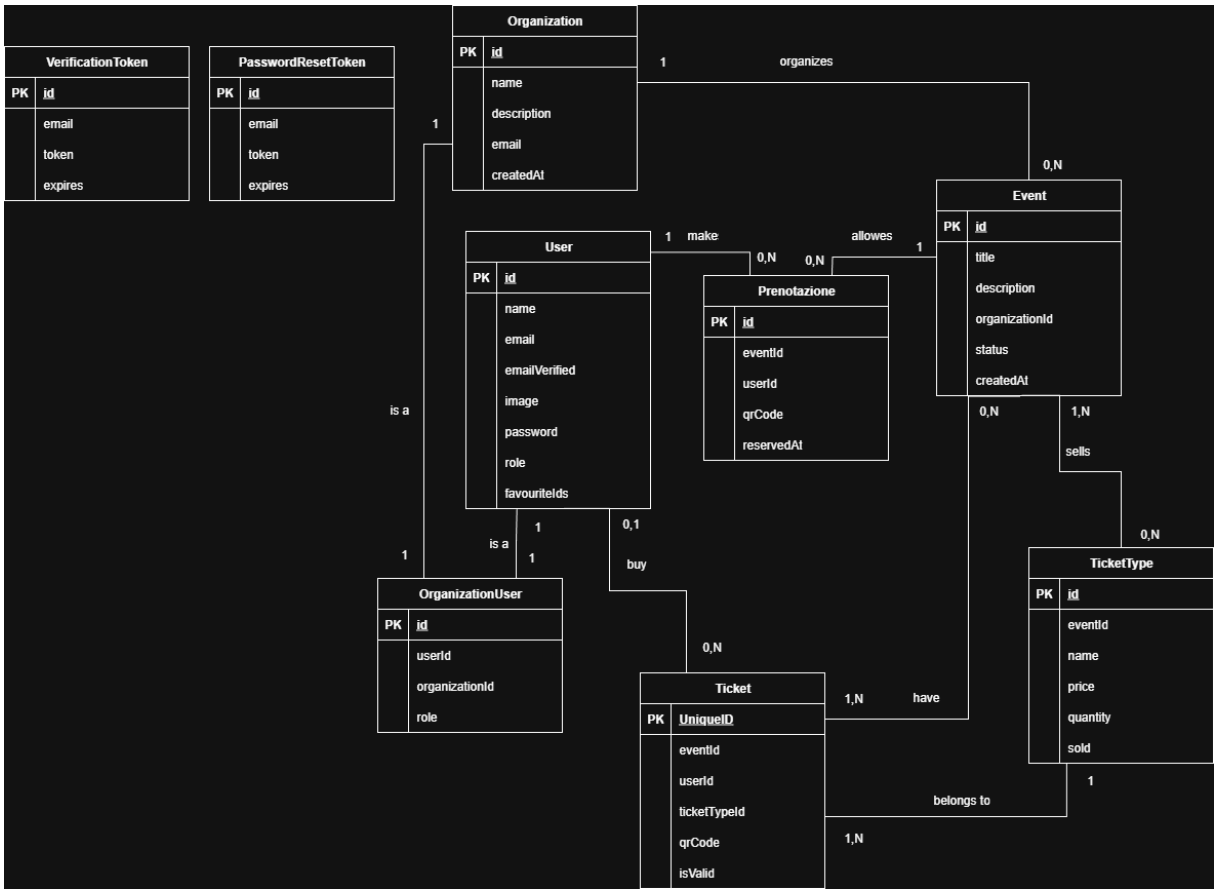

Figure 1: Diagramma Entità-Relazione Evently

.

3.2 Architettura delle componenti

Inizialmente, è stato realizzato un **Component Diagram** della piattaforma, che ha consentito di visualizzare i principali moduli e le loro interconnessioni a un livello di astrazione superiore rispetto al codice sorgente, permettendo di isolare soltanto le informazioni necessarie per l'analisi. L'analisi dei componenti ha fornito le basi per introdurre le nuove funzionalità in maniera coerente con l'architettura esistente, garantendo una facile integrazione con i moduli già presenti.

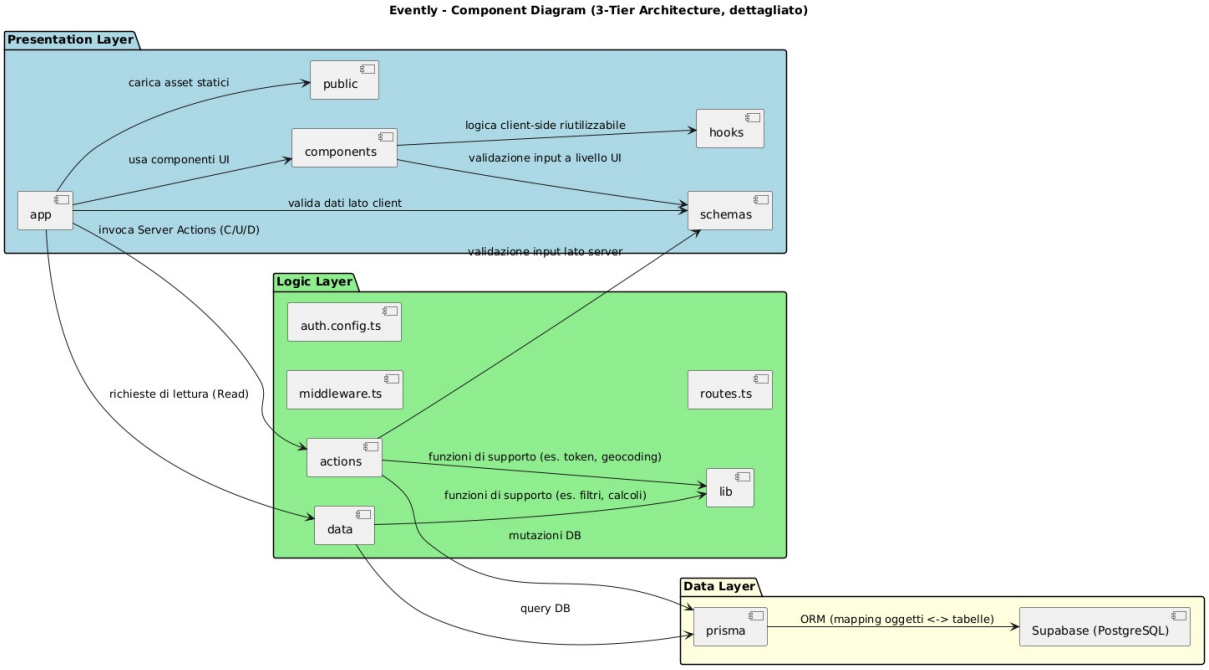

Figure 2: Component Diagram di Evently

.

3.3 Analisi delle dipendenze

A seguire, è stata analizzata la **Matrice delle dipendenze** raffigurata nell'immagine sottostante. La quale mostra il legame fra l'utilizzo di una funzionalità di un modulo in un altro, inserendo nella cella in numero di volte in cui ciò avviene.

Modulo	app	actions	data	components	hooks	lib	prisma	public	schemas	middleware	routes-auth
app	5	2	23	54	2	29	0	0	0	0	0
actions	0	0	11	0	0	21	0	0	7	0	1
data	0	0	0	0	0	8	0	0	0	0	0
components	0	11	1	111	3	13	0	0	8	0	0
hooks	0	1	0	0	0	0	0	0	0	0	0
lib	0	0	4	0	0	2	0	0	0	0	0
prisma	0	0	0	0	0	0	0	0	0	0	0
public	0	0	0	0	0	0	0	0	0	0	0
schemas	0	0	0	0	0	0	0	0	0	0	0
middleware	0	0	0	0	0	0	0	0	0	0	0
routes-auth	0	0	1	0	0	0	0	0	1	0	0

Table 1: Matrice delle dipendenze (conteggi import) per Evently a livello di cartelle.

Dall'analisi della matrice delle dipendenze del sistema emergono alcuni aspetti chiave:

- La cartella **app/** si conferma il fulcro del Presentation Layer, coordinando la composizione delle pagine, il montaggio dei componenti UI e l'interazione con i dati tramite query e utility.
- Il modulo **components/** è molto interconnesso, con numerose auto-dipendenze tipiche di una libreria di componenti riutilizzabili, e richiama direttamente alcune server actions, segno di una forte integrazione con la logica lato server.
- **actions/** svolge un ruolo centrale nella gestione delle operazioni lato server, appoggiandosi a utility condivise, funzioni di lettura già definite e schemi di validazione per mantenere consistenza nei dati.
- La cartella **data/** è volutamente semplice, con dipendenze quasi esclusivamente da **lib/**, che centralizza l'accesso al database attraverso Prisma.
- **lib/**, pur essendo snello, è un nodo strategico: concentra funzioni condivise e la configurazione del database, fungendo da ponte tra logica e dati.
- Infine, moduli come **prisma/**, **public/**, **schemas/**, **middleware/** e **routes-auth/** hanno un ruolo più infrastrutturale, con pochissime dipendenze, come ci si aspetta per elementi di configurazione, asset statici e validazioni.

Nel complesso, la matrice evidenzia una chiara separazione dei livelli architetturali (Presentation, Logic e Data), con responsabilità ben definite e moduli chiave facilmente individuabili. Questo tipo di analisi è utile per comprendere quali parti del sistema siano più critiche e come intervenire in modo mirato per manutenzione o evoluzioni future.

3.4 Sequence Diagram

Di seguito è riportato il **Sequence Diagram** della funzione di creazione di un evento. È stata analizzata questa nello specifico perché si prevede essere quella più impattata dalle modifiche che verranno apportate. Il diagramma rappresenta il flusso di esecuzione della funzione durante la creazione di un evento, evidenziando anche i possibili errori in caso di campi non validi. La rappresentazione schematica delle funzioni ha consentito di osservare le interazioni tra oggetti a un livello di astrazione più alto rispetto al codice, avendo così facilitato la comprensione dei processi e del flusso d'esecuzione.

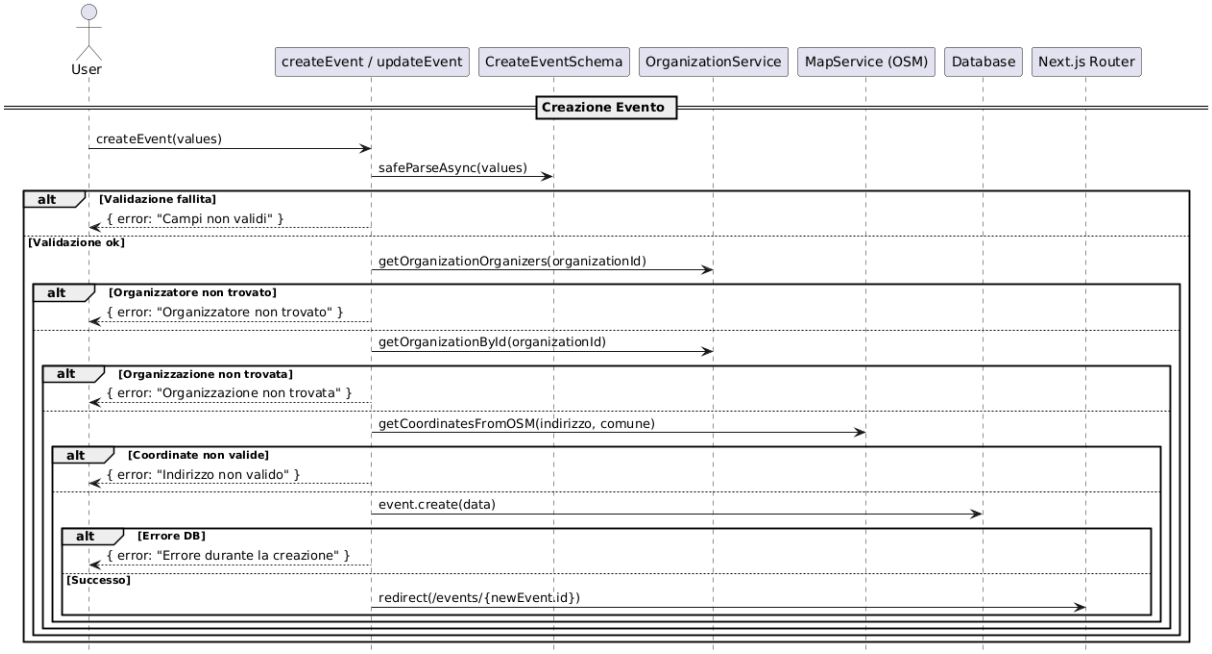

Figure 3: Sequence Diagram della creazione di un evento

.

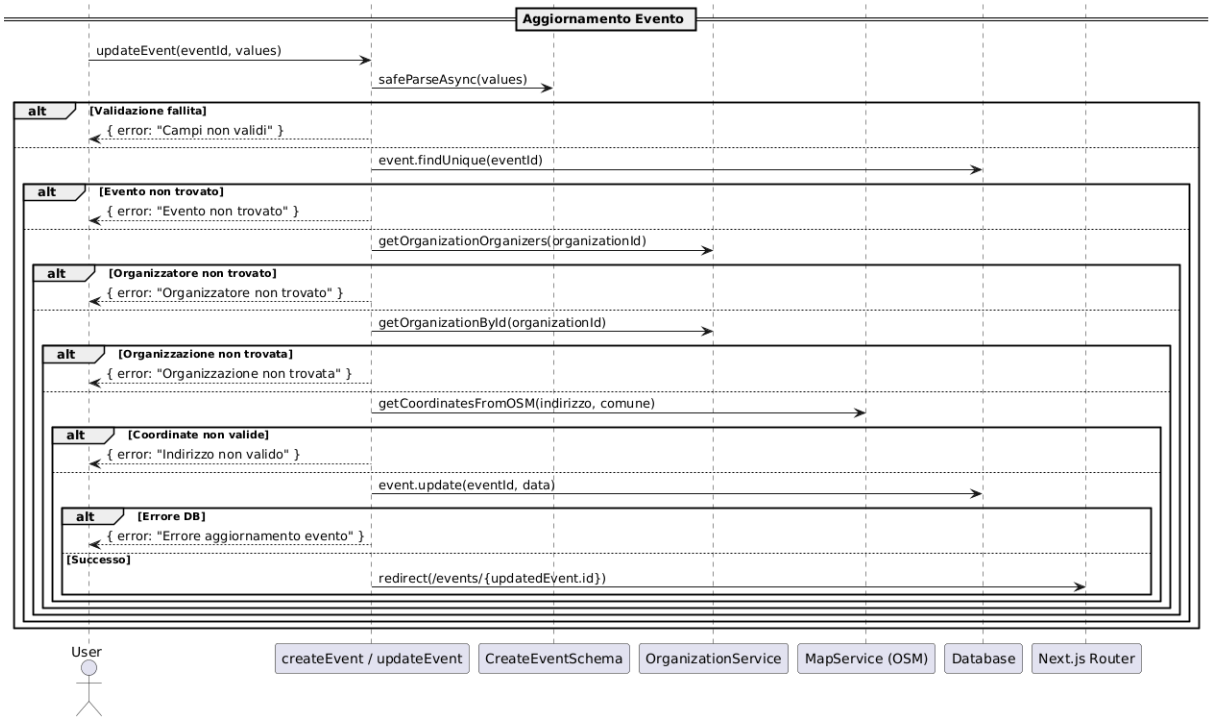

Figure 4: Sequence Diagram della modifica di un evento

.

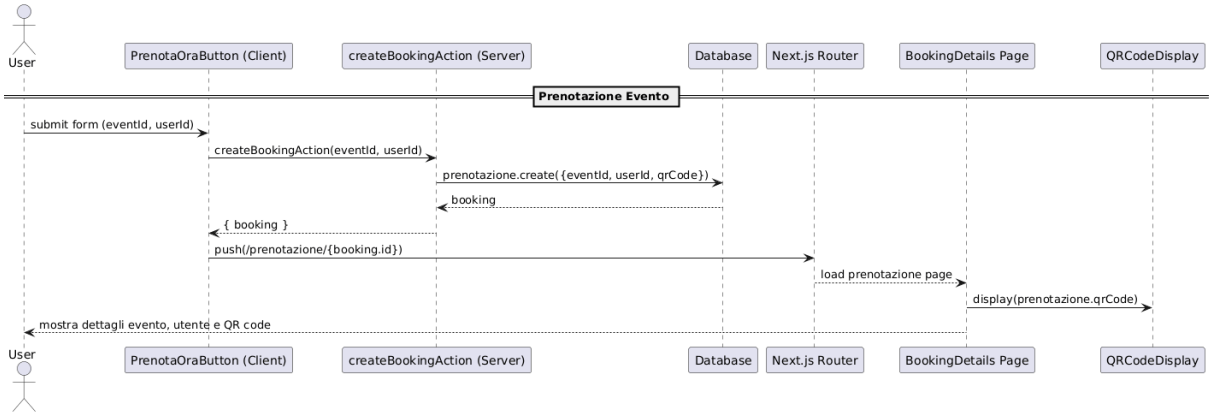

Figure 5: Sequence Diagram della prenotazione ad un evento

.

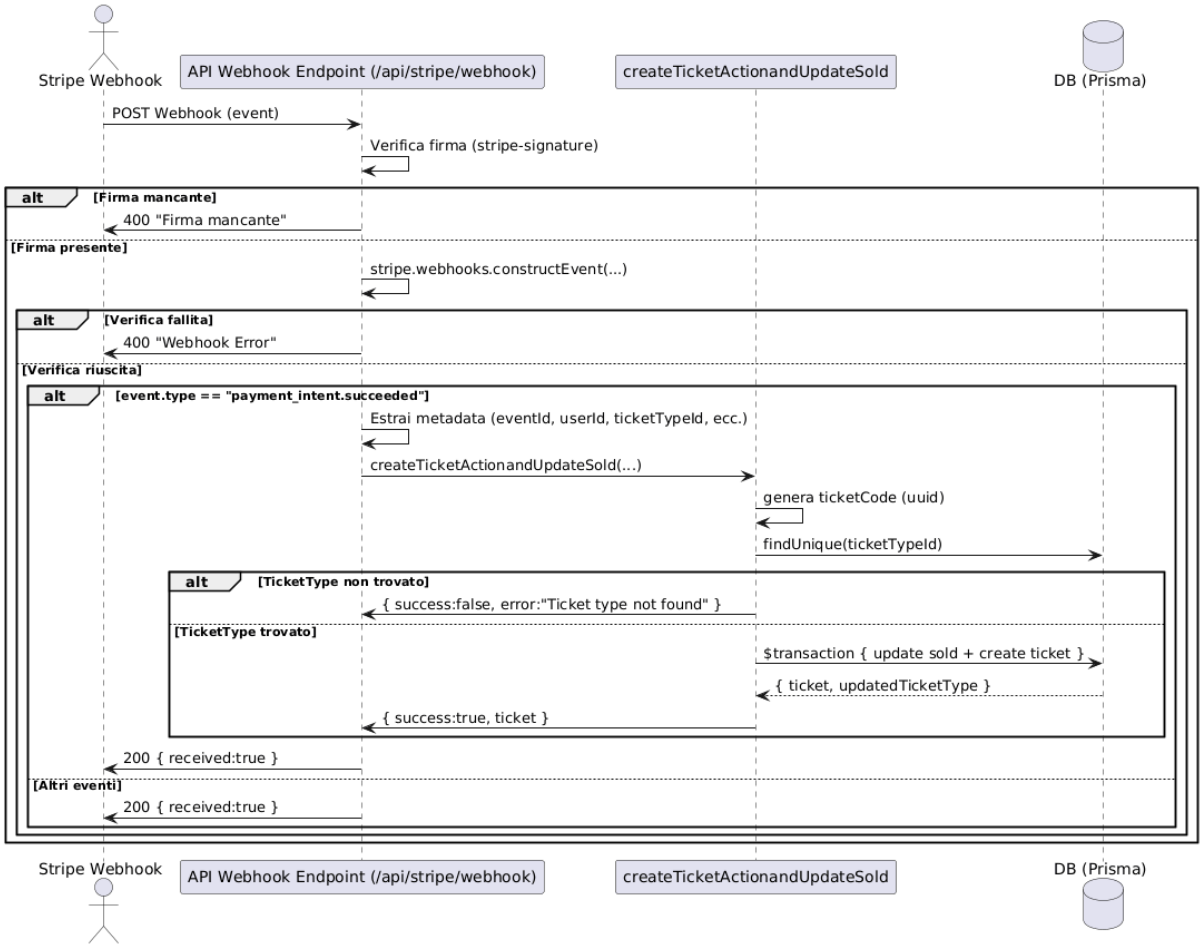

Figure 6: Sequence Diagram dell'acquisto di un biglietto

.

3.5 Analisi dei Requisiti

A seguire sono riportati i Requisiti Funzionali pre-esistenti della piattaforma Evently. Per ciascun requisito, è indicato cosa gli utenti o gli organizzatori possono fare all'interno del sistema, così da avere una visione chiara delle funzionalità già presenti prima dell'introduzione delle nuove feature.

ID	**Requisito Funzionale**	**Descrizione**
RF_1	Registrazione	L'ospite si registra nel sistema.
RF_2	Login	L'utente accede al sistema con le proprie credenziali.
RF_3	Visualizzazione eventi pubblici	L'ospite può visualizzare gli eventi pubblici disponibili.
RF_4	Gestione profilo utente	L'utente aggiorna le proprie informazioni personali.
RF_5	Navigazione eventi	L'utente naviga tra gli eventi pubblici disponibili.
RF_7	Eventi Preferiti	L'utente può aggiungere degli eventi ai preferiti.
RF_8	Prenotazione eventi	L'utente si prenota per eventi gratuiti.
RF_9	Gestione organizzazioni	L'organizzatore crea e modifica le proprie organizzazioni.
RF_10	Gestione eventi	L'organizzatore crea e modifica eventi.
RF_11	Gestione biglietti	L'organizzatore crea diverse tipologie di biglietti per i propri eventi.
RF_12	Acquisto biglietti	L'utente acquista biglietti per gli eventi a pagamento.
RF_13	Prenotazione evento	L'utente può effettuare prenotazioni per gli eventi.
RF_14	Logout	L'utente si disconnette dal sistema.

Table 2: Requisiti funzionali pre-esistenti della piattaforma Evently, aggiornati con la gestione prenotazioni

4 Change Request

Le due Change Request proposte risultano tra loro strettamente collegate.
In particolare, la funzionalità di notifiche (CR_2) alla creazione di un evento si basa sulla possibilità per l'utente di seguire una o più organizzazioni (CR_1). Quando un'organizzazione seguita crea un nuovo evento, il sistema deve infatti generare una notifica rivolta a tutti gli utenti che hanno aggiunto quella organizzazione ai propri preferiti.
Per questa ragione, l'implementazione delle due CR avverrà in maniera sequenziale:

- prima la **CR_1**, che introduce la gestione delle organizzazioni preferite;
- successivamente la **CR_2**, che sfrutta i dati raccolti dalla CR_1 per la generazione e la visualizzazione delle notifiche.

4.1 CR_1 – Organizzazioni Preferite

Descrizione Aggiungere un pulsante "Segui" nella pagina organizzazione. Gli utenti potranno visualizzare le organizzazioni seguite in una nuova sezione della dashboard.

Modifiche al sistema

- Database: nuova tabella `user_favorites`.
- Backend: azioni per seguire/smettere di seguire.
- Frontend: pulsante "Segui" + UI sezione "Organizzazioni Preferite".

Priority	Alta
Effort	Medio
Conseguenze della non accettazione	Gli utenti non avranno la possibilità di seguire organizzazioni e mantenere un legame costante con esse, con riduzione del coinvolgimento.

4.2 CR_2 – Notifiche

Descrizione Creazione di una sezione "Notifiche" accessibile dalla dashboard o dall'header. Le notifiche saranno mostrate in elenco ordinato per data. Tipologie principali:

- creazione di un nuovo evento da parte di un'organizzazione seguita;
- conferma di acquisto biglietto;
- conferma di prenotazione evento.

Modifiche al sistema

- Database: nuova tabella `notifications`.
- Backend: azioni per generare e recuperare notifiche.
- Frontend: sezione "Notifiche" nella dashboard.

Priority	Alta
Effort	Alto
Conseguenze della non accettazione	L'utente non riceverebbe feedback tempestivo né aggiornamenti dagli organizzatori seguiti, riducendo la fidelizzazione e l'esperienza d'uso.

5 Impact Analysis

Poiché le modifiche introdotte dalla Change Request possono avere effetti collaterali sul sistema, è stata condotta un'impact analysis di ogni funzionalità aggiunta. L'obiettivo era identificare quali moduli potrebbero essere interessati e come la modifica possa propagarsi all'interno della piattaforma **Evently**.

Il processo seguito è stato il seguente:

1. **Identificazione dei requisiti coinvolti:** per ciascuna nuova funzionalità (Organizzazioni Preferite e Notifiche Utente), sono stati individuati i requisiti funzionali corrispondenti, così da mappare la relazione tra funzionalità e moduli coinvolti.
2. **Analisi della matrice delle dipendenze:** ha aiutato a verificare i legami tra le cartelle del progetto indicate nel **Component Diagram** 2, per comprendere come una modifica in un modulo potesse influenzare altri moduli, così da poter definire lo **Starting Impact Set** e il **Candidate Impact Set**.
3. **Esame del codice sorgente:** sono state esplorate le funzioni e i componenti coinvolti per confermare quali parti del sistema sarebbero effettivamente modificate.

Di seguito è riportata la descrizione degli Impact Set individuati:

- **Starting Impact Set (SIS):** insieme iniziale di moduli che presumibilmente verranno influenzati dalla CR, basato sui requisiti funzionali.
- **Candidate Impact Set (CIS):** insieme di moduli che, secondo la matrice delle dipendenze e l'analisi preliminare, potrebbero essere interessati dalla CR.
- **Discovered Impact Set (DIS):** nuovi moduli rilevati durante lo sviluppo della CR che si sono rivelati influenzati, non previsti inizialmente nel CIS.
- **Actual Impact Set (AIS):** moduli effettivamente modificati a seguito dell'implementazione della CR.
- **False Positive Impact Set (FPIS):** moduli stimati come influenzati dalla CR ma che, alla fine, non hanno subito alcuna modifica.

Gli ultimi tre Impact Sets sono stati individuati dopo aver implementato le modifiche delle due Change Requests.

Grazie a questa metodologia, è stato possibile valutare con precisione le conseguenze delle nuove funzionalità su **Evently**, minimizzando il rischio di introdurre errori in moduli non coinvolti.

5.1 CR_1 – Organizzazioni Preferite Impact Set

Nella tabella seguente sono elencati i moduli che potrebbero essere impattati dalla CR_1:

Modulo	SIS	CIS	AIS	DIS	FPIS
`prisma/schema.prisma`	X	X	X		
`component/organization-client.tsx`	X	X	X		
`app/organization/[organizationId]/page.tsx`		X	X		
`app/my-favorites/page.tsx`	X	X	X		

Table 3: Impact Set CR_2

- `prisma/schema.prisma`: definizione del modello del database, da aggiornare per includere la nuova tabella `favourites_organizations`.
- `component/organization-client.tsx`:
- `app/organization/[organizationId]/page.tsx`: modulo impattato dalla propagazione della modifica di organization-client.tsx
- `app/my-favorites/page.tsx`: modulo impattato per la sezione "preferiti"

5.2 CR_2 – Notifiche Impact Set

Nella tabella seguente sono elencati i moduli che potrebbero essere impattati dalla CR_2:

Modulo	SIS	CIS	AIS	DIS	FPIS
`prisma/schema.prisma`	X	X	X		
`app/layout.tsx`		X			X
`components/altre/user-menu.tsx`	X	X	X		
`actions/event.ts`	X	X	X		
`actions/ticket.ts`	X	X	X		
`actions/prenotazioni.ts`	X	X	X		

Table 4: Impact Set CR_2

- `prisma/schema.prisma`: necessario estendere il modello dei dati con la tabella `notifications`.
- `components/altre/user-menu.tsx`: componente già esistente che gestisce la navigazione per l'utente per mostrare l'icona di accesso alle notifiche.
- `action/event.ts`: quando si crea un evento, bisogna generare notifiche per gli utenti che seguono l'organizzazione.
- `action/ticket.ts`: quando un utente acquista un biglietto, bisogna generare una notifica.
- `action/prenotazioni.ts`: quando un utente effettua la prenotazione di un evento, bisogna generare una notifica.

6 Attuazione delle modifiche

Dopo aver analizzato i possibili impatti sul sistema **Evently**, sono stati creati due branch dedicati nel repository GitHub ufficiale, così da separare in modo chiaro le implementazioni:

- `feature/organizzazioni-preferite`
- `feature/notifiche-utente` (derivato dal branch `organizzazioni-preferite`, per estendere il lavoro già svolto con un ulteriore livello funzionale)

6.1 CR_1 – Estensione con "Organizzazioni Preferite"

L'estensione ha introdotto un nuovo flusso che consente all'utente di seguire le organizzazioni di interesse.

Database

- Creazione della tabella `UserFavoriteOrganization` per gestire la relazione molti-a-molti tra utenti e organizzazioni.

Backend (Server Actions)

- Azioni per seguire o smettere di seguire un'organizzazione.
- Azioni per recuperare la lista delle organizzazioni preferite associate a un utente.

Frontend (Next.js/React)

- Pulsante "Segui" nella pagina di dettaglio di ogni organizzazione.
- Nuova sezione "I miei preferiti" nella dashboard, con l'elenco delle organizzazioni seguite.

6.2 CR_2 – Estensione con sistema di Notifiche Utente

Il sistema di notifiche interne permette di avvisare l'utente in tempo reale rispetto a eventi rilevanti.

Database

- Creazione della tabella `Notifications`.
- Creazione della tabella pivot `UserNotification` per associare ogni notifica ai destinatari.

Backend (Server Actions)

- Azioni per recuperare l'elenco delle notifiche di un utente.
- Generazione automatica di notifiche in tre casi principali:
  - Pubblicazione di un nuovo evento da parte di un'organizzazione seguita.
  - Acquisto di un biglietto.
  - Conferma di prenotazione.

Frontend (Next.js/React)

- Nuova sezione “Notifiche” accessibile dalla dashboard e dall’header.
- Interfaccia per la visualizzazione delle notifiche in ordine cronologico.

6.3 Modello ER aggiornato

Il modello Entità-Relazione è stato aggiornato con l’aggiunta delle nuove entità e relazioni necessarie per supportare le funzionalità di **Organizzazioni Preferite** e **Notifiche Utente**.

- Aggiunta della tabella `FavoriteOrganization` per rappresentare la relazione molti-a-molti tra `User` e `Organization`.
- Aggiunta della tabella `Notification` per memorizzare le notifiche generate dal sistema (tipo, messaggio, timestamp).
- Aggiunta della tabella pivot `UserNotification` per associare le notifiche ai singoli utenti destinatari, con lo stato di lettura (letto/non letto).

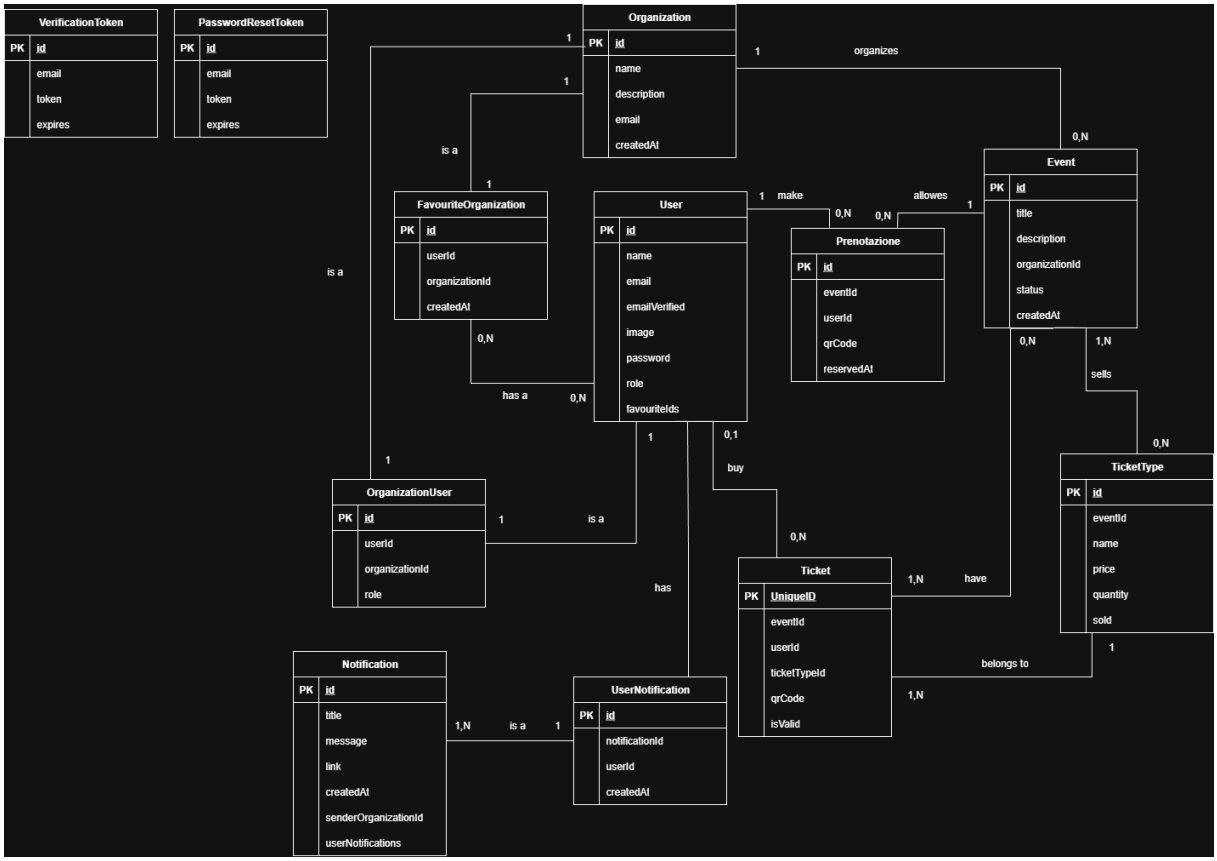

Figure 7: Diagramma Entità-Relazione aggiornato di Evently

6.4 Sequence Diagram – Gestione eventi

Il diagramma di sequenza relativo alla gestione eventi è stato aggiornato per includere la generazione automatica di notifiche al momento della creazione o modifica di un evento, quando questo appartiene a un'organizzazione seguita dagli utenti.

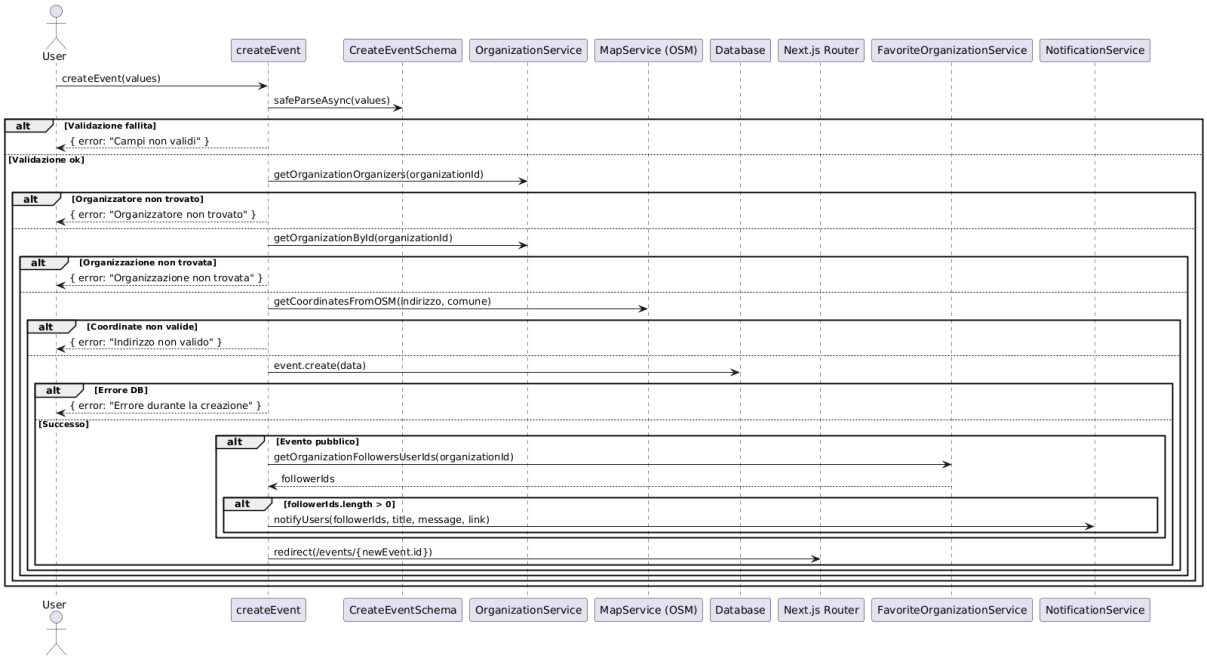

Figure 8: Sequence Diagram aggiornato – Creazione eventi

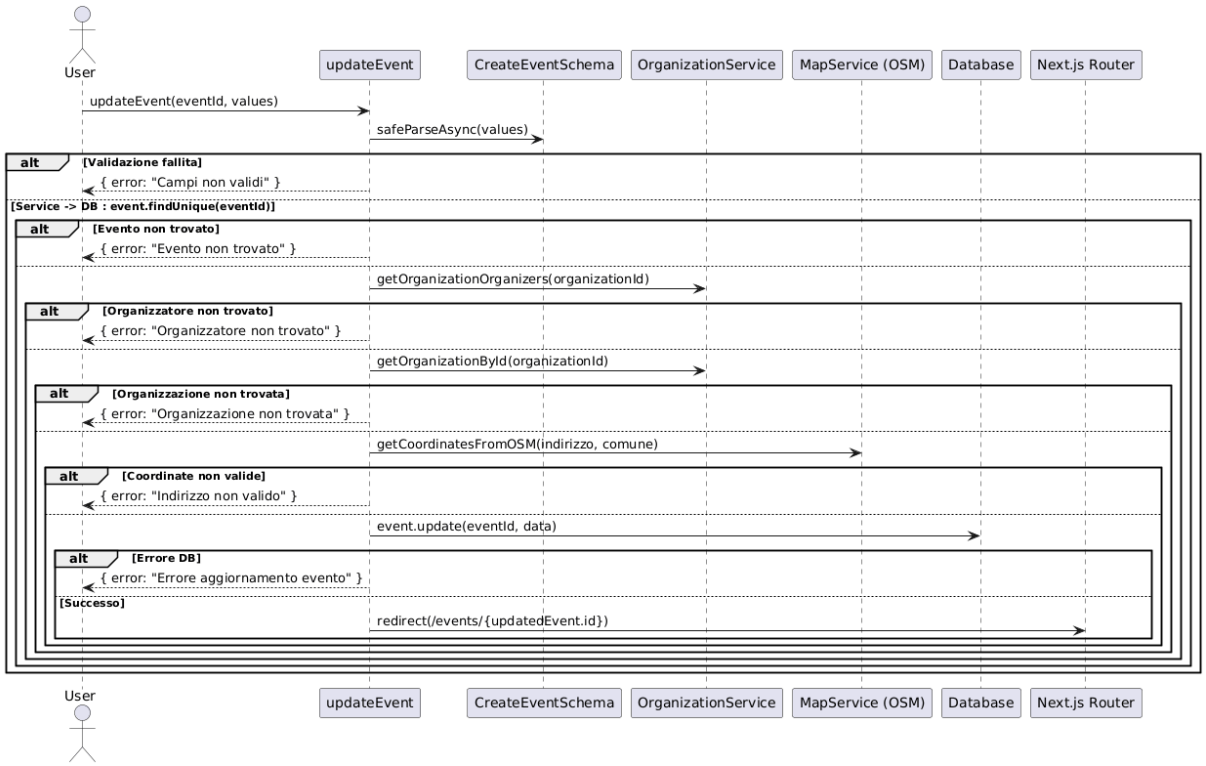

Figure 9: Sequence Diagram aggiornato – Modifica eventi

6.5 Sequence Diagram – Acquisto biglietti

Il diagramma di sequenza dell'acquisto biglietti è stato aggiornato per mostrare la generazione e l'invio di una notifica all'utente a seguito di un acquisto completato.

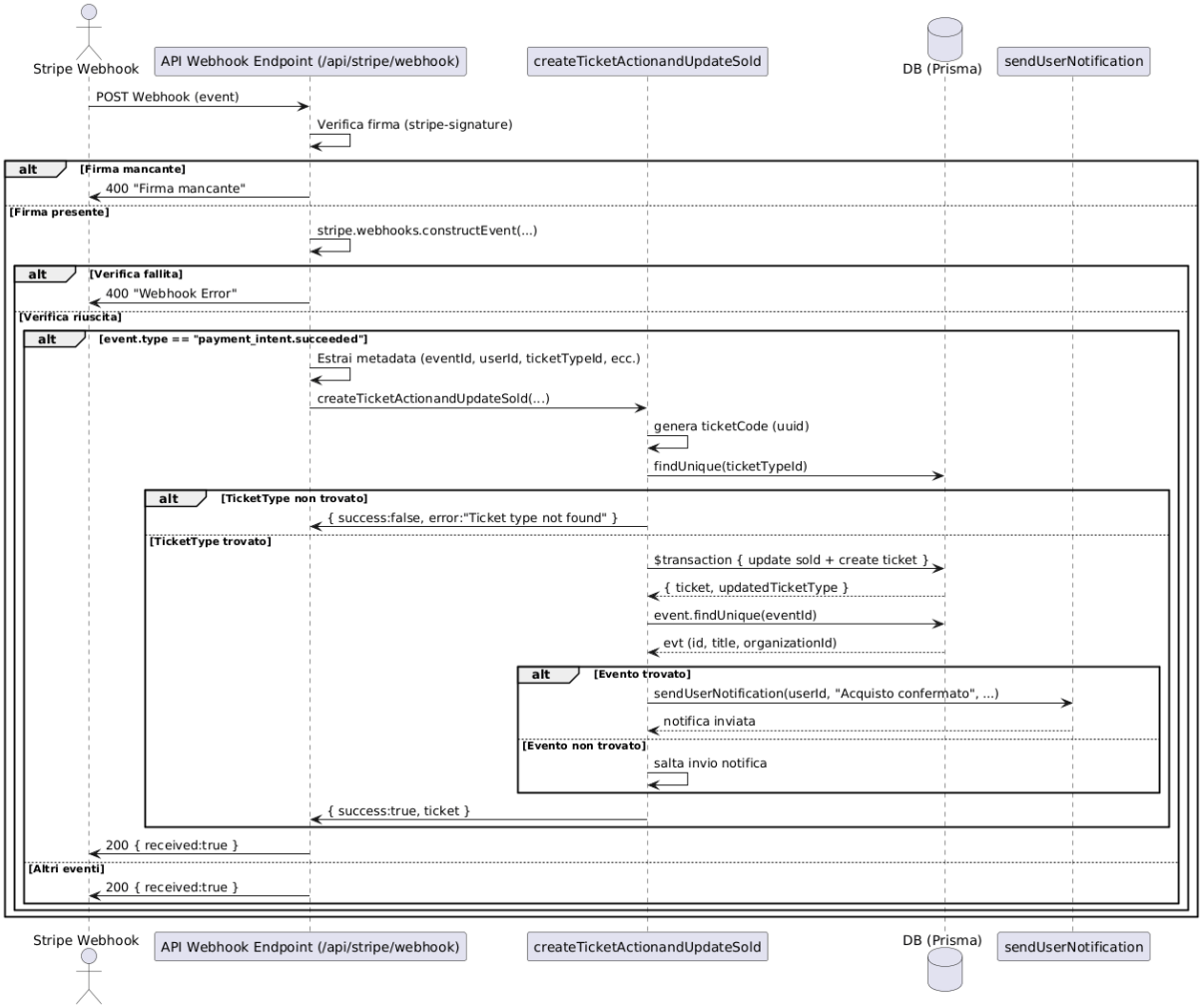

Figure 10: Sequence Diagram aggiornato – Acquisto biglietti

6.6 Sequence Diagram – Prenotazione evento

Il diagramma di sequenza relativo alle prenotazioni è stato aggiornato per includere la generazione di una notifica di conferma prenotazione inviata all'utente.

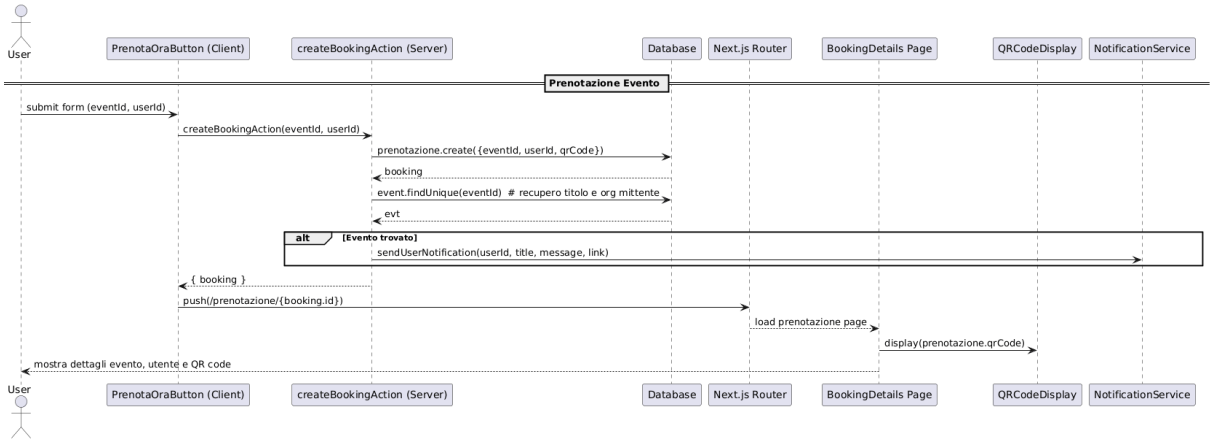

Figure 11: Sequence Diagram aggiornato – Prenotazione evento

6.7 Requisiti Funzionali aggiornati

A seguire c'è la tabella dei requisiti funzionali aggiornata all'attuazione delle modifiche.

ID	Requisito Funzionale	Descrizione
RF_1	Registrazione	L'ospite si registra nel sistema.
RF_2	Login	L'utente accede al sistema con le proprie credenziali.
RF_3	Visualizzazione eventi pubblici	L'ospite può visualizzare gli eventi pubblici disponibili.
RF_4	Gestione profilo utente	L'utente aggiorna le proprie informazioni personali.
RF_5	Navigazione eventi	L'utente naviga tra gli eventi pubblici disponibili.
RF_7	Eventi Preferiti	L'utente può aggiungere degli eventi ai preferiti.
RF_8	Prenotazione eventi	L'utente si prenota per eventi gratuiti.
RF_9	Gestione organizzazioni	L'organizzatore crea e modifica le proprie organizzazioni.
RF_10	Gestione eventi	L'organizzatore crea e modifica eventi.
RF_11	Gestione biglietti	L'organizzatore crea diverse tipologie di biglietti per i propri eventi.
RF_12	Acquisto biglietti	L'utente acquista biglietti per gli eventi a pagamento.
RF_13	Prenotazione evento	L'utente può effettuare prenotazioni per gli eventi.
RF_14	Logout	L'utente si disconnette dal sistema.
RF_15	**Organizzazioni Preferite**	L'utente si può aggiungere le organizzazioni alla lista dei preferiti.
RF_16	**Visualizzazione notifiche**	L'utente può visualizzare le notifiche relative alla creazione di un'evento di un'organizzazione preferita, alla prenotazione di un evento ed all'acquisto di un biglietto.

Table 5: Requisiti funzionali pre-esistenti della piattaforma Evently, aggiornati con la gestione prenotazioni