UNIVERSITÀ DEGLI STUDI DI SALERNO

DIPARTIMENTO DI INFORMATICA

Test Plan Evently

Progetto di Ingegneria, Gestione ed Evoluzione del Software

Giulio De Pascale Matr.0522502049
Nicoletta Mauro Matr.0522502072

Contents

1 Introduzione

L'obiettivo del seguente **Test Plan** è garantire la correttezza e l'affidabilità delle nuove funzionalità introdotte dalle Change Request CR_1 (Organizzazioni Preferite) e CR_2 (Notifiche). Queste modifiche mirano a migliorare il coinvolgimento degli utenti, consentendo loro di seguire le organizzazioni di interesse e ricevere notifiche personalizzate, in linea con gli obiettivi del progetto Evently.

Il test plan prevede la verifica di:

- corretto aggiornamento del database con le nuove tabelle `user_favorites` e `notifications`;
- corretto funzionamento delle server actions per seguire/smettere di seguire e per generare e recuperare notifiche;
- visualizzazione corretta delle nuove sezioni nella dashboard;
- assenza di regressioni nelle funzionalità pre-esistenti (login, visualizzazione eventi, gestione profilo).

Al termine dell'esecuzione del piano di test, ci si attende di validare che:

1. l'utente possa gestire le proprie organizzazioni preferite in modo semplice e intuitivo;
2. le notifiche vengano generate e mostrate correttamente;
3. l'esperienza complessiva risulti coerente e senza errori, favorendo il mantenimento del legame tra utenti e organizzazioni.

2 Testing Approach

Poiché le modifiche previste per Evently riguardano l'evoluzione di un sistema già esistente, è fondamentale garantire che le nuove funzionalità non compromettano il corretto funzionamento dell'applicazione. Per questo motivo, il piano di test prevede l'adozione di più strategie di testing, con l'obiettivo di validare sia le modifiche introdotte, sia la stabilità del sistema complessivo.

2.1 Regression Testing

Verranno rieseguiti i test esistenti sulle funzionalità già presenti, come la registrazione e le prenotazioni da parte dell'utente sui due nuovi branch `feature/notifiche-utente` e `feature/organizzazioni-preferite` al fine di accertare che l'introduzione delle CR_1 e CR_2 non abbia generato nuovi malfunzionamenti o regressioni.

2.2 Unit Testing

Sono stati creati e lanciati unit test per le funzionalità introdotte dalle Change Requests CR1 e CR2, in particolare:

- **CR1 - Organizzazioni Preferite**: test delle azioni per seguire o smettere di seguire un'organizzazione, verificando il corretto aggiornamento dello stato lato client e server e la gestione degli errori;
- **CR2 - Notifiche**: test delle funzioni di creazione eventi, prenotazioni e biglietti, con verifica dell'invio delle notifiche agli utenti e della gestione dei casi di fallimento nell'invio.

Tutti i test sono stati eseguiti tramite Jest, simulando le chiamate alle API e verificando il comportamento delle funzioni in scenari sia positivi sia negativi.

3 Features da testare

In questa sezione vengono descritte ad alto livello le funzionalità che saranno oggetto di test, raggruppate per Change Request.

3.1 CR_1 – Organizzazioni Preferite

Test Item: L'obiettivo della modifica è introdurre la possibilità per l'utente di seguire le organizzazioni, tramite un pulsante "Segui" sulla pagina di dettaglio dell'organizzazione. I dati saranno salvati nella nuova tabella `user_favorites` e mostrati in una nuova sezione della dashboard.

Tested Features:

- Creazione e gestione dei record nella tabella `user_favorites`.
- Visualizzazione delle organizzazioni seguite nella dashboard.
- Funzionamento del pulsante "Segui" / "Smetti di seguire".

3.2 CR_2 – Notifiche

Test Item: L'obiettivo della modifica è introdurre una sezione "Notifiche" accessibile dalla dashboard. Il sistema deve generare notifiche per i seguenti flussi principali, come verificato dai test:

- Creazione di un nuovo evento pubblico da parte di un'organizzazione seguita.
- Conferma prenotazione di un evento da parte dell'utente.
- Conferma acquisto di un biglietto da parte dell'utente.

Tested Features:

- Generazione automatica delle notifiche in corrispondenza degli eventi previsti.
- Invio della notifica agli utenti/follower interessati.
- Gestione corretta degli errori di invio notifiche (fallimento non blocca il flusso principale).
- Recupero e visualizzazione delle notifiche nella dashboard o nell'header.

4 Category Partition Post-Modifiche

4.1 Testing funzionale di CR_1 – Organizzazioni Preferite

Il sistema deve permettere agli utenti di seguire o smettere di seguire le organizzazioni, aggiornando correttamente lo stato lato client e server.

Descrizione L'utente visualizza la pagina dell'organizzazione e può:

- seguire un'organizzazione se non già seguita,
- smettere di seguire un'organizzazione se già seguita.

Il sistema verifica:

- validità dell'utente loggato,
- esistenza dell'organizzazione,
- corretto aggiornamento dello stato lato client e server,
- rollback e messaggio in caso di errore.

Parametri userId, organizationId, initialFollowing, pending

Categorie e Scelte

- **initialFollowing**:
  - F1: già seguito,
  - F2: non seguito
- **userId**:
  - U1: utente loggato,
  - U2: utente non loggato
- **organizationId**:
  - O1: organizzazione esistente,
  - O2: organizzazione inesistente
- **pending**:
  - P1: in corso,
  - P2: completata

Vincoli

- L'utente deve essere loggato (U1), altrimenti il sistema mostra un errore e non esegue l'azione.
- L'organizzazione deve esistere in DB (O1), altrimenti il sistema mostra un errore e non modifica lo stato di follow.
- Lo stato `pending` (P1/P2) rappresenta l'avanzamento dell'operazione: non possono esserci più richieste simultanee per la stessa organizzazione.

Test Case ID	**Test Frame**	**Result**
TC_1_CR_1	U1, O1, F2	Success: follow eseguito
TC_1_1_CR_1	U1, O1, F1	Success: unfollow eseguito
TC_1_2_CR_1	U2, O1, F2	Error: utente non loggato
TC_1_3_CR_1	U1, O2, F2	Error: organizzazione inesistente
TC_1_4_CR_1	U1, O1, F2	Error: follow fallito
TC_1_5_CR_1	U1, O1, F1	Error: unfollow fallito

Table 1: Category Partition CR_1 – Organizzazioni preferite

Test Case ID	TC_1_CR_1
Test Frame	U1, O1, F2
Pre-conditions	Utente autenticato; organizzazione esistente; utente non segue l'organizzazione.
Flow of events	L'utente imposta i parametri: - userId: valido - organizationId: valido - initialFollowing: false - pending: false e preme sul pulsante `Segui`.
Oracle	Il sistema registra il follow lato server e aggiorna lo stato lato client; il pulsante diventa `Smetti di seguire`.

Table 2: Test Case Specification TC_1_CR_1 – Successo follow eseguito

Test Case ID	TC_1_1_CR_1
Test Frame	U1, O1, F1
Pre-conditions	Utente autenticato; organizzazione esistente; utente segue già l'organizzazione.
Flow of events	L'utente imposta i parametri: - userId: valido - organizationId: valido - initialFollowing: true - pending: false e preme sul pulsante `Smetti di seguire`.
Oracle	Il sistema registra l'unfollow lato server e aggiorna lo stato lato client; il pulsante diventa `Segui`.

Table 3: Test Case Specification TC_1_1_CR_1 – Successo unfollow eseguito

Test Case ID	TC_1_2_CR_1
Test Frame	U2, O1, F2
Pre-conditions	Utente non autenticato; organizzazione esistente.
Flow of events	L'utente imposta i parametri: - userId: null - organizationId: valido - initialFollowing: false - pending: false e preme sul pulsante `Segui`.
Oracle	Il sistema mostra il messaggio di errore "Effettua il login" e non registra il follow.

Table 4: Test Case Specification TC_1_2_CR_1 – Errore utente non loggato

Test Case ID	TC_1_2_CR_1
Test Frame	U2, O1, F2
Pre-conditions	Utente non autenticato; organizzazione esistente.
Flow of events	L'utente imposta i parametri: - userId: null - organizationId: valido - initialFollowing: false - pending: false e preme sul pulsante `Segui`.
Oracle	Il sistema mostra il messaggio di errore "Effettua il login" e non registra il follow.

Table 5: Test Case Specification TC_1_2_CR_1 – Errore utente non loggato

Test Case ID	TC_1_3_CR_1
Test Frame	U1, O2, F2
Pre-conditions	Utente autenticato; organizzazione non esistente.
Flow of events	L'utente imposta i parametri: - userId: valido - organizationId: inesistente - initialFollowing: false - pending: false e preme sul pulsante `Segui`.
Oracle	Il sistema mostra il messaggio di errore "Organizzazione non trovata" e non registra il follow.

Table 6: Test Case Specification TC_1_3_CR_1 – Errore organizzazione inesistente

Test Case ID	TC_1_4_CR_1
Test Frame	U1, O1, F2
Pre-conditions	Utente autenticato; organizzazione esistente; utente non segue l'organizzazione.
Flow of events	L'utente imposta i parametri: - userId: valido - organizationId: valido - initialFollowing: false - pending: false e preme sul pulsante `Segui`; il server restituisce errore.
Oracle	Il sistema fa rollback dello stato lato client e mostra un messaggio di errore.

Table 7: Test Case Specification TC_1_4_CR_1 – Errore follow fallito

Test Case ID	TC_1_5_CR_1
Test Frame	U1, O1, F1
Pre-conditions	Utente autenticato; organizzazione esistente; utente segue già l'organizzazione.
Flow of events	L'utente imposta i parametri: - userId: valido - organizationId: valido - initialFollowing: true - pending: false e preme sul pulsante `Smetti di seguire`; il server restituisce errore.
Oracle	Il sistema fa rollback dello stato lato client e mostra un messaggio di errore.

Table 8: Test Case Specification TC_1_5_CR_1 – Errore unfollow fallito

4.2 Testing funzionale CR_2 – Notifiche

Il sistema deve permettere all'utente di visualizzare le notifiche relative alla creazione di un evento da parte di un'organizzazione seguita, alla prenotazione di un evento e all'acquisto di un biglietto.

Descrizione Le notifiche vengono mostrate in una sezione dedicata nella dashboard o nell'header dell'applicazione. I flussi principali generano notifiche quando:

- un'organizzazione seguita crea un nuovo evento pubblico;
- l'utente prenota un evento;
- l'utente acquista un biglietto.

Parametri userId, eventId, ticketTypeId, paymentId, organizationId, title, message, link, senderOrganizationId

Categorie e Scelte

- **Tipo di notifica**
  - E1: nuovo evento organizzazione seguita
  - B1: conferma acquisto biglietto
  - P1: conferma prenotazione evento
- **Esito invio**
  - N1: notifica inviata correttamente
  - N2: errore invio (loggato, flow principale completato)
- **Follower/Utente presente**
  - F1: follower/utente presente
  - F2: follower/utente assente

Vincoli

- L'invio della notifica non deve bloccare il flusso principale (evento creato, prenotazione o biglietto).
- In caso di errore nell'invio, il sistema deve loggare l'errore.

Test Case ID	**Test Frame**	**Result**
TC_1_RF_CR2	E1, F1, N1	Success: nuovo evento pubblico, notifica inviata a tutti i follower
TC_2_RF_CR2	E1, F2	Success: nuovo evento pubblico, nessun follower, nessuna notifica inviata
TC_3_RF_CR2	E1, F1, N2	Error: invio notifica fallito, evento creato comunque
TC_4_RF_CR2	E1, S2	Success: evento privato creato, nessuna notifica inviata

Table 9: Category Partition CR_2 – Notifiche generazione eventi

Test Case ID	**Test Frame**	**Result**
TC_5_RF_CR2	P1, F1, N1	Success: prenotazione evento confermata, notifica inviata all'utente
TC_6_RF_CR2	P1, F1, N2	Error: invio notifica fallito, prenotazione salvata comunque

Table 10: Category Partition CR_2 – Notifiche prenotazione evento

Test Case ID	**Test Frame**	**Result**
TC_7_RF_CR2	B1, F1, N1	Success: acquisto biglietto confermato, notifica inviata all'utente
TC_8_RF_CR2	B1, F1, N2	Error: invio notifica fallito, biglietto creato comunque

Table 11: Category Partition CR_2 – Notifiche acquisto biglietto

Test Case ID	TC_1_RF_CR2
Test Frame	Event creation, public event, notification to all followers
Pre-conditions	Utente autenticato; organizzazione esistente; utenti follower presenti.
Flow of events	Creazione di un nuovo evento pubblico con: - title: "Concerto Jazz" - description: "Evento musicale di jazz con artisti locali" - organizationId: "org123" - eventDate: "2025-10-10T20:30:00" - status: "pubblico" - isReservationActive: true Il sistema invia notifica a tutti i follower tramite `notifyUsers`.
Oracle	`notifyUsers` viene chiamato con successo. Tutti i follower ricevono la notifica.

Table 12: TC_1_RF_CR2 – Nuovo evento pubblico, notifica inviata a tutti i follower

Test Case ID	TC_2_RF_CR2
Test Frame	Event creation, public event, no followers
Pre-conditions	Utente autenticato; organizzazione esistente; nessun follower presente.
Flow of events	Creazione di un nuovo evento pubblico con gli stessi parametri di TC_1. Il sistema tenta di inviare notifica tramite `notifyUsers`.
Oracle	`notifyUsers` viene chiamato ma non invia notifiche, perché non ci sono follower.

Table 13: TC_2_RF_CR2 – Nuovo evento pubblico, nessun follower, nessuna notifica

Test Case ID	TC_3_RF_CR2
Test Frame	Event creation, public event, notification failure
Pre-conditions	Utente autenticato; organizzazione esistente; utenti follower presenti.
Flow of events	Creazione di un nuovo evento pubblico con parametri identici al TC_1. `notifyUsers` fallisce (mockRejectedValueOnce).
Oracle	Evento creato comunque. L'errore di notifica non blocca il processo.

Table 14: TC_3_RF_CR2 – Nuovo evento pubblico, invio notifica fallito, evento creato comunque

Test Case ID	TC_4_RF_CR2
Test Frame	Event creation, private event
Pre-conditions	Utente autenticato; organizzazione esistente.
Flow of events	Creazione di un nuovo evento privato con: - status: "privato" - altri parametri identici al TC_1. Nessuna notifica inviata.
Oracle	Evento creato correttamente senza invio di notifiche.

Table 15: TC_4_RF_CR2 – Evento privato creato, nessuna notifica inviata

Test Case ID	TC_5_RF_CR2
Test Frame	Booking creation, notification sent
Pre-conditions	Utente autenticato; evento esistente; prenotazione possibile.
Flow of events	Creazione prenotazione con: - eventId: "event123" - userId: "user123" Il sistema invia notifica tramite `sendUserNotification`.
Oracle	`sendUserNotification` viene chiamato con successo. Prenotazione confermata.

Table 16: TC_5_RF_CR2 – Prenotazione confermata, notifica inviata

Test Case ID	TC_6_RF_CR2
Test Frame	Booking creation, notification failure
Pre-conditions	Utente autenticato; evento esistente; prenotazione possibile.
Flow of events	Creazione prenotazione con i parametri TC_5. `sendUserNotification` fallisce (mockRejectedValueOnce).
Oracle	Prenotazione salvata comunque. L'errore di notifica non blocca il processo.

Table 17: TC_6_RF_CR2 – Prenotazione, invio notifica fallito, prenotazione salvata comunque

Test Case ID	TC_7_RF_CR2
Test Frame	Ticket purchase, notification sent
Pre-conditions	Utente autenticato; evento esistente; biglietto disponibile.
Flow of events	Acquisto biglietto con: - eventId: "event123" - userId: "user123" - ticketTypeId: "T-OK" - paymentId: "stripe123" - paymentMethod: "method123" - quantity: 15 Notifica inviata tramite `sendUserNotification`.
Oracle	`sendUserNotification` viene chiamato con successo. Biglietto creato.

Table 18: TC_7_RF_CR2 – Acquisto biglietto confermato, notifica inviata

Test Case ID	TC_8_RF_CR2
Test Frame	Ticket purchase, notification failure
Pre-conditions	Utente autenticato; evento esistente; biglietto disponibile.
Flow of events	Acquisto biglietto con i parametri TC_7. `sendUserNotification` fallisce (mockRejectedValueOnce).
Oracle	Biglietto creato comunque. L'errore di notifica non blocca il processo.

Table 19: TC_8_RF_CR2 – Acquisto biglietto, invio notifica fallito, biglietto creato comunque