**EvenTour**

18 gennaio 2022

Corso di Informatica IIIB AA. 2019/2020

Progetto Progettazione e Algoritmi

Colombi Simone

Scarpellini Stefano

Contents

[Manuale Utente 7](#_Toc92277980)

[1. Avvio del server API 7](#_Toc92277981)

[2. Avvio del client web 7](#_Toc92277982)

[Iterazione 0 8](#_Toc92277983)

[1. Requisiti utente 8](#_Toc92277984)

[2. Tool Chain 10](#_Toc92277985)

[3. Use Cases 11](#_Toc92277986)

[Descrizione testuale 11](#_Toc92277987)

[Use Case diagram 20](#_Toc92277988)

[4. Architettura 21](#_Toc92277989)

[Architecture model 21](#_Toc92277990)

[Deployment diagram 22](#_Toc92277991)

[Database schema 23](#_Toc92277992)

[Component diagram 26](#_Toc92277993)

[Pattern architetturali 27](#_Toc92277994)

[Iterazione 1 (implementazione Iterazione 0) 28](#_Toc92277995)

[1. Class diagram 28](#_Toc92277996)

[2. Testing interfaccia grafica 31](#_Toc92277997)

[Analisi statica 31](#_Toc92277998)

[Analisi dinamica 32](#_Toc92277999)

[3. Testing API 33](#_Toc92278000)

[Analisi statica 33](#_Toc92278001)

[Analisi dinamica 33](#_Toc92278002)

[Iterazione 2 34](#_Toc92278003)

[1. Use Cases 34](#_Toc92278004)

[Descrizione testuale 34](#_Toc92278005)

[Use Case diagram 39](#_Toc92278006)

[2. Architettura 40](#_Toc92278007)

[Deployment diagram 40](#_Toc92278008)

[Component diagram 41](#_Toc92278009)

[3. Novità introdotte 42](#_Toc92278010)

[4. Class diagram 42](#_Toc92278011)

[5. Algoritmo evenTour 44](#_Toc92278012)

[6. Testing interfaccia grafica 45](#_Toc92278013)

[Analisi statica 45](#_Toc92278014)

[Analisi dinamica 46](#_Toc92278015)

[7. Testing API 47](#_Toc92278016)

[Analisi statica 47](#_Toc92278017)

[Analisi dinamica 47](#_Toc92278018)

[Iterazione 3 48](#_Toc92278019)

[1. Use Cases 48](#_Toc92278020)

[Descrizione testuale 48](#_Toc92278021)

[Use Case diagram 53](#_Toc92278022)

[2. Architettura 54](#_Toc92278023)

[Deployment diagram 54](#_Toc92278024)

[Component diagram 55](#_Toc92278025)

[3. Novità introdotte 56](#_Toc92278026)

[4. Class diagram 56](#_Toc92278027)

[5. Algoritmo evenTour 60](#_Toc92278028)

[6. Testing interfaccia grafica 60](#_Toc92278029)

[Analisi statica 60](#_Toc92278030)

[Analisi dinamica 61](#_Toc92278031)

[7. Testing API 64](#_Toc92278032)

[Analisi statica 64](#_Toc92278033)

[Analisi dinamica 64](#_Toc92278034)

[Sviluppi futuri 64](#_Toc92278035)

# Manuale Utente

Per vedere in opera il progetto EvenTour devono essere scaricati tutti i file contenuti nel repository di GitHub al seguente indirizzo: INDIRIZZO GITHUB

Una volta scaricato in locale, per ogni iterazione, saranno disponibili una cartella contenente il progetto Eclipse del server che fornisce le API e una cartella zippata (consigliamo 7Zip O winRAR come programma di decompressione della cartella compressa [[[1]](#footnote-1)]) contenente il progetto Vue.

## Avvio del server API

1. Aprire Eclipse (la versione consigliata è *Eclipse 2021.06 IDE for Enterprise Java and Web Developers* [[[2]](#footnote-2)], che è portable e, dopo estrazione con winRAR o 7zip, può essere aperto)

b. Posizionarsi come workspace la cartella *<cartella di download del progetto>/iterazioneN/SERVER\_API/*.

c. Il progetto può essere importato in questo modo: nella sezione Package Explorer, effettuare *Import* o *Import Project>Maven>Existing Maven Projects*. Cliccando su Next, inserire poi il path sopracitato. In questo modo si potrà selezionare il file POM e verranno importate tutte le dipendenze necessarie.

d. Posizionandosi sul progetto “*eventour*”, espandendolo, posizionandosi sulla source folder *src/main/java* e cliccando con il tasto destro, si potrà avviare l’esecuzione del server API tramite *Run As>Java application*.

## Avvio del client web

1. Installare Node.js seguendo la guida che fa riferimento al proprio sistema operativo:

* Windows: <https://www.nodeacademy.it/installare-node-js-windows/>
* Linux: <https://www.nodeacademy.it/installare-node-js-ubuntu-linux/>
* Mac Os: <https://www.nodeacademy.it/installare-node-js-mac-os-x/>

1. Dopo aver installato Node.js, è necessario recarsi tramite console dei comandi all’interno del progetto Vue.
2. Eseguire il comando npm start
3. Per fermare l’esecuzione del server web, premere CTRL+C, digitare s e premere invio.

Si trovano anche online ai seguenti indirizzi:

Iterazione 1: <https://eventouriterazione1.netlify.app/>

Iterazione 2: <https://eventouriterazione2.netlify.app/>

Iterazione 3: <https://eventouriterazione3.netlify.app/>

IMPORTANTE!

È importante che per l’uso della iterazione 1 venga usato il server api presente in esso, e così via per la 2 e la 3.

Nel documento “Credenziali di accesso” sono esplicitate le credenziali di accesso per la prova delle funzionalità.

# Iterazione 0

## Requisiti utente

L’azienda milanese evenTOUR Italia s.p.a. ha deciso di collaborare con l’Università degli Studi di Bergamo per fornire a tutti gli utenti sul territorio italiano una piattaforma per la prenotazione di eventi sparsi lungo tutto il territorio resi disponibili da associazioni ed enti/organizzazioni.

Gli eventi disponibili sulla piattaforma possono essere sia gratuiti che a pagamento, a discrezione dell’ente che lo ha inserito, con, in quest’ultimo caso, modalità di pagamento online che si basano su circuiti nazionali come Bancomat o carte di credito dei maggiori circuiti internazionali quali Visa, Mastercard, American Express.

All’interno della piattaforma esistono differenti ruoli che interagiscono con l’evento e con altri ruoli, quali utenti, manager (enti organizzatori), ticket inspector e administrator.

Gli utenti, per poter prenotare qualsiasi tipo di evento, valutare gli eventi a cui hanno partecipato, visualizzare gli eventi sulla base di differenti filtri o sfruttare la potenza dell’algoritmo evenTour, su cui si basa la collaborazione e l’intero progetto, devono essere registrati alla piattaforma fornendo i propri dati quali:

* Nome e cognome
* Sesso
* Data di nascita (l’utente deve essere maggiorenne)
* Luogo di residenza
* Tipi di evento preferiti
* Username, mail e password di accesso

Per quanto riguarda i manager, essi hanno la possibilità di gestire diversi eventi illimitatamente tramite il pagamento di una quota d’iscrizione per il mantenimento del proprio status e tramite la registrazione sulla piattaforma, fornendo i propri dati quali:

* Nome e cognome
* Data di nascita (il manager deve essere maggiorenne)
* Luogo di residenza
* Partita IVA
* Ragione sociale
* Mail e password

Oltre alle funzionalità precedentemente descritte, il manager ha anche la possibilità di visualizzare un report relativo a tutti gli eventi passati con dati relativi alla valutazione media del singolo evento e a tutto ciò che è relativo alla affluenza e alla parte economica legata alla manifestazione.

I ticket inspectors sono dei dipendenti, associati a vari eventi, che permettono il controllo delle prenotazioni all’ingresso dell’evento. Anche loro possono accedere alla piattaforma digitando un codice e una password che vengono inviati a loro tramite comunicazione da parte del manager e della piattaforma. Il controllo delle prenotazioni avviene tramite uno strumento smart di lettura di codici QR. L’utilizzo della pagina associata al controllo dei biglietti deve essere molto semplice e intuitiva anche per l’utilizzo su dispositivi mobili.

Gli administrator sono i veri e propri amministratori della piattaforma: essi gestiscono le richieste di iscrizione da parte dei manager e verificano che i dati inseriti al momento dell’iscrizione siano relativi a enti reali esistenti sul territorio, andando ad escludere le iscrizioni potenzialmente dannose o false, che non permettono alla piattaforma di proseguire nel suo compito nel migliore dei modi. Controllano altresì il pagamento delle quote di iscrizione o di rinnovo. Possono, a loro discrezione, attribuire dei malus ai manager che non rispettano i termini di servizio imposti dall’azienda.

Gli utenti, dopo una fase di registrazione e un successivo login, possono abilitare la funzionalità legata alla newsletter grazie alla quale sono sempre a conoscenza delle migliori proposte e delle novità riguardo a eventi già osservati in precedenza, nonché alle questioni organizzative di eventi già prenotati.

## Tool Chain

Per la realizzazione del software presentato in questo report sono stati utilizzati i seguenti tool:

* **Modellazione**
  + Draw.io[[[3]](#footnote-3)]: utilizzato per la modellizzazione di Use Case Diagram, Architecture Diagram, Deployment Diagram e Component Diagram.
  + Enterprise Architect (trial edition)[[[4]](#footnote-4)]: test di modellizzazione Class Diagram
  + UML Lab (trial edition)[[[5]](#footnote-5)] e CDA[[[6]](#footnote-6)]: Class Diagram
* **Implementazione software**
  + Eclipse: IDE per l’implementazione del codice Java lato server (realizzazione dell’API) tramite SpringBoot
  + Visual Studio Code: Editor per lo sviluppo dell’interfaccia web tramite il framework Vue.js
  + MongoDB Compass e Studio3T (academic license): Strumenti per la gestione del database MongoDB presente su un cluster Atlas.
* **Analisi del software**
  + NightWatch.js: per l’analisi dinamica del codice javascript
  + Eslint: per l’analisi statica delle componenti Vue
  + JUnit: per l’analisi dinamica del codice Java
  + Spotbugs e PMD: per l’analisi statica del codice Java
  + JArchitect: report per la qualità del codice
* **Altri tools**
  + Discord: canale di discussione del progetto
  + GitHub: per il versioning con interfaccia grafica
  + Microsoft Word: per la scrittura della documentazione
  + JSON-server e faker.js: per la generazione randomica di dati di test
  + Postman: piattaforma API per la gestione delle API

## Use Cases

### Descrizione testuale

|  |  |
| --- | --- |
| **Nome** | UC1: Visualizzazione Eventi |
| **Descrizione** | Mostra tutti gli eventi disponibili |
| **Attori** | / |
| **Obiettivi** | Mostrare gli eventi disponibili |
| **Precondizioni** | / |
| **Passi** | 1. L’utente entra nella pagina di visualizzazione degli eventi 2. Il sistema stampa a video tutti gli eventi disponibili 3. L’utente può selezionare un evento di interesse 4. Il sistema mostra altri dettagli relativi a esso |
| **Casi d’uso associati / flussi alternativi** | Generalizzazione:            UC1.1: Visualizzazione per data            UC1.2: Visualizzazione per luogo |
| **Postcondizioni** | / |

|  |  |
| --- | --- |
| **Nome** | UC1.1: Visualizzazione per data |
| **Descrizione** | Mostra tutti gli eventi disponibili ordinati per data ascendente o discendente |
| **Attori** | User |
| **Obiettivi** | Mostrare gli eventi disponibili in un certo ordine |
| **Precondizioni** | / |
| **Passi** | 1. L’utente entra nella pagina di visualizzazione degli eventi 2. Il sistema stampa a video tutti gli eventi disponibili in ordine crescente di data (dal più vicino al più lontano) 3. L’utente sceglie l’ordine con il quale mostrare gli eventi (dal più vicino al più lontano come date o viceversa) 4. Il sistema mostra a video gli eventi ordinati secondo l’ordine richiesto 5. L’utente può selezionare un evento di interesse 6. Il sistema mostra altri dettagli relativi a esso |
| **Casi d’uso associati / flussi alternativi** | Specializzazione UC1: Visualizzazione Eventi |
| **Postcondizioni** | / |

|  |  |
| --- | --- |
| **Nome** | UC1.2: Visualizzazione per luogo |
| **Descrizione** | Mostra tutti gli eventi disponibili ordinati per luogo in ordine alfabetico o inverso |
| **Attori** | User |
| **Obiettivi** | Mostrare gli eventi disponibili svolti in luoghi ordinati alfabeticamente |
| **Precondizioni** | / |
| **Passi** | 1. L’utente entra nella pagina di visualizzazione degli eventi 2. Il sistema stampa a video tutti gli eventi disponibili in ordine crescente di data (dal più vicino al più lontano) 3. L’utente richiede l’ordinamento crescente (alfabetico) o decrescente (alfabetico inverso) per regione, provincia, comune 4. Il sistema mostra a video gli eventi relativi all’ordine richiesto 5. L’utente può selezionare un evento di interesse 6. Il sistema mostra altri dettagli relativi a esso |
| **Casi d’uso associati / flussi alternativi** | Specializzazione UC1: Visualizzazione Eventi |
| **Postcondizioni** | / |

|  |  |
| --- | --- |
| **Nome** | UC2: Login |
| **Descrizione** | L’utente può accedere al suo account per effettuare prenotazioni, se utente, o altre funzioni, se manager o ticket inspector |
| **Attori** | User, Manager, Ticket Inspector |
| **Obiettivi** | Accedere alla propria area privata nel sito per svolgere funzioni |
| **Precondizioni** | Essere registrato |
| **Passi** | 1. L’attore entra nella pagina di login 2. L’attore compila il campo “username” 3. L’attore compila il campo “password” 4. L’attore richiede di essere autenticato con quelle credenziali 5. Il sistema autorizza l’accesso all’utente e lo porta nella sua area riservata   5.a. Nel caso di autenticazione di un User, il sistema riporta alla pagina di visualizzazione evento  5.b. Nel caso di autenticazione di un Manager, il sistema riporta alla pagina di visualizzazione eventi del manager  5.c. Nel caso di autenticazione di un Ticket Inspector, il sistema riporta alla pagina di scansione del QR |
| **Casi d’uso associati / flussi alternativi** | / |
| **Postcondizioni** | L’utente è autenticato |

|  |  |
| --- | --- |
| **Nome** | UC2.2: Login Errato |
| **Descrizione** | L’utente, nell’accesso alla piattaforma, non viene autenticato |
| **Attori** | User, Manager, Ticket Inspector |
| **Obiettivi** | Non permettere l’accesso alla propria area privata nel sito a causa di errori di autenticazione |
| **Precondizioni** | Essere registrato |
| **Passi** | 1. L’attore entra nella pagina di login 2. L’attore compila il campo “username” 3. L’attore compila il campo “password” 4. L’attore richiede di essere autenticato con quelle credenziali 5. Il sistema non autorizza l’accesso all’utente e segnala l’errore   5.a. Nel caso di autenticazione di un User, il sistema riporta alla pagina di login segnalando l’errore e l’utente ha la possibilità di ritentare l’accesso.  5.b. Nel caso di autenticazione di un Manager, il sistema riporta alla pagina di login segnalando le credenziali come errate e il manager ha la possibilità di ritentare l’accesso.  5.c. Nel caso di autenticazione di un Ticket Inspector, il sistema riporta alla pagina di accesso. |
| **Casi d’uso associati / flussi alternativi** | <<extend>> UC2: Login |
| **Postcondizioni** | Non essere autenticato |

|  |  |
| --- | --- |
| **Nome** | UC3: Sign Up |
| **Descrizione** | Permette la registrazione di un visitatore alla piattaforma |
| **Attori** | User |
| **Obiettivi** | Permettere di registrarsi alla piattaforma |
| **Precondizioni** | L’utente non deve essere già registrato |
| **Passi** | 1. L’utente entra nella pagina di registrazione alla piattaforma 2. L’utente compila i dati anagrafici e di autenticazione (mail e password) 3. Il sistema verifica che non esistano account associati alla stessa mail 4. Il sistema salva i dati dell’utente nel database 5. Il sistema segnala all’utente l’avvenuta registrazione alla piattaforma e riporta alla pagina di visualizzazione eventi |
| **Casi d’uso associati / flussi alternativi** | / |
| **Postcondizioni** | L’utente è registrato, se la registrazione rispetta i controlli fatti |

|  |  |
| --- | --- |
| **Nome** | UC4: Prenotazione Evento |
| **Descrizione** | L’utente può prenotare il posto per un evento o rimuovere una prenotazione da un evento già prenotato. |
| **Attori** | User |
| **Obiettivi** | Prenotare un posto per un evento da parte di un utente registrato |
| **Precondizioni** | * L’utente è loggato al sito * L’evento ha posti disponibili * L’evento avviene in una data futura * L’utente non è già prenotato all’evento in caso di prenotazione. * L’utente è già prenotato all’evento in caso di disdetta della prenotazione. * L’utente si trova nella pagina di visualizzazione degli eventi |
| **Passi** | 1.a. Prenotazione a un evento:  1.a.1. L’utente richiede la prenotazione all’evento nella pagina di visualizzazione dell’evento  1.a.2. Il sistema controlla le condizioni richieste  1.a.3. Il sistema, dopo aver verificato la disponibilità del posto richiesto, prenota il posto per l’evento e aggiorna il database  1.a.4. Il sistema conferma la prenotazione del posto richiesto  1.b. Disdetta di una prenotazione  1.b.1. L’utente richiede la disdetta della prenotazione all’evento nella pagina di visualizzazione dell’evento        1.b.2. Il sistema disdice la prenotazione il posto per l’evento e aggiorna il database  1.b.3. Il sistema conferma la disdetta della prenotazione del posto richiesto |
| **Casi d’uso associati / flussi alternativi** | <<Include>> UC2: Login |
| **Postcondizioni** | L’utente è prenotato all’evento e ha un posto riservato nel caso di prenotazione |

|  |  |
| --- | --- |
| **Nome** | UC5: Scan QR Code |
| **Descrizione** | Il sistema permette al Ticket Inspector di scansionare il QR presente sulla prenotazione di un utente e autorizzarne l’accesso all’evento |
| **Attori** | Ticket Inspector |
| **Obiettivi** | Validare una prenotazione di un utente a un evento |
| **Precondizioni** | Il Ticket Inspector deve essere autenticato |
| **Passi** | 1. Il Ticket Inspector scansiona con lo scanner il QR presente sulla prenotazione 2. Il sistema verifica che la prenotazione sia valida      3.a. Il sistema segnala, in caso di esito positivo, l’accesso consentito all’utente che ha prenotato     3.b. Il sistema segnala, in caso di esito negativo, l’accesso rifiutato all’utente per prenotazione non valida |
| **Casi d’uso associati / flussi alternativi** | <<Include>> UC2: Login |
| **Postcondizioni** | / |

|  |  |
| --- | --- |
| **Nome** | UC6: Inserimento Evento |
| **Descrizione** | Il sistema permette a un manager la creazione di un evento per l’inserimento di esso sulla piattaforma |
| **Attori** | Manager |
| **Obiettivi** | Creazione di un evento da parte del manager |
| **Precondizioni** | Il Manager deve essere autenticato |
| **Passi** | 1. Il Manager richiede l’inserimento di un evento al sistema 2. Il sistema mostra una pagina da compilare con i dati relativi all’evento da inserire 3. Il Manager compila i dati relativi all’evento 4. Il sistema verifica la correttezza dei dati inseriti (data futura, ad esempio)   5.a. Il sistema, in caso di esito positivo, segnala l’avvenuto inserimento dell’evento sulla piattaforma  5.b. Il sistema, in caso di esito negativo, segnala gli errori commessi nell’inserimento dell’evento |
| **Casi d’uso associati / flussi alternativi** | <<Include>> UC2: Login |
| **Postcondizioni** | L’evento è presente sulla piattaforma |

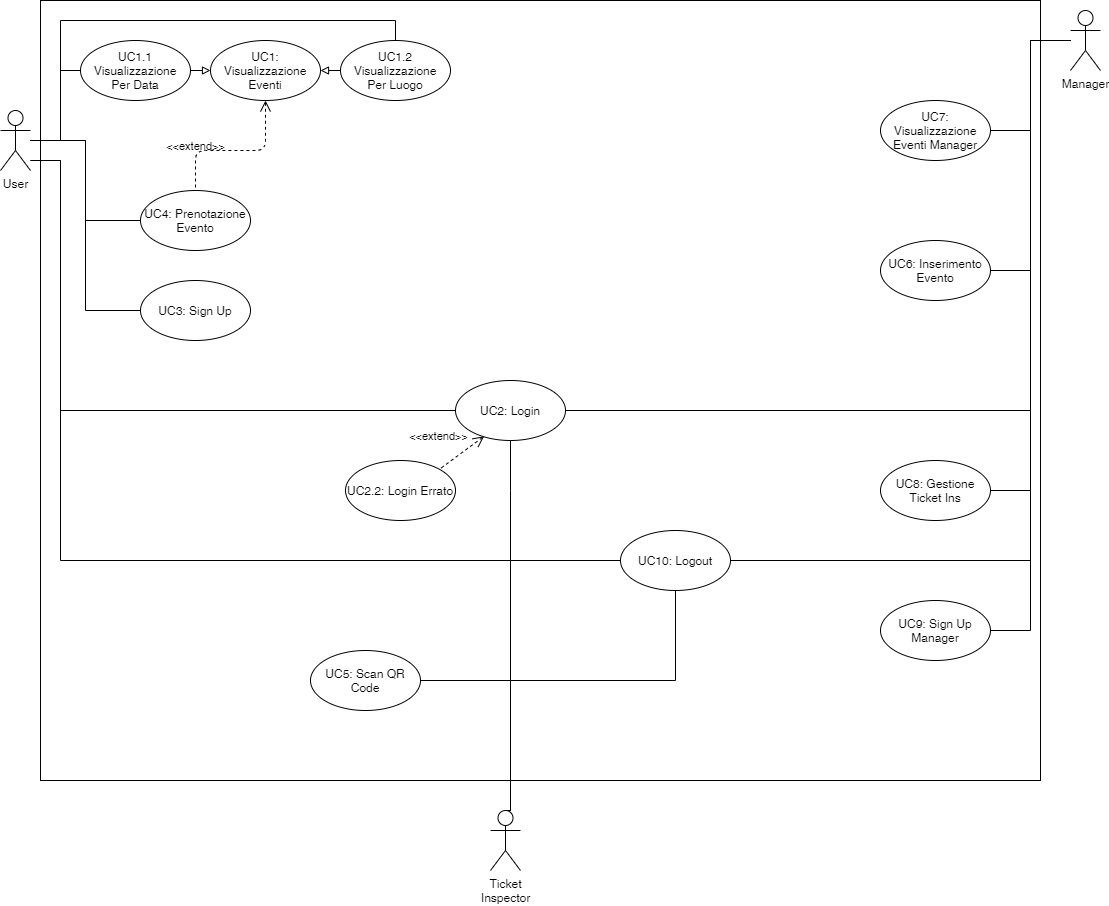
|  |  |
| --- | --- |
| **Nome** | UC7: Visualizzazione Eventi Manager |
| **Descrizione** | Il sistema permette la visualizzazione degli eventi relativi al manager |
| **Attori** | Manager |
| **Obiettivi** | Mostrare gli eventi inseriti da un manager |
| **Precondizioni** | Il Manager deve essere autenticato |
| **Passi** | 1. Il sistema mostra i dati di base degli eventi relativi al manager   2.1. Il manager può richiedere la rappresentazione dettagliata di un evento  2.2. Il sistema mostra i dettagli dell’evento richiesto |
| **Casi d’uso associati / flussi alternativi** | <<Include>> UC2: Login |
| **Postcondizioni** | / |

|  |  |
| --- | --- |
| **Nome** | UC8: Gestione Ticket Inspector |
| **Descrizione** | Il sistema permette al Manager l’organizzazione dei propri Ticket Inspector. |
| **Attori** | Manager |
| **Obiettivi** | Organizzare i Ticket Inspector di un Manager |
| **Precondizioni** | Il Manager deve essere autenticato |
| **Passi** | 1. Il Manager richiede l’accesso alla pagina di visualizzazione di un evento dettagliato 2. Il sistema mostra la pagina dove poter visualizzare i Ticket Inspector di un evento. |
| **Casi d’uso associati / flussi alternativi** | <<Include>> UC2: Login |
| **Postcondizioni** | / |

|  |  |
| --- | --- |
| **Nome** | UC9: Sign Up Manager |
| **Descrizione** | Il sistema permette all’utente di registrarsi come Manager (organizzatore di eventi) |
| **Attori** | Manager |
| **Obiettivi** | Registrare un Manager come tale |
| **Precondizioni** | Il Manager non è ancora registrato |
| **Passi** | 1. L’utente richiede la registrazione alla piattaforma come Manager 2. Il sistema mostra la pagina di registrazione alla piattaforma come Manager 3. L’utente compila tutti i dati anagrafici e dell’organizzazione 4. L’utente richiede la registrazione con i dati inseriti 5. Il sistema controlla i dati inseriti   6.a.1. In caso di validazione corretta, il sistema conferma la richiesta di registrazione via mail al manager  6.b.1. In caso di validazione errata, il sistema richiede la modifica dei dati errati. |
| **Casi d’uso associati / flussi alternativi** | <<include>> UC2: Login |
| **Postcondizioni** | Il Manager è registrato se la registrazione è andata a buon fine |

### Use Case diagram

Il diagramma seguente mostra i casi d’uso ad alta priorità, le fondamenta su cui basare l’intero progetto. Le funzionalità principali sono l’autenticazione di un utente, di un organizzatore e di un ticket inspector, una prenotazione molto basilare di un evento, la visualizzazione degli eventi per data o per luogo (ordine alfabetico della regione, a seguire ordine alfabetico della provincia e infine ordine alfabetico del comune), la registrazione alla piattaforma da parte di utente o manager, l’inserimento da parte del manager di un evento con la possibile gestione dei propri ticket inspector associati all’evento e, per il ticket inspector, la possibilità di scansionare un codice QR della prenotazione e quindi permettere o negare l’accesso all’evento.



Modello 2.1. Use Case Diagram

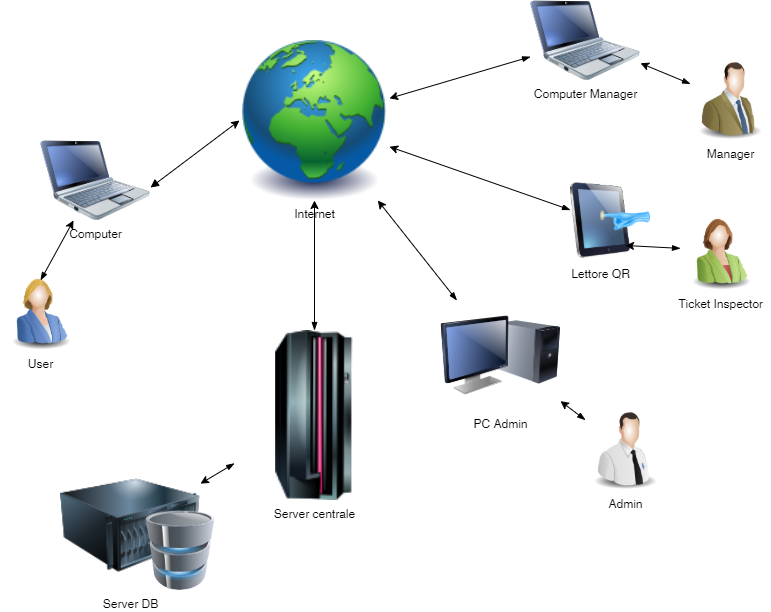
## Architettura

Data la natura del progetto, si decide di realizzare una piattaforma che sfrutti tecnologie web che comunichino attraverso internet.

### 

### Architecture model

Il modello architetturale di massima mostra la possibilità per un utente o per un manager di accedere attraverso un computer a internet, tipicamente tramite l’uso di un browser a suo piacimento. Ciò non vieta tuttavia di usare un dispositivo mobile per la navigazione nel portale. Si è pensato di collocare su un server centrale il server web che fornisce l’interfaccia utente e il server che fornisce le API necessarie al corretto funzionamento della piattaforma. Il ticket inspector, essendo in un luogo fisico in mobilità e avendo necessità di scansionare dei codici QR contenuti sulle prenotazioni degli utenti, potrà usare tablet o smartphone connessi a Internet per l’espletazione della sua funzione. Il server API comunicherà poi con un DB server che fornirà e permetterà una gestione separata della banca dati su cui fa da fondamenta alla piattaforma.



Modello 2.2. Hardware Architecture diagram

### Deployment diagram

Nel modello 2.2 e 2.3 sono mostrati l’Hardware Architecture Diagram e il Deployment Diagram del sistema progettato per lo sviluppo della piattaforma EvenTour.

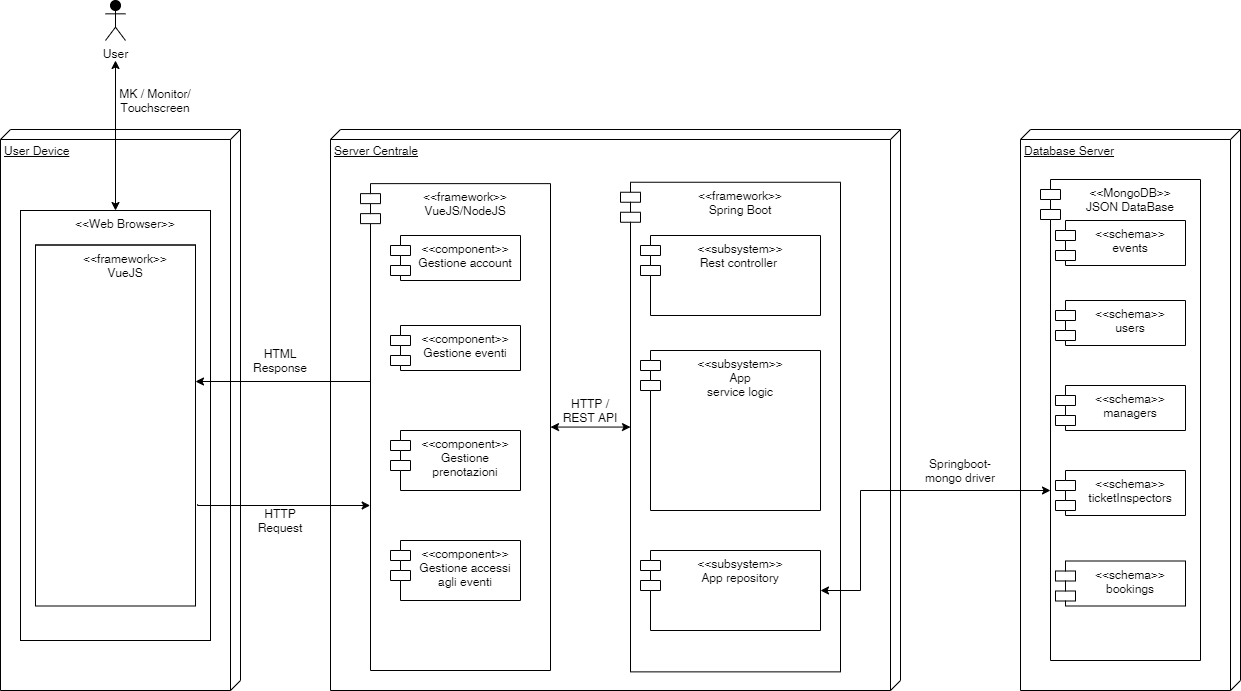
Si può osservare che si tratta di un’architettura ***Three Tiers***:

1. A sinistra si individua lo *User Device*, ovvero il dispositivo su cui è presente un browser web che permetta la fruizione dei contenuti della piattaforma. Si nota che esso sfrutterà il framework di Vue.js per la comunicazione con il server web messo a disposizione sempre dal framework e gestito da esso, basato su Node.js;

2. Nella parte centrale individuiamo il secondo layer, che contiene sia il server web sopracitato sia il server delle API messe a disposizione per l’uso della piattaforma. Questa scelta è stata fatta per mantenere una comunicazione sicura tra web server e api di accesso ai dati, avere sulla stessa macchina il controllo di tutte le operazioni e immaginando che l’impresa EvenTOUR abbia a disposizione un solo server.

3. Nella parte destra del modello 2.3 si individua il DB server (MongoDB Atlas) che contiene il cluster con presenti tutti i dati di eventi, utenti, manager, prenotazioni e risorse ausiliarie.

Per la comunicazione tra il dispositivo da cui è connesso l’utente e il server centrale si utilizzano meccanismi di richieste HTTP e risposte HTML, le cui pagine vengono generate dinamicamente dal server web e restituite al richiedente. La comunicazione tra web server e server API viene effettuata attraverso HTTP e REST API. La comunicazione con il database server sfrutta i driver disponibili per Spring Boot per la comunicazione con database non relazionali come mongoDB.



Modello 2.3. Deployment Diagram

### Database schema

Si è scelto di utilizzare un database non relazionale perché più indicato per applicazioni che fanno un uso massiccio di moli di dati grandi, per una latenza d’utilizzo bassa e per avere modelli di dati flessibili. Nella iterazione iniziale si è scelto di inserire le collezioni di dati aventi struttura specificata dalla Tabella 2.1. Per quanto concerne le località di svolgimento degli eventi o di residenza dei manager o degli utenti, saranno approfonditi i campi nella Tabella 2.2.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Collezione** | **Campo** | **Tipo di dato** | **Indice** | **Descrizione** |
| events | \_id | ObjectId | ✔️ Segno Di Spunta Emoji | ID univoco generato all’inserimento nel database |
| title | String |  | Titolo dell’evento |
| description | String |  | Descrizione dell’evento |
| dataOra | Date |  | Data e ora dell’evento |
| location | Object |  | Struttura spiegata successivamente, contiene il luogo nel quale viene svolto l’evento |
| types | Array |  | Vettore di stringhe con i codici dei tipi dell’evento |
| managerId | ObjectId |  | ID univoco del manager che ha creato l’evento |
| urlImage | String |  | URL dell’immagine di “copertina” dell’evento |
| totSeat | Int32 |  | Posti totali disponibili per l’evento |
| freeSeat | Int32 |  | Posti ancora liberi per l’evento |
| price | Double |  | Prezzo dell’evento (0 se gratuito) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Collezione** | **Campo** | **Tipo di dato** | **Indice** | **Descrizione** |
| managers | \_id | ObjectId | ✔️ Segno Di Spunta Emoji | ID univoco generato all’inserimento nel database |
| mail | String |  | Mail (univoca) di registrazione del manager |
| password | String |  | Password di accesso |
| name | String |  | Nome del manager |
| surname | String |  | Cognome del manager |
| dateOfBirth | Date |  | Data di nascita del manager |
| residence | Object |  | Struttura spiegata successivamente, contiene il luogo nel quale risiede il manager |
| codicePIVA | String |  | Partita IVA del manager |
| ragioneSociale | String |  | Ragione sociale |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Collezione** | **Campo** | **Tipo di dato** | **Indice** | **Descrizione** |
| users | \_id | ObjectId | ✔️ Segno Di Spunta Emoji | ID univoco generato all’inserimento nel database |
| username | String |  | Username per l’accesso alla piattaforma previa registrazione |
| mail | String |  | Mail (univoca) di registrazione dell’utente |
| password | String |  | Password di accesso |
| name | String |  | Nome dell’utente |
| surname | String |  | Cognome dell’utente |
| dateOfBirth | Date |  | Data di nascita dell’utente |
| sex | String |  | Sesso dell’utente |
| residence | Object |  | Struttura spiegata successivamente, contiene il luogo nel quale risiede l’utente |
| types | Array |  | Vettore di stringhe con i codici dei tipi di evento preferiti |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Collezione** | **Campo** | **Tipo di dato** | **Indice** | **Descrizione** |
| bookings | \_id | ObjectId | ✔️ Segno Di Spunta Emoji | ID univoco generato all’inserimento nel database |
| userID | ObjectId |  | Username per l’accesso alla piattaforma previa registrazione |
| eventID | ObjectId |  | Mail (univoca) di registrazione dell’utente |
| prenotedSeat | Int32 |  | Password di accesso |
| come | Boolean |  | Nome dell’utente |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Collezione** | **Campo** | **Tipo di dato** | **Indice** | **Descrizione** |
| ticketInspectors | \_id | ObjectId | ✔️ Segno Di Spunta Emoji | ID univoco generato all’inserimento nel database |
| code | String |  | Codice generato alla creazione per identificare il ticket inspector |
| password |  |  | Password di accesso |
| fullName | String |  | Nome e cognome del ticket inspector |
| eventId | ObjectId |  | ID univoco dell’evento a cui è associato il ticket inspector |

Tabella 2.1 Collezioni presenti su mongodb con descrizione dei singoli campi

Per quanto concerne gli indici, le spunte nere sono indici, con ordine ascendente, relativi all’identificativo univoco di ogni oggetto. Le spunte colorate di blu, verde e rosso identificano invece indici ausiliari per verificare che non esistano più manager con la stessa mail di iscrizione, più utenti con la stessa mail di iscrizione o più ticket inspector con coppia <code, eventId> uguale.

|  |  |  |  |
| --- | --- | --- | --- |
| **Oggetto** | **Campo** | **Tipo di dato** | **Descrizione** |
| location/residence | locality | String | Località, via e civico |
| city | String | Città o paese |
| cap | String | Codice di avviamento postale del comune |
| provincia | String | Provincia del comune specificato |
| sigla | String | Sigla della provincia |
|  | regione | String | Regione di appartenenza |
|  | lat | Double | Latitudine in gradi |
|  | lng | Double | Longitudine in gradi |

Tabella 2.2 Descrizione dei campi presenti negli oggetti location e residence

Si riportano inoltre di seguito i codici e i relativi nomi associati ai tipi di evento. Essi saranno utilizzati sia nella API sia nell’applicativo web (un evento identificabile come un concerto punk rock sarà identificato dal tipo 1.2.3):

1. concerto
   1. musica commerciale
      1. pop
      2. disco/dance
      3. altro
   2. rock
      1. hard rock
      2. metal
      3. punk
      4. altro
   3. rap
      1. trap
      2. freestyle
      3. battle
   4. classica
      1. lirica
      2. orchestrale
      3. strumentale
      4. gospel
2. teatro
   1. musical
   2. commedia
   3. opera lirica
   4. prosa
   5. magia/cabaret
   6. tragedia
   7. spettacolo di danza
3. altri eventi
   1. sfilata
   2. proiezione speciale (non solo al cinema)
   3. evento sportivo
   4. firmacopie/presentazione
   5. circo

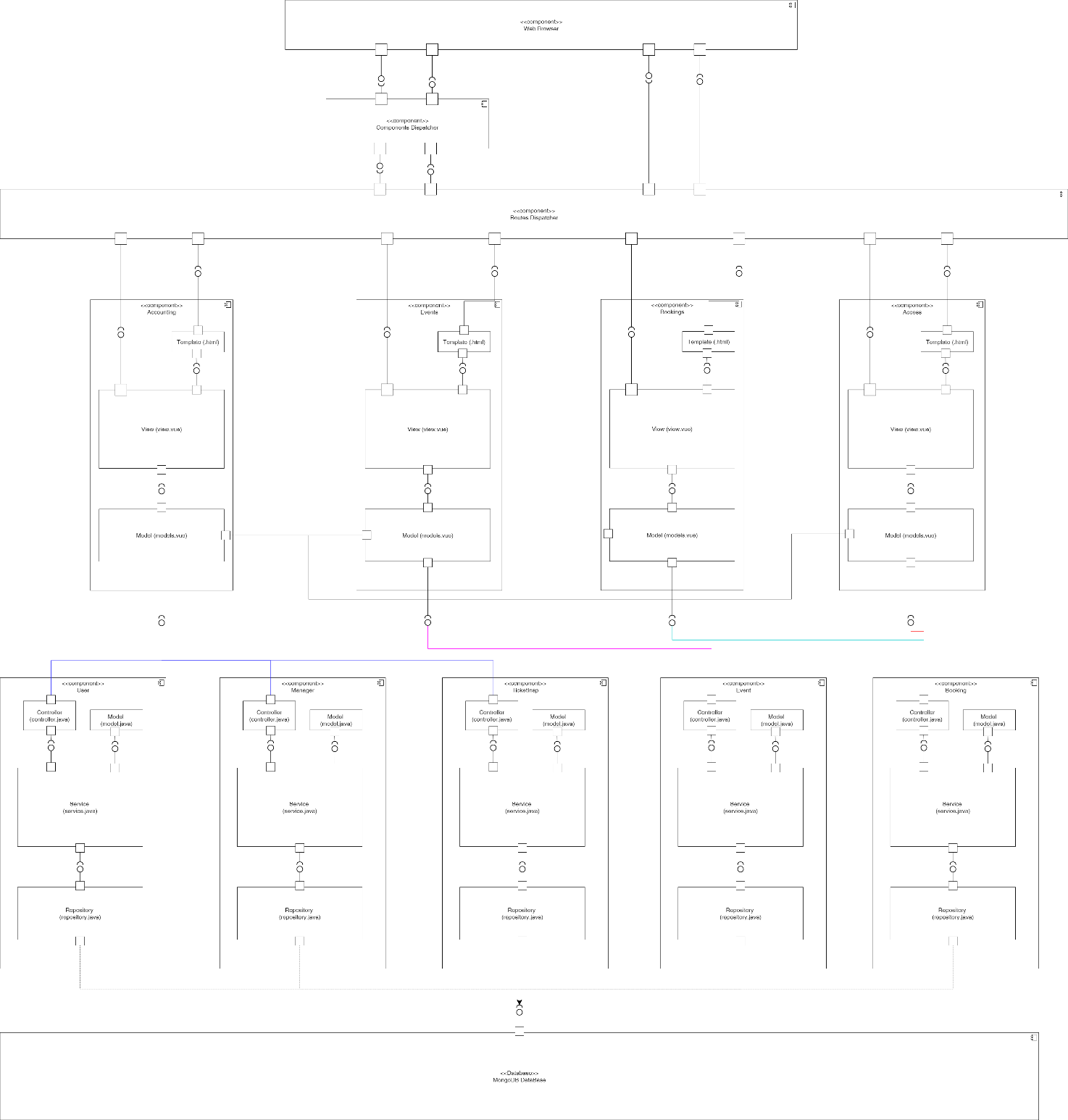
### Component diagram

Il diagramma a componenti riportato successivamente (visibile nella sua interezza nella cartella DOCUMENTAZIONE/model\_it0) permette di visualizzare i componenti che compongono funzionalmente le interconnessioni e le interfacce di comunicazione tra i diversi componenti.

L’utente accede attraverso un web browser alla piattaforma. Il web browser comunica direttamente con il routes dispatcher, che fornisce i percorsi di accesso alle varie pagine web che compongono EvenTour, e con il component dispatcher, che permette il salvataggio di componenti che hanno un impatto maggiore sul sistema e che vengono caricati frequentemente. Il component dispatcher ha bisogno anch’esso di comunicare con il routes dispatcher per l’accesso alle risorse. L’interazione è bidirezionale.

Per accedere alla pagina web richiesta dall’utente, il routes dispatcher richiede al macro-componente, Accounting, Events, Bookings e Access, la vista o dei singoli componenti tramite la view del componente vue oppure tramite un template in formato html che definisce la struttura e che comunica a sua volta con la vista in formato vue. Ogni vista, grazie al pattern architetturale MVVM, comunica direttamente e bidirezionalmente con il modello del macrocomponente. Essi comunicano tra loro tramite dei costrutti del View-Model, i quali permettono una comunicazione efficiente tra le due logiche.

Ogni model, oltre che a comunicare con i modelli degli altri macrocomponenti, richiede i dati o elaborazioni da altri componenti, presenti come API, quali Utenti, Manager, Eventi, Prenotazioni e Ticket inspectors. La struttura base di ogni componente citato è formata secondo il pattern CSR (Controller-Service-Repository), in cui il Controller comunica direttamente con funzionalità del Service, che elabora dati sia grazie alle strutture dei dati presenti nel database come esplicitate nei modelli sia grazie ai repository tramite l’uso di interfacce Java e driver. Ogni repository richiederà dati grezzi al MongoDB database. Seppur non espresso in termini grafici nello schema sottostante, ogni service può avere accesso a più di un repository grazie al pattern singleton di Spring che mantiene l’univocità delle istanze.



### Pattern architetturali

All’interno del progetto è identificabile il design pattern Singleton che in Spring non possiede tutte le caratteristiche. Se in linea di massima, il Singleton puro garantisce l’esistenza di una sola istanza di un oggetto per applicazione, anche il Singleton legato al framework Spring possiede tale caratteristica, ma questo vincolo può essere rilassato dato che Spring limita invece Singleton a un oggetto per contenitore, andando a creare solo un bean di ogni tipo per contesto applicativo. Quindi differisce dalla definizione standard di Singleton poiché un’applicazione può avere più di un contenitore Spring. Pertanto, più oggetti della stessa classe possono esistere in una singola applicazione se abbiamo più contenitori.

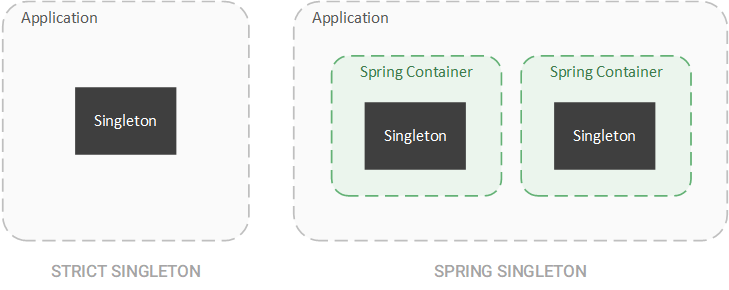


Figura 2.1 Springoboot Singleton pattern

Lo schema rappresenta il singleton visto dal framework Spring. Come standard, Spring crea tutti i Bean come singleton. Nel nostro progetto troviamo vari esempi di questo utilizzo; ecco qualche esempio:



Codice 2.1 Codice classe UserService



Codice 2.2 Codice classe BookingService

Come si può osservare dal codice delle classi UserService e BookingService, entrambe fanno riferimento tramite l’annotazione @Autowired allo stesso repository (UserRepository), inserendo lo stesso bean in entrambi i Service. I controller citati in precedenza contengono solamente il service legato alla classe base di riferimento; quindi, applica ancora una volta Singleton anche su questi componenti.

Un altro pattern architetturale sfruttato è legato al framework Vue.js, ossia MVVM, Model-View-ViewModel, variante del Presentation Model Design. Il pattern astrae lo stato di view e il suo comportamento. Esistono 4 componenti che identificano il pattern MVVM:

* Model
* View
* View-Model: intermediario tra la vista e il modello, responsabile della logica della vista, interagendo principalmente con il modello invocando i metodi delle classi del modello, fornendo i dati in una forma che la vista può usare facilmente.
* Binder: meccanismo che permette la costante sincronizzazione tra tra la View e il View-Model, garantendo che i dati modificati dall’utente tramite la View vengano automaticamente riportati nel View-Model e che le modifiche avvenute su quest’ultimo vengano mostrato all’utente tramite la vista.

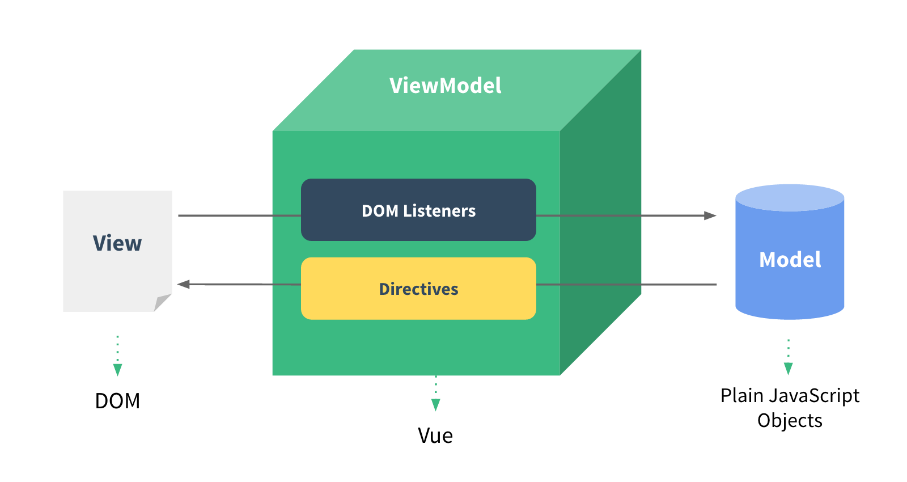


Figura 2.2 Schema modello MVVM

Un ulteriore pattern architetturale che si può trovare all’interno del progetto è legato all’architettura dei microservizi, in netta contrapposizione rispetto a quella monolitica, visto che è realizzata con componenti indipendenti che eseguono ciascun processo applicativo come un servizio che comunicano attraverso un’interfaccia ben definita che utilizza API, mantenendo tutti i vantaggi associati all’architettura quali agilità, scalabilità, flessibilità e semplicità di distribuzione. Di seguito uno schema che mostra l’utilizzo di un’architettura a microservizi, in particolare quella legata al concetto di API Gateway:

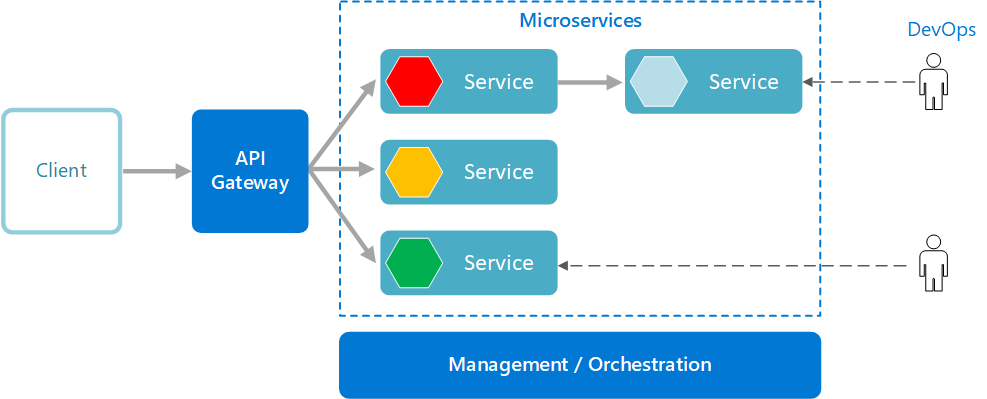


Figura 2.3 Microservice structure

Il client si interfaccia con l’API Gateway per effettuare delle richieste e quest’ultimo richiamerà i servizi creati dagli sviluppatori per rispondere alle esigenze del client. Nel nostro caso, i servizi sono quelli legati agli utenti, alle prenotazioni, agli eventi, ai manager e ai ticket inspector.

# Iterazione 1 (implementazione Iterazione 0)

In questa iterazione si è implementata una prima parte del codice con riferimento alla modellizzazione espressa nella iterazione 0.

## Class diagram dell’API

Si mostrano ora tutti i digrammi delle classi suddivisi per ruolo.

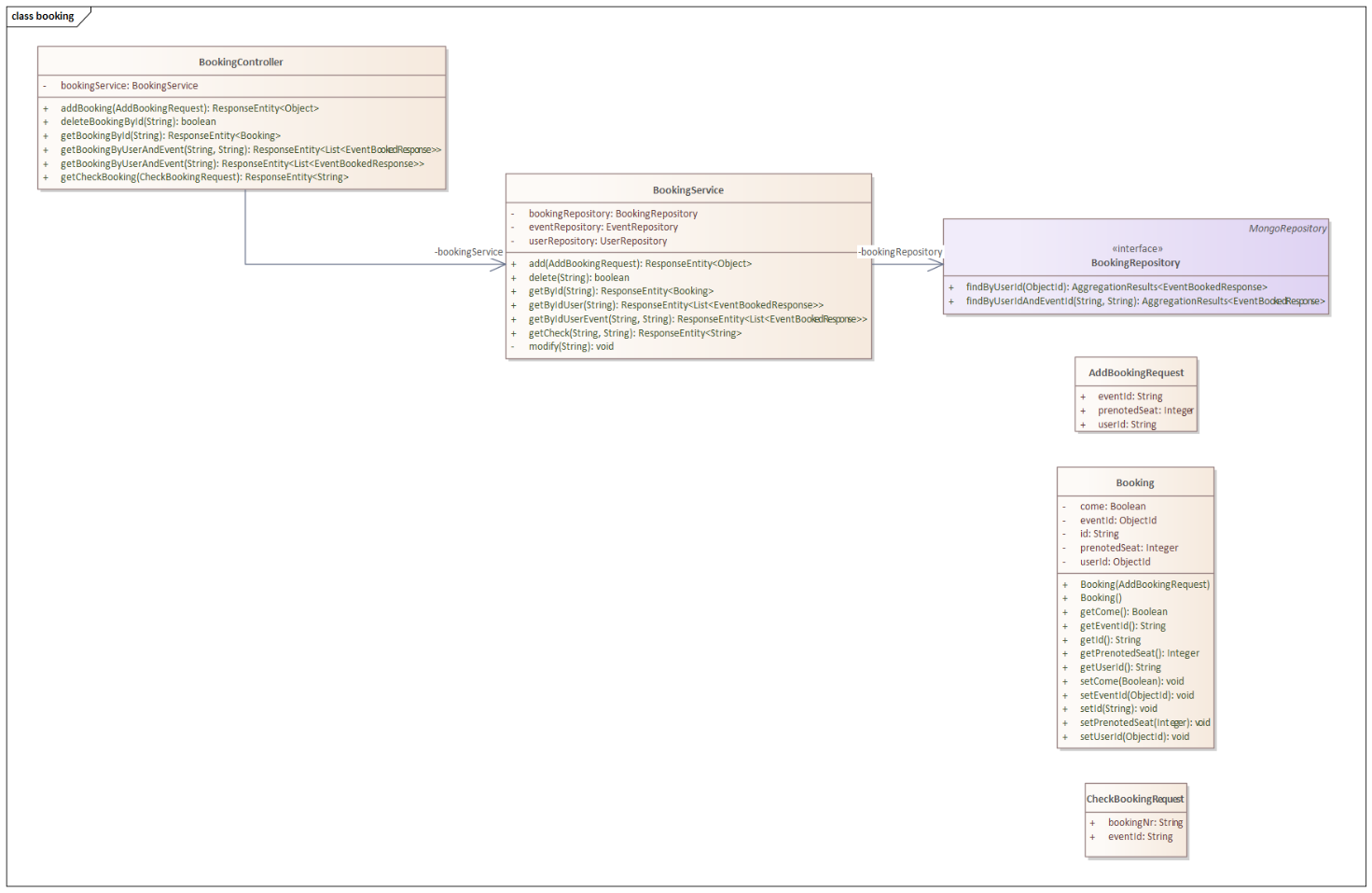


Modello 3.1 User Class Diagram

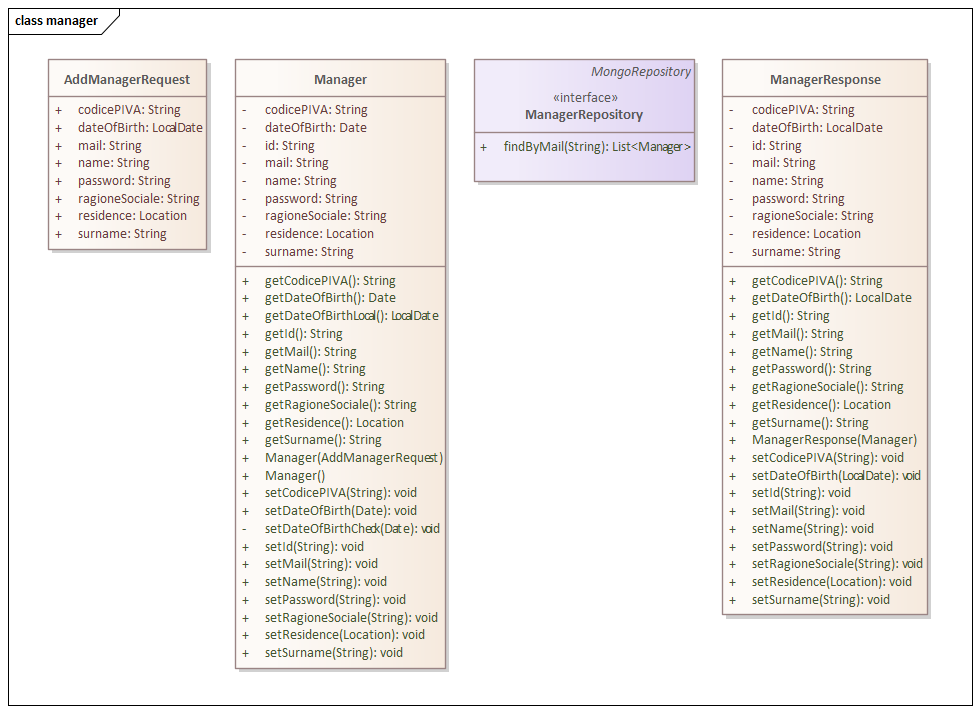
Immagine che contiene tavolo

Descrizione generata automaticamente

Modello 3.2 Event Class Diagram



Modello 3.3. Booking Class Diagram



Modello 3.4 Manager Class Diagram

Immagine che contiene testo

Descrizione generata automaticamente

Modello 3.5 Ticket Inspector Class Diagram

Si vede chiaramente attraverso i diversi class diagram e il diagramma dei package, riportato di seguito, che la struttura uscente fa riferimento a una architettura a microservizi, descritta al meglio nella sezione “Pattern architetturali”.

I campi delle classi dei package “model” sono stati configurati per essere di tipo private, con metodi get e set associati. I class diagram precedenti fanno riferimento a dei package costruiti ad hoc per la migliore visualizzazione della struttura e per mostrare le relazioni di ogni ruolo o struttura dati.

Questo meccanismo si ripercuote in tutte le iterazioni successive, con ampliamenti ai singoli metodi di ogni classe.



Modello 3.6 Package Diagram

## Definizione dei dati di test e di utilizzo

Per la generazione di dati di test e di utilizzo durante lo sviluppo del progetto, sono state effettuate varie generazioni di collezioni di test, per arrivare a circa 16.000 eventi, 76,862 prenotazioni, 5.000 utenti e 50 manager.

In prima istanza, l’idea del progetto era il mantenimento di un json server mock che generasse a ogni avvio nuovi dati. Ciò si ripercuoteva tuttavia sullo sviluppo, andando a complicare il testing e le “prove sul campo”. Si è quindi deciso di sfruttare un database non relazionale come MongoDB con server online (Mongo Atlas) e generare in caso di necessità i dati necessari tramite lo script sviluppato precedentemente. Il progetto dell’intero json-server può essere trovato nella cartella lib del progetto. In esso si fa uso, per la generazione di dati, di una API che fornisce tutti i dati sui comuni e locazioni (durante lo sviluppo essa è stata modificata dall’autore, con conseguente aggiornamento del codice sviluppato da noi) e di una libreria javascript per la generazione randomica di dati di vario tipo.

Si riporta nella pagina seguente la parte di codice che genera i 16000 eventi.



## Algoritmo evenTour

Lo scopo della versione iniziale dell’algoritmo progettato è quella di trovare un numero di eventi, definito dalla invocazione della API, che siano disponibili e che siano nella regione di residenza dell’utente. Essa, in questa versione, è stata solo progettata in quanto la vera e propria implementazione è stata fatta nella iterazione successiva.

La funzione, dopo aver analizzato i dettagli dell’utente che ne richiede l’eventour, ricerca tutti gli eventi della regione di appartenenza dell’utente che siano disponibili ordinandoli per data crescente ed esclude quelli già prenotati dall’utente e quelli che sono nella stessa data di quelli prenotati. Per ogni evento rimanente:

* + 1. Viene effettuato un controllo di raggiungimento della soluzione (sono stati trovati N eventi richiesti):
       1. se veritiero, viene restituita la soluzione,
       2. altrimenti viene selezionato l’evento se rispetta due condizioni:
* Almeno uno dei tipi di quell’evento è uguale a uno dei tipi preferiti dall’utente.
* La soluzione da restituire non contiene ancora eventi oppure l’ultimo evento inserito nella soluzione da restituire non ha la stessa data dell’evento che si vuole aggiungere alla soluzione.
  + 1. Se la soluzione non contiene N eventi e tutti gli eventi sono stati analizzati, allora viene restituito un errore.

Si riporta lo pseudocodice dell’algoritmo:



## Testing interfaccia grafica

### Analisi statica

* EsLint



Codice 3.1 File di configurazione EsLint per l’analisi statica nella parte client

### Analisi dinamica

* NightWatch



Codice 3.2 Test dinamico effettuato sulla classe User

Sono stati effettuati ulteriori test riguardo le classi dei Manager e dei Ticket Inspector, tali test non sono rappresentati poiché brevi e relativi solo all’accesso da parte del tipo di account che si è registrato. In tutti i casi i risultati sono positivi e tutti gli assert sono stati confermati e validati. Ogni test ha un output per ogni classe testata che identifica quanti controlli sono stati effettuati e in quanto tempo.

## Testing API

### Analisi statica

Per quanto riguarda l’analisi statica effettuata sul server per l’esposizione delle API, dopo aver generato una copia di test in MongoDB Atlas e aver modificato la stringa di connessione nell’API, si sono riscontrati circa 25 bugs segnalati da SpotBugs e varie violazioni segnalate grazie a PMD.

Per quanto riguarda Spotbugs, i bug segnalati sono stati risolti partendo da quelli più critici per finire a quelli meno critici.

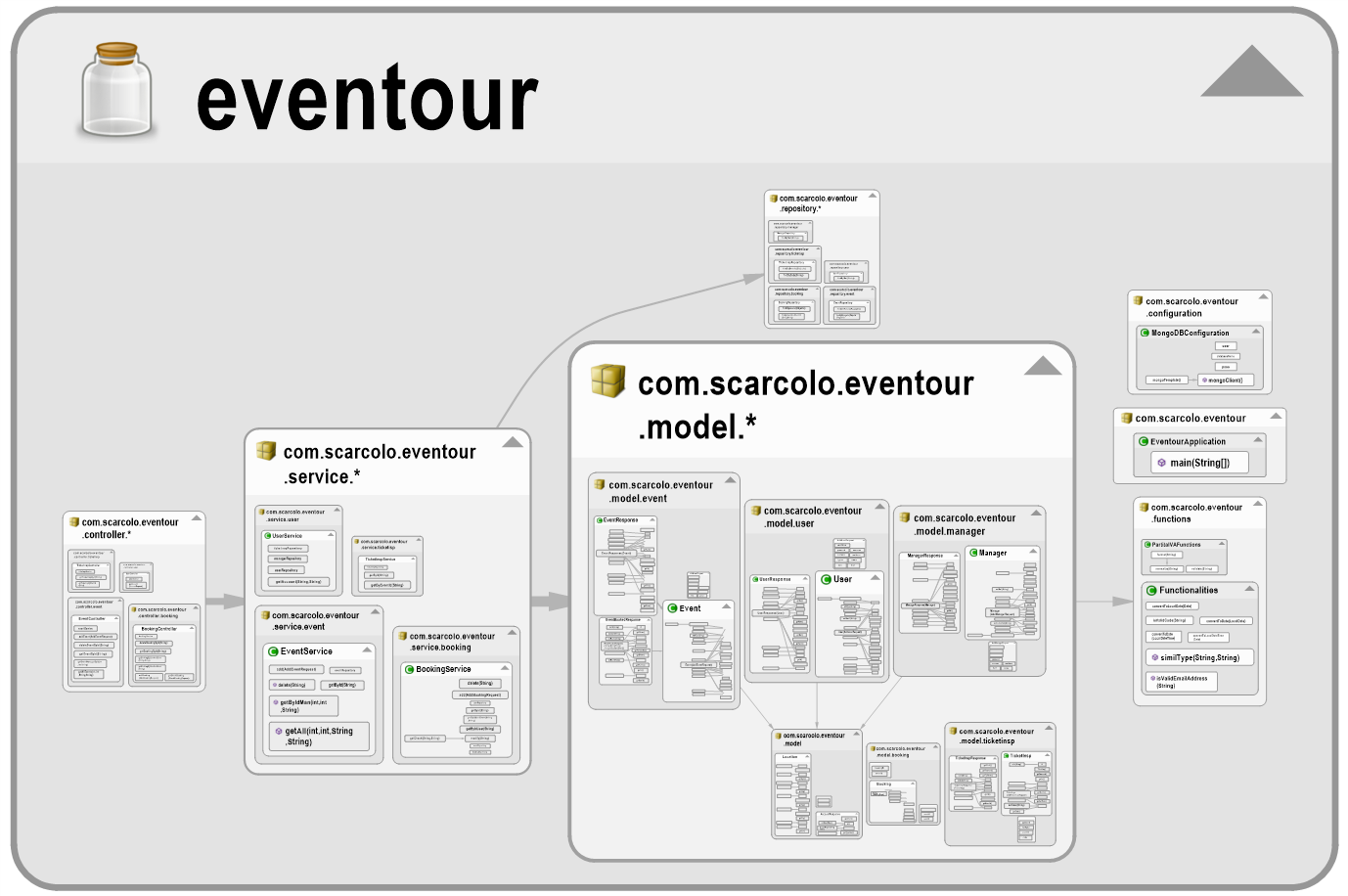
Per quanto riguarda PMD, le violazioni di livello Blocker e Critical (bandiera rossa e azzurra) sono stati risolti, mentre quelli di livello inferiore sono stati controllati e analizzati senza correggerli.

Immagine che contiene mappa

Descrizione generata automaticamente

Figura 3.1 Package progetto in Eclipse con visualizzazione dei bug segnalati da spotbugs e delle violazioni di livello alto mostrate

Il report di JArchitect evidenzia inoltre il grafo delle dipendenze mostrato di seguito:



### 

### Analisi dinamica

Per l’esecuzione dei test, dopo previa modifica della stringa di connessione contenuta nel file MongoDBConfiguration.java, nel package *com.scarcolo.eventour.configuration*, si deve avviare i test posizionandosi nella source folder dei test e con tasto destro, effettuare *Run As>JUnit Test*.

Questo vale per tutte le iterazioni

Tramite JUnit si sono svolti numerosi test, come vis

Immagine che contiene tavolo

Descrizione generata automaticamente

Immagine che contiene tavolo

Descrizione generata automaticamente

# Iterazione 2

Durante questa iterazione sono state introdotte numerose funzionalità quali ricerche di eventi più complete, possibilità di visualizzare un evenTour, pagamento di prenotazioni, visualizzazione degli eventi prenotati e gestione completa degli eventi da parte del manager.

## Use Cases

### Descrizione testuale

Sono stati definiti nuovi casi d’uso:

|  |  |
| --- | --- |
| **Nome** | UC1.3: Visualizzazione per tipo |
| **Descrizione** | Mostra tutti gli eventi disponibili di un tipo o insieme di tipi indicato/i |
| **Attori** | User |
| **Obiettivi** | Mostrare gli eventi disponibili del tipo/tipi scelto/i |
| **Precondizioni** | L’utente deve essere autenticato |
| **Passi** | * + - 1. L’utente entra nella pagina di visualizzazione degli eventi       2. Il sistema stampa a video tutti gli eventi disponibili in ordine crescente di data (dal più vicino al più lontano)       3. L’utente richiede la visualizzazione di eventi di uno o più tipi specifici       4. Il sistema mostra a video tutti gli eventi ordinati per data crescente appartenenti a uno dei tipi specificati       5. L’utente può selezionare un evento di interesse       6. Il sistema mostra altri dettagli relativi a esso e la possibilità di gestirne la prenotazione |
| **Casi d’uso associati / flussi alternativi** | Specializzazione UC1: Visualizzazione Eventi |
| **Postcondizioni** | / |

|  |  |
| --- | --- |
| **Nome** | UC1.4: Visualizzazione per preferenze |
| **Descrizione** | Mostra tutti gli eventi disponibili in base a uno o più tipi preferiti dall’utente |
| **Attori** | User |
| **Obiettivi** | Mostrare gli eventi disponibili di tipi preferiti |
| **Precondizioni** | L’utente deve essere autenticato |
| **Passi** | * + - 1. L’utente entra nella pagina di visualizzazione degli eventi       2. Il sistema stampa a video tutti gli eventi disponibili in ordine crescente di data (dal più vicino al più lontano)       3. L’utente richiede la visualizzazione di eventi secondo le sue preferenze       4. Il sistema mostra a video tutti gli eventi ordinati per data crescente con tipi preferiti dall’utente inseriti nella registrazione       5. L’utente può selezionare un evento di interesse       6. Il sistema mostra altri dettagli relativi a esso e la possibilità di gestirne la prenotazione |
| **Casi d’uso associati / flussi alternativi** | Specializzazione UC1: Visualizzazione Eventi |
| **Postcondizioni** | / |

|  |  |
| --- | --- |
| **Nome** | UC11: Impostazioni Utente |
| **Descrizione** | Permette di cambiare le impostazioni legate all’utente |
| **Attori** | User |
| **Obiettivi** | Cambiare impostazioni dell’utente |
| **Precondizioni** | L’utente deve essere loggato |
| **Passi** | 1. L’utente entra nella pagina delle impostazioni 2. Il sistema mostra tutte le informazioni legate all’utente 3. Se l’utente vuole modificare i dati    1. L’utente inserisce nei campi disponibili per la modifica i valori che intende modificare   3.2. L’utente salva le informazioni  3.3. Il sistema aggiorna le informazioni modificate nel  database |
| **Casi d’uso associati / flussi alternativi** | Flusso alternativo: L’utente può uscire dalla modifica delle impostazioni senza effettuare il salvataggio, perdendo così tutte le modifiche effettuate. |
| **Postcondizioni** | Le informazioni sono modificate, se l’utente ha effettuato cambiamenti. |

|  |  |
| --- | --- |
| **Nome** | UC12: Eventi Utente |
| **Descrizione** | Visualizzazione degli eventi prenotati da un utente e gestione di essi |
| **Attori** | User |
| **Obiettivi** | Visualizzazione degli eventi prenotati |
| **Precondizioni** | L’utente deve essere loggato |
| **Passi** | 1. L’utente accede alla pagina di visualizzazione degli eventi prenotati 2. Il sistema mostra tutti gli eventi prenotati dall’utente 3. L’utente può selezionare un evento tra quelli prenotati 4. Il sistema mostra altri dettagli relativi a esso e la possibilità di gestirne la prenotazione 5. In caso di cambio di alcune prenotazione, ritornando sulla pagina saranno riaggiornate le prenotazioni |
| **Casi d’uso associati / flussi alternativi** | / |
| **Postcondizioni** | Se sono state apportate modifiche alla prenotazione, i dati delle prenotazioni sono aggiornati. |

|  |  |
| --- | --- |
| **Nome** | UC13: Eventour |
| **Descrizione** | Visualizzazione di un eventour |
| **Attori** | User |
| **Obiettivi** | Visualizzazione di un tour di eventi |
| **Precondizioni** | L’utente deve essere loggato |
| **Passi** | 1. L’utente accede alla pagina di visualizzazione dell’eventour 2. Il sistema mostra un tour di eventi ottimizzato sulla base dell’utente e delle sue informazioni 3. L’utente può visualizzare più dettagli su un evento 4. Il sistema mostra altri dettagli relativi a esso e la possibilità di gestirne la prenotazione |
| **Casi d’uso associati / flussi alternativi** | / |
| **Postcondizioni** | L’utente, in caso di prenotazione di uno, alcuni o tutti gli eventi proposti dall’eventour, ritroverà questi eventi prenotati nella pagina dedicata |

|  |  |
| --- | --- |
| **Nome** | UC14: Pagamento |
| **Descrizione** | Viene effettuato il pagamento dell’evento da prenotare. |
| **Attori** | User |
| **Obiettivi** | Effettua il pagamento di un evento |
| **Precondizioni** | L’utente deve essere loggato  L’utente deve essere nella pagina della visualizzazione dei dettagli di un evento o nella pagina della visualizzazione di un eventour |
| **Passi** | 1. Il sistema, durante la prenotazione, controlla che l’evento non sia gratuito e mostra in tal caso la richiesta di inserimento dei dati di pagamento   2.a. L’attore vuole confermare il pagamento e la prenotazione         2.a.1. Il sistema carica una pagina di inserimento dati per il metodo di pagamento         2.a.2. L’attore inserisce i dati della carta         2.a.3. L’attore conferma i dati inseriti         2.a.4. Il sistema controlla i dati inseriti e richiede il pagamento         2.a.5.a. Se la transazione viene confermata, il sistema avvisa della conferma del pagamento e conferma la prenotazione         2.a.5.b. Se la transazione non viene confermata, il sistema avvisa dell’errore riscontrato durante il pagamento e non conferma la prenotazione  2.b. L’attore non vuole confermare il pagamento         2.b.1. L’attore annulla la richiesta di pagamento o riaggiorna la pagina         2.b.2. Il sistema non effettua la prenotazione |
| **Casi d’uso associati / flussi alternativi** | <<extend>> UC4: Prenotazione evento  Flusso alternativo 1: L’attore non conferma i dati inseriti con conseguente cancellazione delle informazioni.  Flusso alternativo 2: L’attore inserisce le informazioni del metodo di pagamento, ma non richiede il pagamento, uscendo prima della conferma di transazione. |
| **Postcondizioni** | L’attore ottiene un messaggio di notifica di avvenuto pagamento se tutto è andato a buon fine |

|  |  |
| --- | --- |
| **Nome** | UC15: Report Manager |
| **Descrizione** | Visualizzazione dei report relativi agli eventi e ai dati del manager. |
| **Attori** | Manager |
| **Obiettivi** | Visualizzazione report attività manager |
| **Precondizioni** | Il manager deve essere loggato |
| **Passi** | 1. Il manager entra nella pagina dei report 2. Il sistema mostra a video l’elenco di tutti i report degli eventi passati |
| **Casi d’uso associati / flussi alternativi** | / |
| **Postcondizioni** | / |

|  |  |
| --- | --- |
| **Nome** | UC16: Cancellazione Evento |
| **Descrizione** | Un evento viene cancellato dall’elenco degli eventi creati. |
| **Attori** | Manager |
| **Obiettivi** | Cancellare un evento |
| **Precondizioni** | * Il manager deve essere loggato * Il manager deve essere nella pagina di visualizzazione di un evento specifico |
| **Passi** | 1. Il manager seleziona un evento creato 2. Il manager clicca su “Cancella evento” 3. Il sistema cancella l’evento dal database. 4. Il sistema aggiorna la pagina di visualizzazione degli eventi |
| **Casi d’uso associati / flussi alternativi** | / |
| **Postcondizioni** | * L’evento è cancellato |

|  |  |
| --- | --- |
| **Nome** | UC17: Modifica Evento |
| **Descrizione** | Vengono modificate le informazioni relative ad un evento |
| **Attori** | Manager |
| **Obiettivi** | Modifica informazioni relative ad un evento |
| **Precondizioni** | * Il manager deve essere loggato * Il manager deve essere nella pagina di visualizzazione di un evento specifico |
| **Passi** | 1. Il manager clicca su “Modifica Evento” 2. Il sistema mostra le informazioni modificabili dell’evento 3. Il manager può modificare le informazioni compilando i campi modificabili 4. Il manager clicca sul pulsante di salvataggio delle informazioni 5. Il sistema salva le informazioni dell’evento 6. Il sistema mostra a video l’evento e i suoi dettagli aggiornati. |
| **Casi d’uso associati / flussi alternativi** | Flusso alternativo: L’uscita dalla pagina prima del salvataggio, comporta la mancata modifica delle informazioni. |
| **Postcondizioni** | Se sono state apportate modifiche, i dati dell’evento sono aggiornati. |

Alcuni casi d’uso sono stati modificati o perfezionati per l’implementazione successiva di funzionalità aggiuntive:

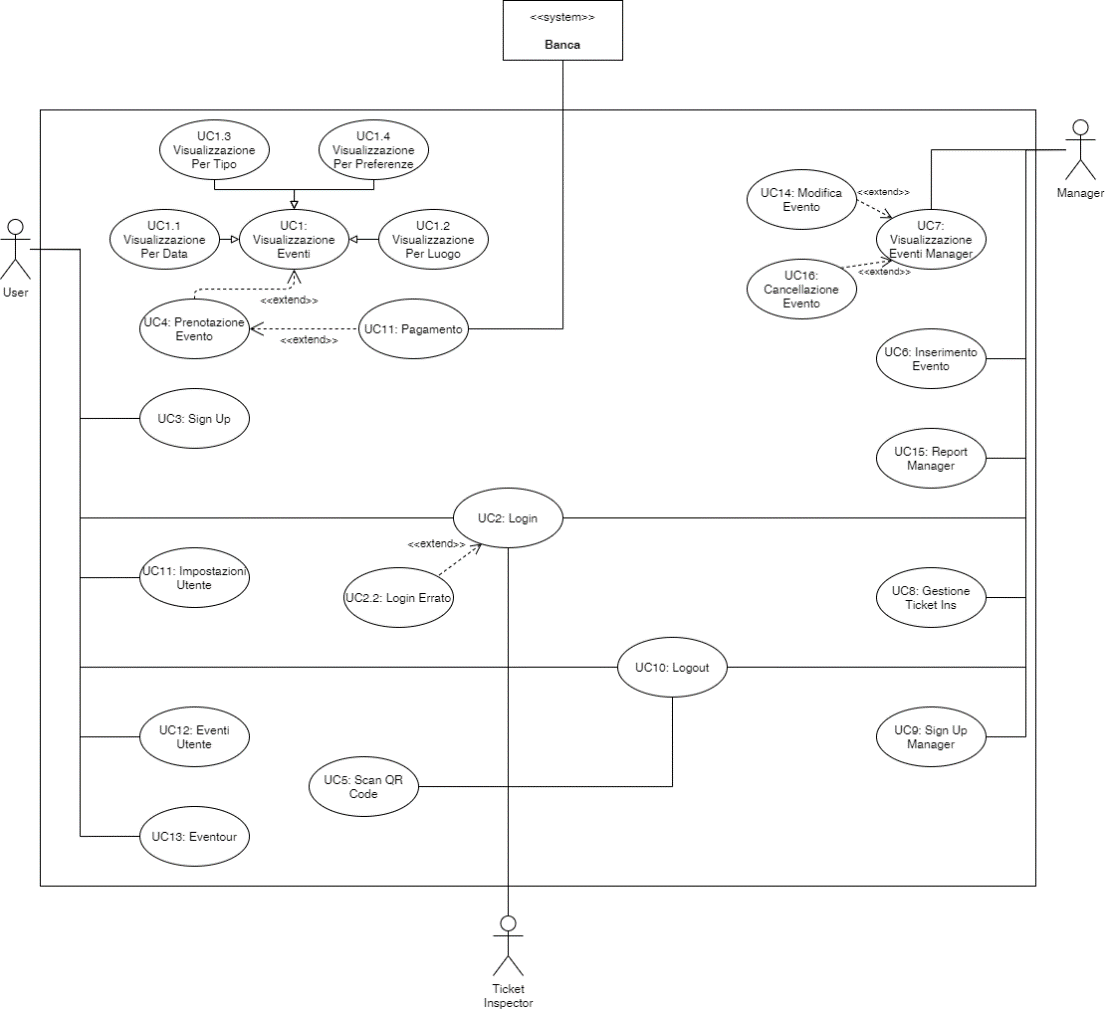
|  |  |
| --- | --- |
| **Nome** | UC1.1: Visualizzazione per data |
| **Descrizione** | Mostra tutti gli eventi disponibili ordinati per data o in intervallo di date |
| **Attori** | User |
| **Obiettivi** | Mostrare gli eventi disponibili in un certo ordine o data |
| **Precondizioni** | Se richiesto per data specifica, l’utente deve essere autenticato |
| **Passi** | 1. L’utente entra nella pagina di visualizzazione degli eventi 2. Il sistema stampa a video tutti gli eventi disponibili in ordine crescente di data (dal più vicino al più lontano)   3.a. Vista ordinata di un certo ordine  3.a.1. L’utente sceglie l’ordine con il quale mostrare gli eventi (dal più vicino al più lontano come date o viceversa)  3.a.2. Il sistema mostra a video gli eventi ordinati secondo l’ordine richiesto  3.b. Vista di eventi in una certa data o date  3.b.1. L’utente richiede di visualizzare gli eventi in una data o intervallo di date  3.b.2 Il sistema mostra a video gli eventi ordinati per data crescente inclusi nell’intervallo selezionato  4. L’utente può selezionare un evento di interesse  5. Il sistema mostra altri dettagli relativi a esso e la possibilità di gestirne la prenotazione |
| **Casi d’uso associati / flussi alternativi** | Specializzazione UC1: Visualizzazione Eventi |
| **Postcondizioni** | / |

|  |  |
| --- | --- |
| **Nome** | UC1.2: Visualizzazione per luogo |
| **Descrizione** | Mostra tutti gli eventi disponibili di un luogo indicato |
| **Attori** | User |
| **Obiettivi** | Mostrare gli eventi disponibili svolti in luoghi scelti |
| **Precondizioni** | L’utente deve essere autenticato |
| **Passi** | 1. L’utente entra nella pagina di visualizzazione degli eventi 2. Il sistema stampa a video tutti gli eventi disponibili in ordine crescente di data (dal più vicino al più lontano)   3.a. Vista di eventi in una regione  3.a.1. L’utente richiede la visualizzazione di eventi in una regione specifica  3.a.2. Il sistema mostra a video tutti gli eventi ordinati per data crescente svolti nella regione richiesta  3.b. Vista di eventi in una provincia  3.b.1. L’utente richiede la visualizzazione di eventi in una provincia specifica  3.b.2. Il sistema mostra a video tutti gli eventi ordinati per data crescente svolti nella provincia richiesta  3.c. Vista di eventi in un comune  3.c.1. L’utente richiede la visualizzazione di eventi in un comune specifico  3.c.2. Il sistema mostra a video tutti gli eventi ordinati per data crescente svolti nel comune richiesto  4. L’utente può selezionare un evento di interesse   1. Il sistema mostra altri dettagli relativi a esso e la possibilità di gestirne la prenotazione |
| **Casi d’uso associati / flussi alternativi** | Specializzazione UC1: Visualizzazione Eventi |
| **Postcondizioni** | / |

|  |  |
| --- | --- |
| **Nome** | UC4: Prenotazione Evento |
| **Descrizione** | L’utente può prenotare il posto per un evento o rimuovere una prenotazione da un evento già prenotato. |
| **Attori** | User |
| **Obiettivi** | Prenotare un posto per un evento da parte di un utente registrato |
| **Precondizioni** | * L’utente è loggato al sito * L’evento ha posti disponibili * L’evento avviene in una data futura * L’utente non è già prenotato all’evento in caso di prenotazione. * L’utente è già prenotato all’evento in caso di disdetta della prenotazione. * L’utente si trova nella pagina di visualizzazione degli eventi |
| **Passi** | 1.a. Prenotazione a un evento:  1.a.1. L’utente richiede la prenotazione all’evento nella pagina di visualizzazione  dell’evento  1.a.2. Il sistema controlla le condizioni richieste  1.a.3. Il sistema, dopo aver verificato la disponibilità del posto o posti richiesto/i, prenota il posto per l’evento e aggiorna il database  1.a.4. Il sistema conferma la prenotazione del posto richiesto  1.b. Disdetta di una prenotazione  1.b.1. L’utente richiede la disdetta della prenotazione all’evento nella pagina di  visualizzazione dell’evento  1.b.2. Il sistema disdice la prenotazione del posto o posti per l’evento e aggiorna il  database  1.b.3. Il sistema conferma la disdetta della prenotazione del posto richiesto |
| **Casi d’uso associati / flussi alternativi** | / |
| **Postcondizioni** | L’utente è prenotato all’evento e ha un posto riservato nel caso di prenotazione |

|  |  |
| --- | --- |
| **Nome** | UC8: Gestione Ticket Inspector |
| **Descrizione** | Il sistema permette al Manager la gestione dei propri Ticket Inspector. |
| **Attori** | Manager |
| **Obiettivi** | Organizzare i Ticket Inspector di un Manager |
| **Precondizioni** | Il Manager deve essere autenticato |
| **Passi** | 1. Il Manager richiede l’accesso alla pagina di visualizzazione di un evento dettagliato 2. Il sistema mostra la pagina dove poter visualizzare i Ticket Inspector di un evento.   3.a. Inserimento di un ticket inspector per l’evento  3.a.1. L’utente richiede l’inserimento di un nuovo ticket inspector  3.a.2. Il sistema richiede i dati del nuovo ticket inspector  3.a.3. L’utente compila i dati richiesti e chiede l’inserimento  3.a.4. Il sistema registra il nuovo ticket inspector salvando le informazioni nel  database e conferma all’utente il nuovo ticket inspector  3.b. Cancellazione di un ticket inspector per l’evento  3.b.1. L’utente richiede la cancellazione di un ticket inspector  3.b.2. Il sistema rimuove il ticket inspector dalla lista dei ticket inspector associati  all’evento |
| **Casi d’uso associati / flussi alternativi** | / |
| **Postcondizioni** | / |

### Use Case diagram

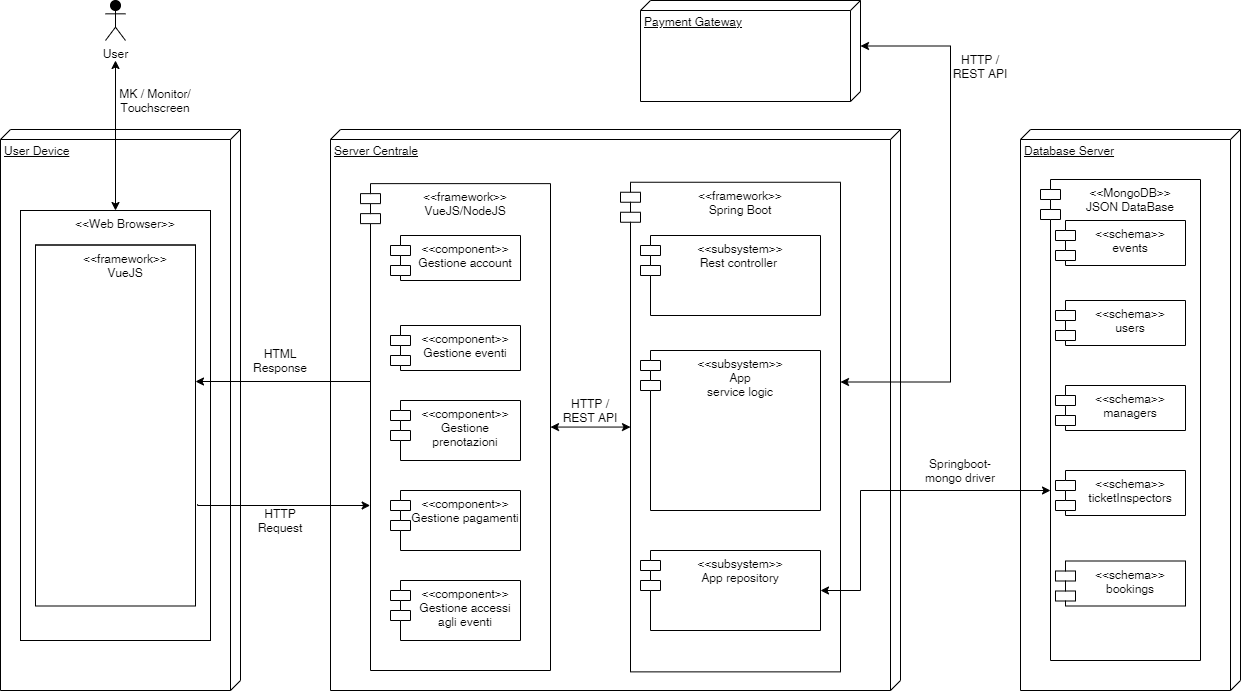


Modello 4.1 Use Case diagram

## Architettura

### Deployment diagram

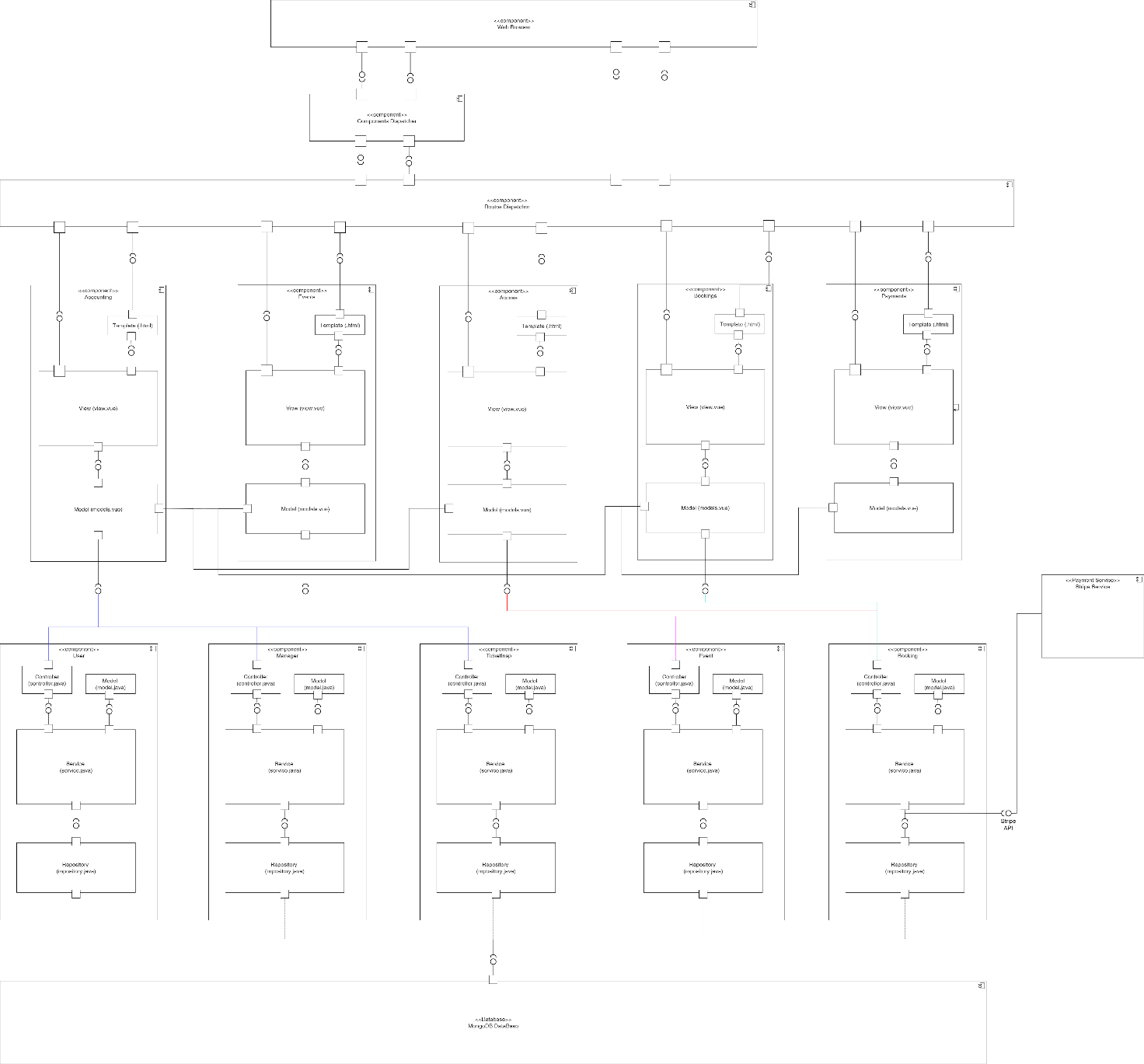
Per quanto concerne il deployment diagram, l’introduzione di nuove funzionalità ha richiesto l’aggiunta di un componente esterno alla piattaforma e alla API che permetta la comunicazione con i principali sistemi di pagamento. È stato quindi selezionato Stripe API che fornisce numerose API per la gestione e il pagamento online. Inoltre, si sono aggiunti nuovi componenti interni alla parte web visuale del sistema per la gestione di pagamenti.



Modello 4.2 Deployment Diagram

### Component diagram

La modifica del deployment diagram si ripercuote strettamente nel component diagram, in cui le iterazioni tra i vari model in Vue si incrementano e in cui viene aggiunto un nuovo macrocomponente per la gestione dei pagamenti. La parte di gestione dei pagamenti a livello di API viene elaborata nel componente di gestione delle prenotazioni, che comunica tramite Stripe API con uno Stripe Service.



Modello 4.3 Component Diagram

## Novità implementative introdotte

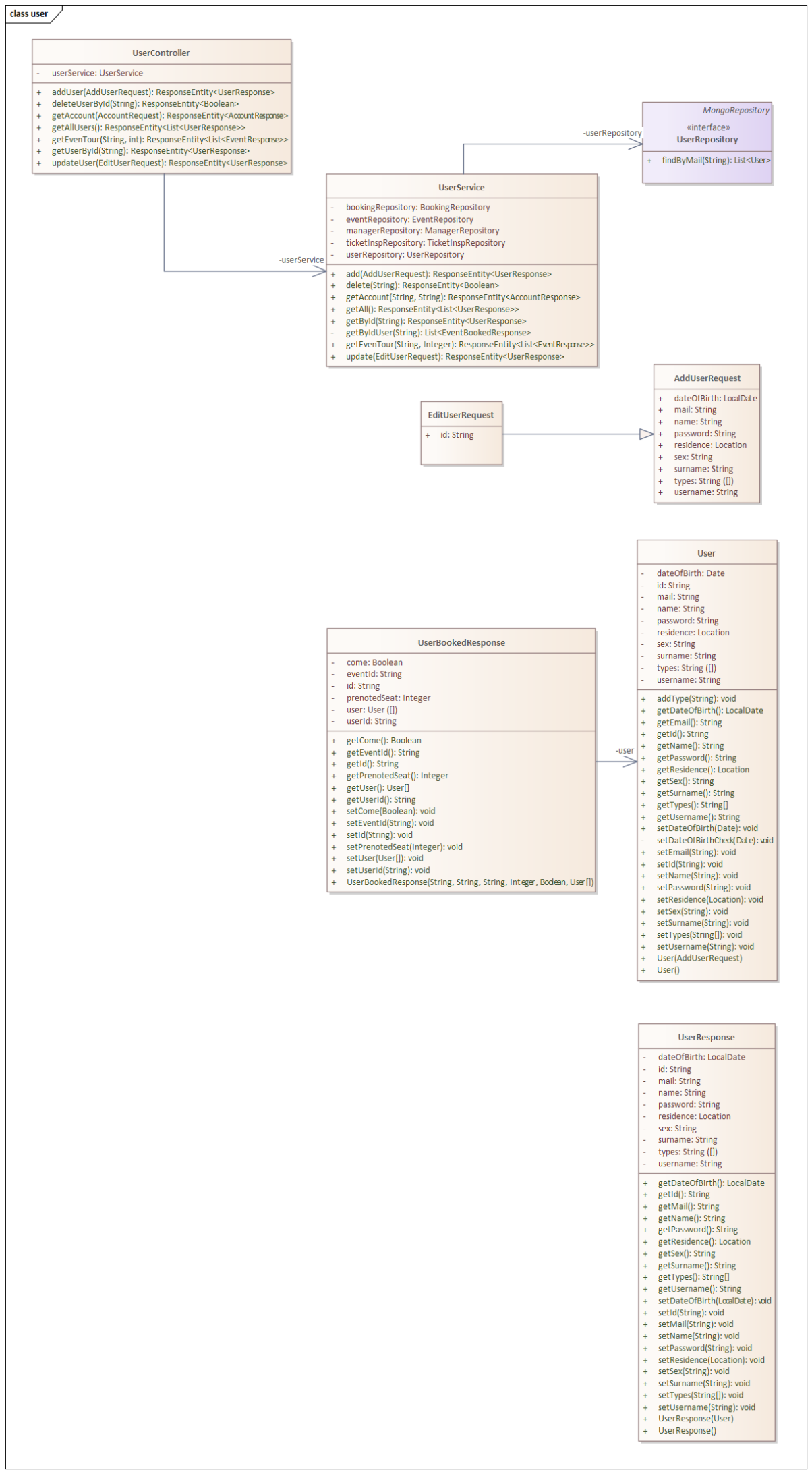
MD5

Mail per cancellazione evento

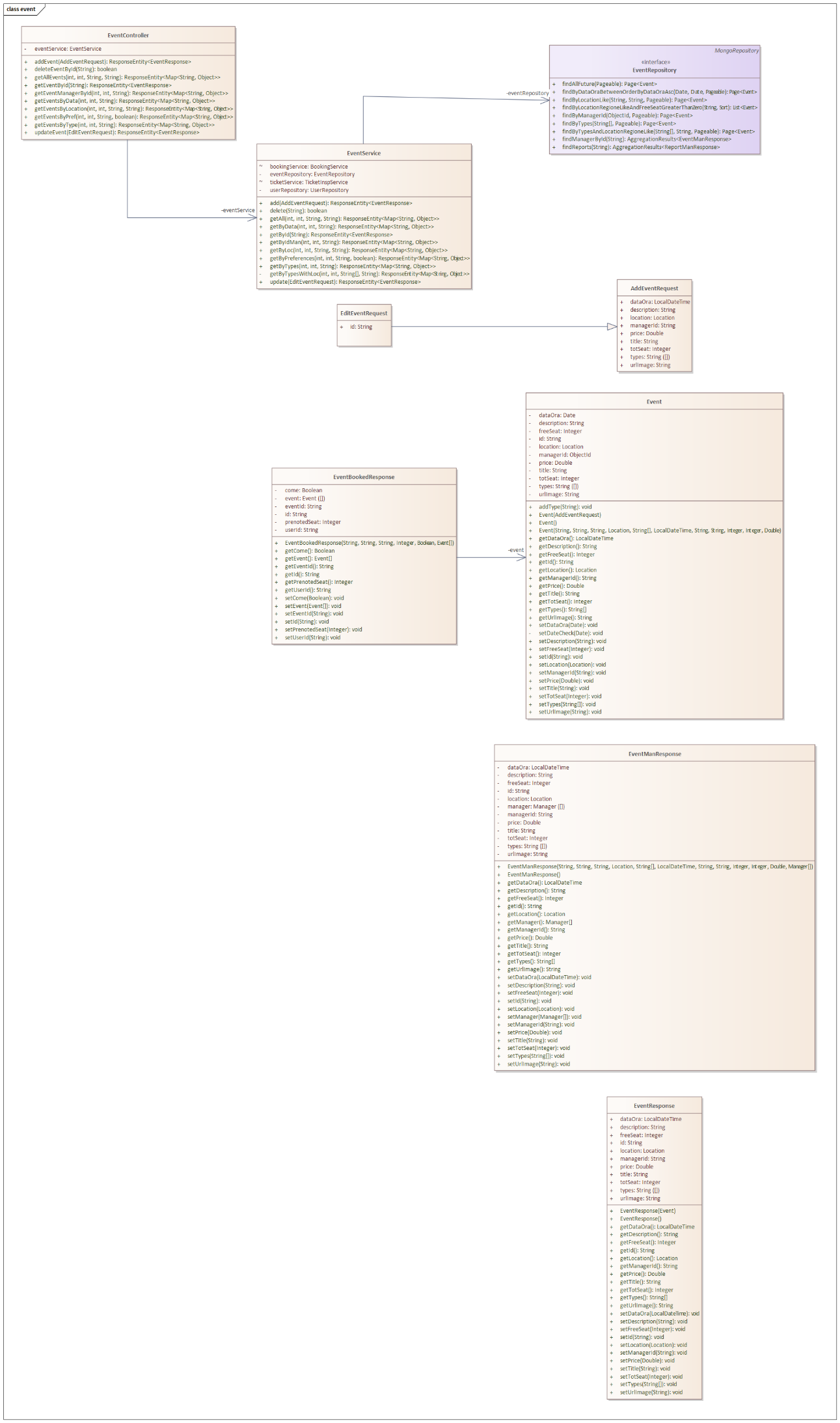
Pagamento (STRIPE api e modello su vue)

## Class diagram

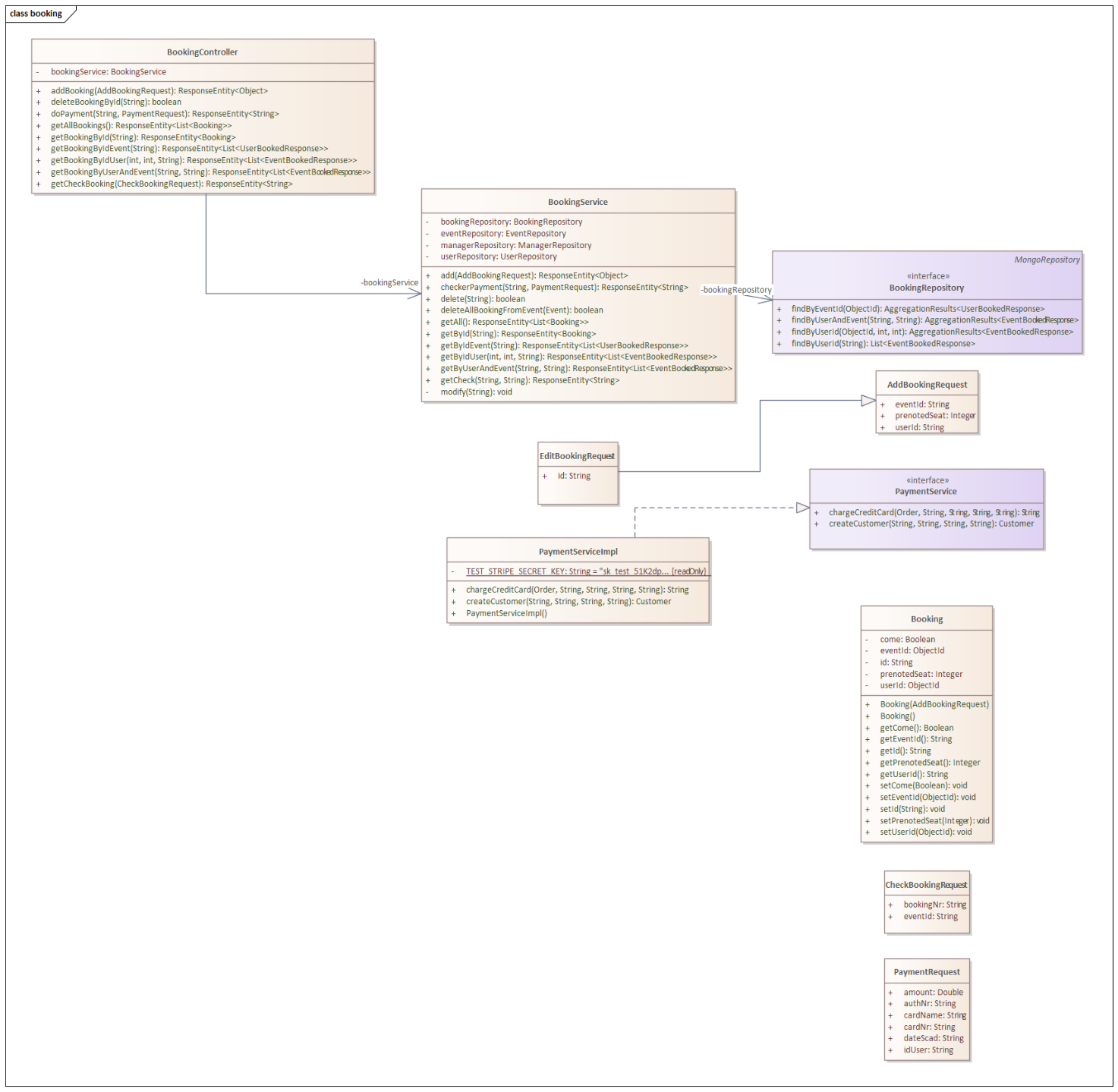
Di seguito sono mostrati i class diagram relativi alle classi utilizzate per l’implementazione dell’iterazione corrente.



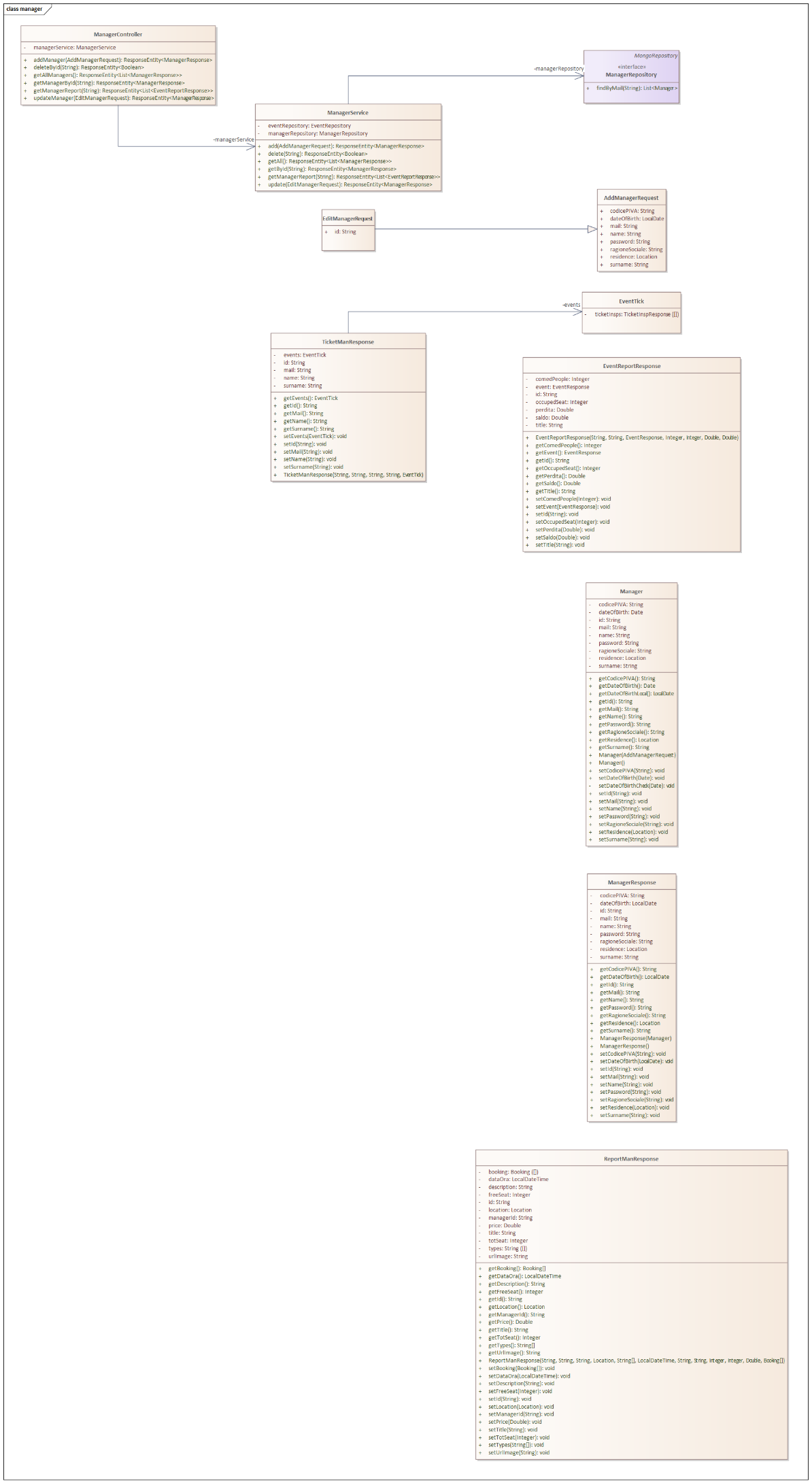
Modello 4.4 User Class Diagram



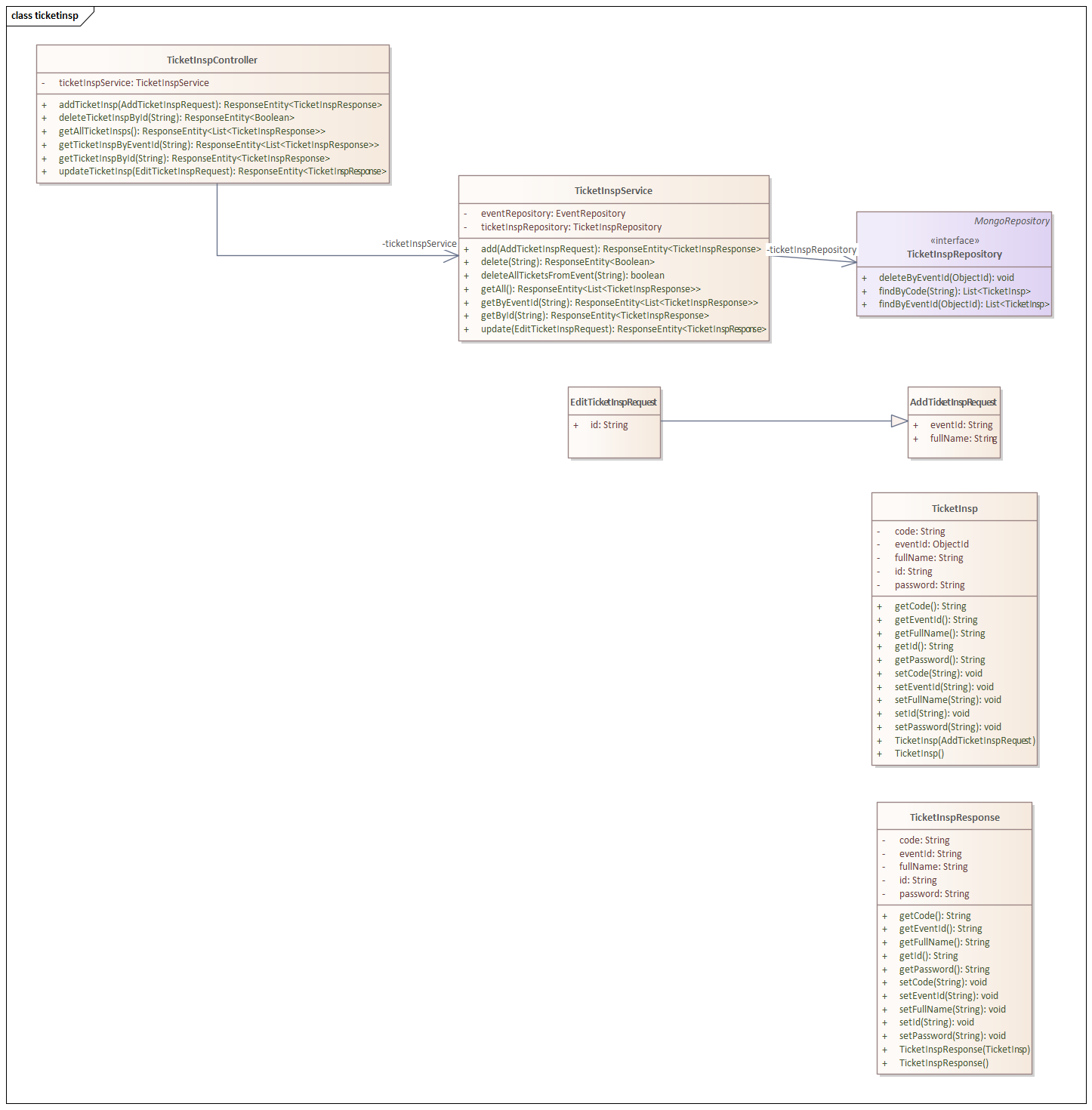
Modello 4.5 Event Class Diagram



Modello 4.6 Booking Class Diagram



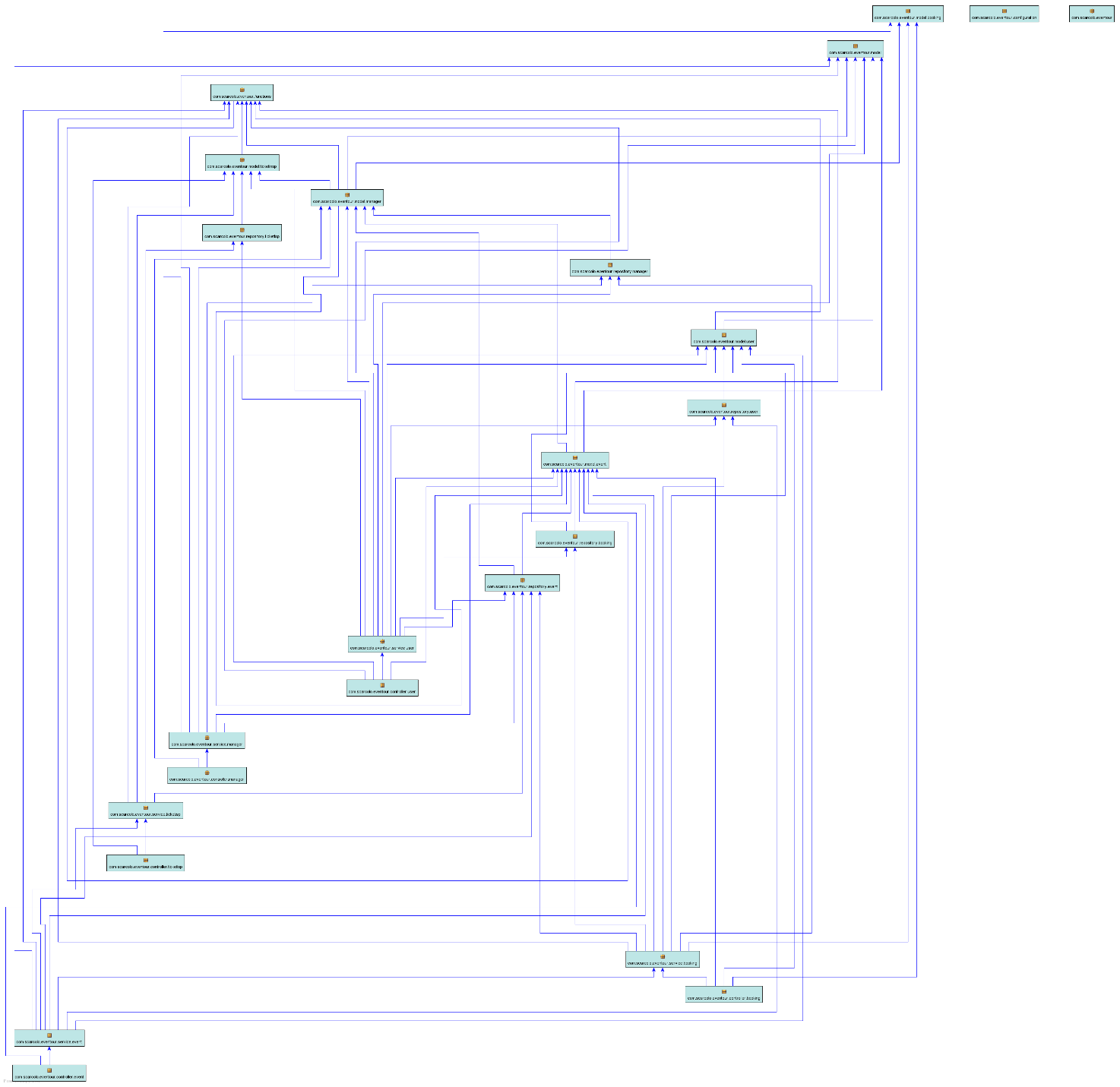
Modello 4.7 Manager Class Diagram



Modello 4.8 Ticket Inspector Class Diagram

A causa dell’aumento di complessità dei casi d’uso introdotti, anche le classi hanno avuto un incremento di metodi e funzioni introdotte.

Si riporta di seguito il package diagram, dove si nota l’aumento di complessità.



Modello 4.9 Package Diagram

## Algoritmo evenTour

Lo scopo della versione corrente dell’algoritmo progettato è quella di trovare ancora un numero di eventi definito dalla invocazione della API, che siano disponibili e che siano nella regione di residenza dell’utente, ma stavolta andando a includere anche gli eventi il cui tipo di evento sia simile a quello preferito dall’utente.

La versione ottimizzata è riportata nella pagina successiva, mentre quella non ottimizzata era realizzata sfruttando un doppio ciclo degli eventi:

Partendo dallo pseudocodice della iterazione precedente, si è deciso in prima istanza di aggiungere successivamente al primo ciclo di tutti gli eventi, un secondo ciclo identico ma in cui venivano aggiunti alla soluzione anche gli eventi con tipo simile (un tipo di evento è simile a un altro se la sua tipologia ha, nella tipizzazione padre, lo stesso valore. Ad esempio, i tipi 1.2.4 e 1.2.2 sono simili, così come sono simili i tipi 3.2 e 3.5, mentre i tipi 1.1.2 e 1.2.1 non lo sono).

Dopo una analisi si è ottimizzato l’algoritmo, per evitare il doppio ciclo, controllando quindi i tipi nel medesimo ciclo, aggiungendoli a due liste separate che vengono poi unite per una parte solo nel caso in cui gli eventi trovati come soluzione siano minori del numero richiesto. In questo modo è stato possibile anche ordinarli per data e ora prima di restituirli.

Nel caso in cui neanche l’unione degli eventi del medesimo tipo di quello preferito dall’utente con quelli con tipologia simile siano abbastanza, l’algoritmo restituisce un errore di soluzione non esistente.

Si riporta lo pseudocodice dell’algoritmo nella pagina successiva.



## Testing interfaccia grafica

### Analisi statica

### Analisi dinamica

* NightWatch



Test classe User relativi alla fase 1



Nuovo test relativo alla classe Manager

## Testing API

### Analisi statica

### Analisi dinamica

# Iterazione 3

L’iterazione 3 ha visto la comparsa di un nuovo ruolo all’interno della piattaforma. Ciò ha comportato l’inserimento di tutta la gestione delle richieste di iscrizione dei manager da parte degli amministratori, con una gestione semplificata dei manager. Si è inoltre introdotta la possibilità di rilasciare un riscontro agli eventi a cui sono stati da parte degli utenti, il rinnovo dell’iscrizione e il pagamento della quota d’iscrizione da parte del manager e la newsletter per gli utenti.

## Use Cases

### Descrizione testuale

Sono stati definiti nuovi casi d’uso:

|  |  |
| --- | --- |
| **Nome** | UC1521: Report |
| **Descrizione** | Visualizzazione di report |
| **Attori** | / |
| **Obiettivi** | Visualizzazione report |
| **Precondizioni** | L’utente che lo richiede deve essere loggato |
| **Passi** | 1. L’utente loggato entra nella pagina dei report 2. Il sistema mostra a video l’elenco di tutti i report |
| **Casi d’uso associati / flussi alternativi** | <<include>> UC2: Login  Specializzazione UC15: Report Manager, UC21: Report Admin |
| **Postcondizioni** | / |

|  |  |
| --- | --- |
| **Nome** | UC18: Feedback |
| **Descrizione** | L’utente può fornire una valutazione relativa all’evento a cui ha partecipato |
| **Attori** | User |
| **Obiettivi** | Fornire dei feedback che possano essere usati per la valutazione del Manager |
| **Precondizioni** | l’utente deve essere loggato |
| **Passi** | 1. L’utente richiede di accedere alla pagina dei feedback 2. Il sistema riporta l’utente nella pagina dei feedback 3. Il sistema mostra tutti gli eventi passati per cui l’utente si è prenotato e presentato e nei quali è possibile fornire un feedback 4. L’utente sceglie uno degli eventi per cui vuole fornire il feedback e modifica o inserisce una valutazione da 1 a 5 del gradimento dell’evento 5. Il sistema salva il feedback dato dall’utente 6. Il sistema aggiorna i dati relativi al feedback dato nel database e aggiorna la pagina dei feedback |
| **Casi d’uso associati / flussi alternativi** | Flusso alternativo: L’uscita dalla pagina senza aver modificato o inserito feedback comporta la mancata modifica di recensioni. |
| **Postcondizioni** | Avere un riscontro da parte dell’utente relativo all’evento svolto |

|  |  |
| --- | --- |
| **Nome** | UC19: Accept Request |
| **Descrizione** | Il sistema permette all’Admin di autorizzare i nuovi manager |
| **Attori** | Admin |
| **Obiettivi** | Autorizzare la registrazione di un manager da parte dell’Admin |
| **Precondizioni** | L’Admin deve essere autenticato |
| **Passi** | 1. L’admin richiede di accedere alla pagina di gestione delle richieste 2. Il sistema mostra la lista dei Manager che richiedono di poter svolgere il loro ruolo nella piattaforma   3.a. Accettazione di una richiesta  3.a.1. L’Admin autorizza un manager che rispetta i requisiti  3.a.2. Il sistema inserisce il manager come autorizzati a operare nella piattaforma  3.a.3. Il sistema notifica ai manager via mail che la loro registrazione è stata autorizzata (NON REALIZZATA)  3.b. Rifiuto di una richiesta  3.b.1. L’Admin elimina la richiesta di un manager che non rispetta i requisiti  3.b.2. Il sistema notifica via mail che la richiesta del manager di registrarsi non è stata autorizzata (non realizzata)  3.b.3. Il sistema elimina la registrazione del manager dalla piattaforma |
| **Casi d’uso associati / flussi alternativi** | <<extend>> UC20: Gestione manager |
| **Postcondizioni** | I manager autorizzati sono operativi sulla piattaforma, se autorizzati. Se sono state rifiutate delle richieste, esse e i relativi manager saranno cancellati |

|  |  |
| --- | --- |
| **Nome** | UC20: Gestione manager |
| **Descrizione** | L’Admin può gestire i manager attribuendo un malus o visualizzando i dati di essi |
| **Attori** | Admin |
| **Obiettivi** | Gestione dei Manager da parte dell’Admin |
| **Precondizioni** | L’Admin deve essere autenticato |
| **Passi** | 1. L’Admin richiede di entrare nella gestione dei manager 2. Il sistema mostra la pagina di gestione dei manager   3.a. L’admin visualizza i manager e i dati collegati  3.b.1. L’admin attribuisce un malus a un manager  3.b.2. Il sistema attribuisce il malus  3.b.3. il sistema riaggiorna la pagina di gestione dei manager |
| **Casi d’uso associati / flussi alternativi** |  |
| **Postcondizioni** | Nel caso di malus attribuito a un manager, il manager sarà costretto al primo accesso a dover ripagare nuovamente il rinnovo dell’iscrizione come penalità. |

|  |  |
| --- | --- |
| **Nome** | UC21: Report Admin |
| **Descrizione** | Stampa di report relativi all’attività dei manager. |
| **Attori** | Admin |
| **Obiettivi** | Stampa report dei manager presenti |
| **Precondizioni** | L’admin deve essere autenticato |
| **Passi** | 1. L’admin entra nella pagina dei report 2. Il sistema mostra a video l’elenco di tutti i report relativi ai manager presenti sulla piattaforma |
| **Casi d’uso associati / flussi alternativi** | Generalizzazione UC1521: Report |
| **Postcondizioni** | / |
|  | |
| **Nome** | UC22: Impostazioni manager |
| **Descrizione** | Il Manager visualizza e modifica eventualmente le proprie impostazioni |
| **Attori** | Manager |
| **Obiettivi** | Visualizzazione e modifica delle impostazioni di un Manager |
| **Precondizioni** | Il Manager deve essere loggato |
| **Passi** | 1. Il Manager richiede l’accesso alla pagina delle proprie impostazioni 2. Il sistema visualizza le impostazioni del Manager   3.a. Il sistema permette l’uscita dalle impostazioni  3.b. modifica di impostazioni  3.b.2. Il sistema mostra i campi disponibili per la modifica, precompilati col valore precedente  3.b.3. Il Manager modifica i dati di interesse  3.b.4. Il sistema controlla le modifiche fatte  3.b.5.a. Se le modifiche sono corrette, il sistema notifica l’avvenuta modifica  3.b.5.b. Se le modifiche son incorrette, il sistema mostra l’errore e richiede la correzione della modifica  3.b.6. Il sistema riporta il manager nella pagina di visualizzazione delle impostazioni |
| **Casi d’uso associati / flussi alternativi** |  |
| **Postcondizioni** | Le informazioni sono state modificate (se richiesto dal Manager) |

|  |  |
| --- | --- |
| **Nome** | UC23: Rinnovo iscrizione |
| **Descrizione** | Il Manager effettua se necessario il rinnovo dell’iscrizione |
| **Attori** | Manager |
| **Obiettivi** | Rinnovo iscrizione alla piattaforma |
| **Precondizioni** |  |
| **Passi** | 1. Il sistema, all’autenticazione alla piattaforma, verifica che il manager sia attivo e debba effettuare il rinnovo dell’iscrizione.   2.a. Rinnovo dell’iscrizione  2.a.1. In caso di necessario rinnovo, il sistema mostra al manager i dati da compilare per effettuare il pagamento del rinnovo dell’iscrizione.  2.a.2. Il sistema riporta alla pagina di pagamento (UC14) e, in caso di esito positivo di esso, permette l’accesso al manager.  2.b. Il Manager esce dal rinnovo dell’iscrizione nel UC14 e non viene autenticato.  2.c. In caso di non necessità di rinnovo, il manager viene regolarmente autenticato |
| **Casi d’uso associati / flussi alternativi** | <<include>> UC14: Pagamento  <<extend>> UC2: Login |
| **Postcondizioni** | L’scrizione alla piattaforma è attiva regolarmente nel caso di pagamento completato |

|  |  |
| --- | --- |
| **Nome** | UC24: Newsletter |
| **Descrizione** | Il sistema invia una newsletter in base alle impostazioni scelte dagli utenti |
| **Attori** | Admin |
| **Obiettivi** | Inviare una newsletter sugli eventi futuri |
| **Precondizioni** | L’Admin deve essere autenticato |
| **Passi** | 1. Il sistema elabora una mail con gli eventi preferiti dall’utente che si è registrato alla newsletter e/o gli eventi sponsorizzati   2.a. Se l’utente è registrato settimanalmente alla newsletter, allora il sistema invia la mail con la newsletter  2.b.a. Se l’utente è registrato bisettimanalmente alla newsletter, se ricade nella settimana in cui non deve ricevere nulla, allora non viene inviata la mail.  2.b.b. Se l’utente è registrato bisettimanalmente alla newsletter, se ricade nella settimana in cui deve ricevere la newsletter, allora viene inviata la mail. |
| **Casi d’uso associati / flussi alternativi** | / |
| **Postcondizioni** | Invio delle newsletter agli utenti registrati completato |

Alcuni casi d’uso sono stati modificati come di seguito:

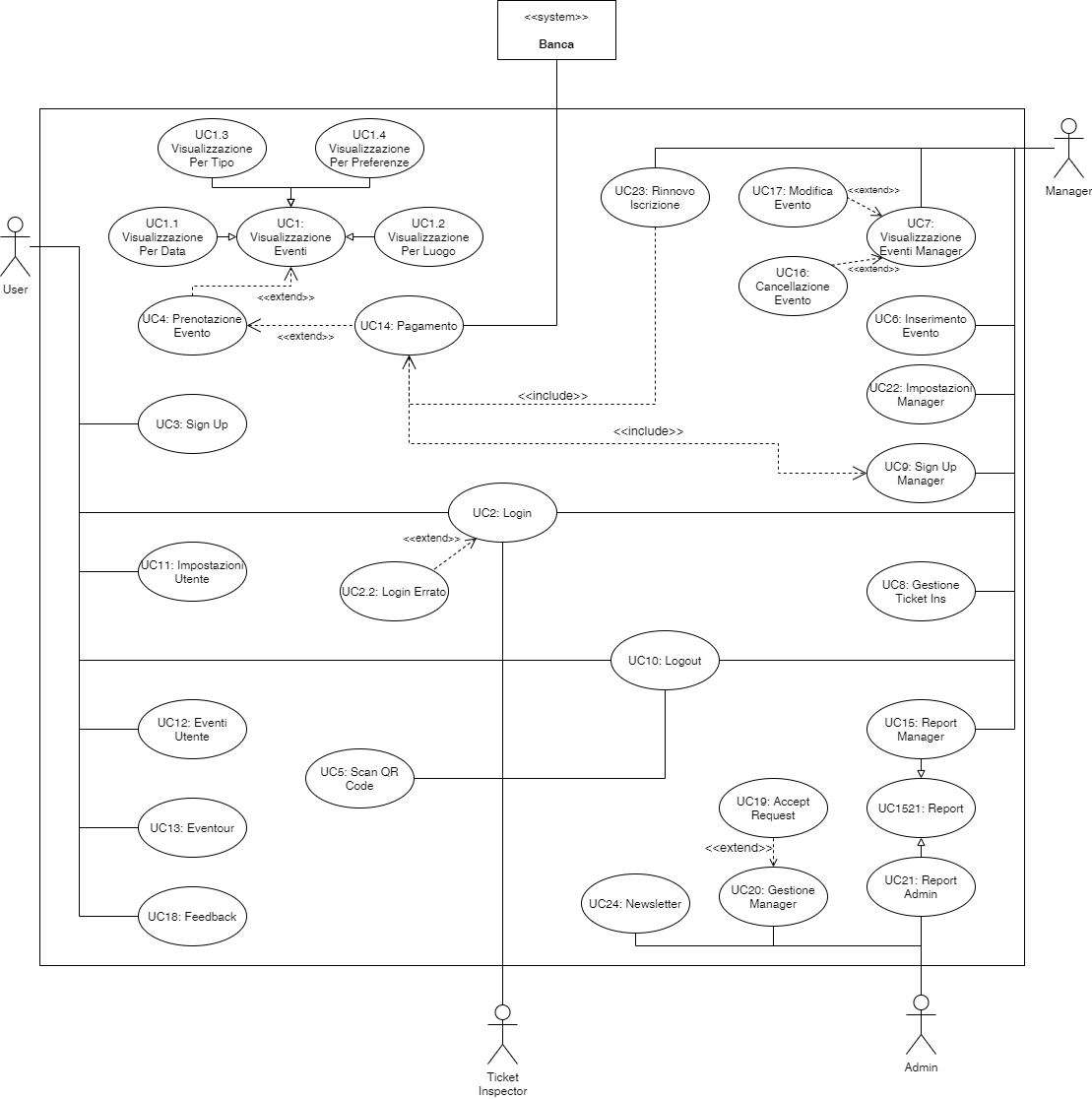
|  |  |
| --- | --- |
| **Nome** | UC2: Login |
| **Descrizione** | L’utente può accedere al suo account per effettuare prenotazioni, se utente, o altre funzioni, se manager o ticket inspector |
| **Attori** | User, Manager, Ticket Inspector, Admin |
| **Obiettivi** | Accedere alla propria area privata nel sito per svolgere funzioni |
| **Precondizioni** | Essere registrato |
| **Passi** | 1. L’attore entra nella pagina di login 2. L’attore compila il campo “username” 3. L’attore compila il campo “password” 4. L’attore richiede di essere autenticato con quelle credenziali 5. Il sistema autorizza l’accesso all’utente e lo porta nella sua area riservata   5.a. Nel caso di autenticazione di un User, il sistema riporta alla pagina di visualizzazione evento  5.b. Nel caso di autenticazione di un Manager, il sistema riporta alla pagina di visualizzazione eventi del manager  5.c. Nel caso di autenticazione di un Ticket Inspector, il sistema riporta alla pagina di scansione del QR  5.d. Nel caso di autenticazione di un Admin, il sistema riporta alla pagina di gestione manager |
| **Casi d’uso associati / flussi alternativi** | / |
| **Postcondizioni** | L’utente è autenticato |

|  |  |
| --- | --- |
| **Nome** | UC9: Sign Up Manager |
| **Descrizione** | Il sistema permette all’utente di registrarsi come Manager (organizzatore di eventi) |
| **Attori** | Manager |
| **Obiettivi** | Registrare un Manager come tale |
| **Precondizioni** | Il Manager non è ancora registrato |
| **Passi** | 1. L’utente richiede la registrazione alla piattaforma come Manager 2. Il sistema mostra la pagina di registrazione alla piattaforma come Manager 3. L’utente compila tutti i dati anagrafici e dell’organizzazione 4. L’utente richiede la registrazione con i dati inseriti 5. Il sistema controlla i dati inseriti   6.a.1. In caso di validazione corretta, il sistema riporta alla pagina di pagamento della quota di iscrizione (UC14).  6.a.2.a. Se il pagamento va a buon fine, il sistema inserirà il manager nella lista dei manager richiedenti l’attivazione da parte di un amministratore.  6.a.2.b. Se il pagamento non va a buon fine, viene rimostrata la pagina di pagamento.  6.b. In caso di validazione errata, il sistema richiede la modifica dei dati errati. |
| **Casi d’uso associati / flussi alternativi** | <<include>> UC14: Pagamento |
| **Postcondizioni** | Il Manager è registrato ma non attivato se la registrazione è andata a buon fine |

|  |  |
| --- | --- |
| **Nome** | UC14: Pagamento |
| **Descrizione** | Viene effettuato il pagamento dell’evento da prenotare o della quota di iscrizione. |
| **Attori** | User, Manager |
| **Obiettivi** | Effettua il pagamento di un evento o della quota di iscrizione |
| **Precondizioni** | L’utente deve essere loggato  L’utente deve essere nella pagina della visualizzazione dei dettagli di un evento o nella pagina della visualizzazione di un eventour o nella pagina di registrazione di un manager o di rinnovo iscrizione |
| **Passi** | Pagamento di un evento o di un eventour  1. Il sistema, durante la prenotazione, controlla che l’evento non sia gratuito e mostra in tal caso la richiesta di inserimento dei dati di pagamento  2.a. L’utente vuole confermare il pagamento e la prenotazione  2.a.1. Il sistema carica una pagina di inserimento dati per il metodo di pagamento  2.a.2. L’utente inserisce i dati della carta  2.a.3. L’utente conferma i dati inseriti  2.a.4. Il sistema controlla i dati inseriti e richiede il pagamento  2.a.5.a. Se la transazione viene confermata, il sistema avvisa della conferma del pagamento e conferma la prenotazione  2.a.5.b. Se la transazione non viene confermata, il sistema avvisa dell’errore riscontrato durante il pagamento e non conferma la prenotazione  2.b. L’utente non vuole confermare il pagamento  2.b.1. L’utente annulla la richiesta di pagamento o riaggiorna la pagina  2.b.2. Il sistema non effettua la prenotazione  Pagamento della quota di iscrizione  1. Il sistema, durante la registrazione o durante il rinnovo dell’iscrizione, in cui verifica se l’utente deve rinnovare l’iscrizione, mostra la richiesta di inserimento dei dati di pagamento  2.a. Il manager vuole confermare il pagamento della quota  2.a.1. Il sistema carica una pagina di inserimento dati per il metodo di pagamento  2.a.2. Il manager inserisce i dati della carta  2.a.3. Il manager conferma i dati inseriti  2.a.4. Il sistema controlla i dati inseriti e richiede il pagamento  2.a.5.a. Se la transazione viene confermata, il sistema avvisa della conferma del pagamento e registra l’utente come da attivare, nel caso della registrazione, o con data di rinnovo odierna, nel caso di rinnovo.  2.a.5.b. Se la transazione non viene confermata, il sistema avvisa dell’errore riscontrato durante il pagamento e attende nuovi dati.  2.b. Il manager non vuole confermare il pagamento  2.b.1. Il manager annulla la richiesta di pagamento o riaggiorna la pagina  2.b.2. Il sistema non effettua il pagamento |
| **Casi d’uso associati / flussi alternativi** | <<extend>> UC4: Prenotazione evento  Flusso alternativo 1: L’attore non conferma i dati inseriti con conseguente cancellazione delle informazioni.  Flusso alternativo 2: L’attore inserisce le informazioni del metodo di pagamento, ma non richiede il pagamento, uscendo prima della conferma di transazione. |
| **Postcondizioni** | L’attore ottiene un messaggio di notifica di avvenuto pagamento se tutto è andato a buon fine |

|  |  |
| --- | --- |
| **Nome** | UC16: Cancellazione Evento |
| **Descrizione** | Un evento viene cancellato dall’elenco degli eventi creati. |
| **Attori** | Manager |
| **Obiettivi** | Cancellare un evento |
| **Precondizioni** | * Il manager deve essere loggato * Il manager deve essere nella pagina di visualizzazione di un evento specifico |
| **Passi** | 1. Il manager seleziona un evento creato 2. Il manager clicca su “Cancella evento” 3. Il sistema cancella l’evento dal database, elimina tutti i ticket inspectors associati a quell’evento e cancella tutte le prenotazioni relative a esso, avvisando via mail gli utenti che avevano prenotato 4. Il sistema aggiorna la pagina di visualizzazione degli eventi |
| **Casi d’uso associati / flussi alternativi** | / |
| **Postcondizioni** | * L’evento è cancellato |

### Use Case diagram

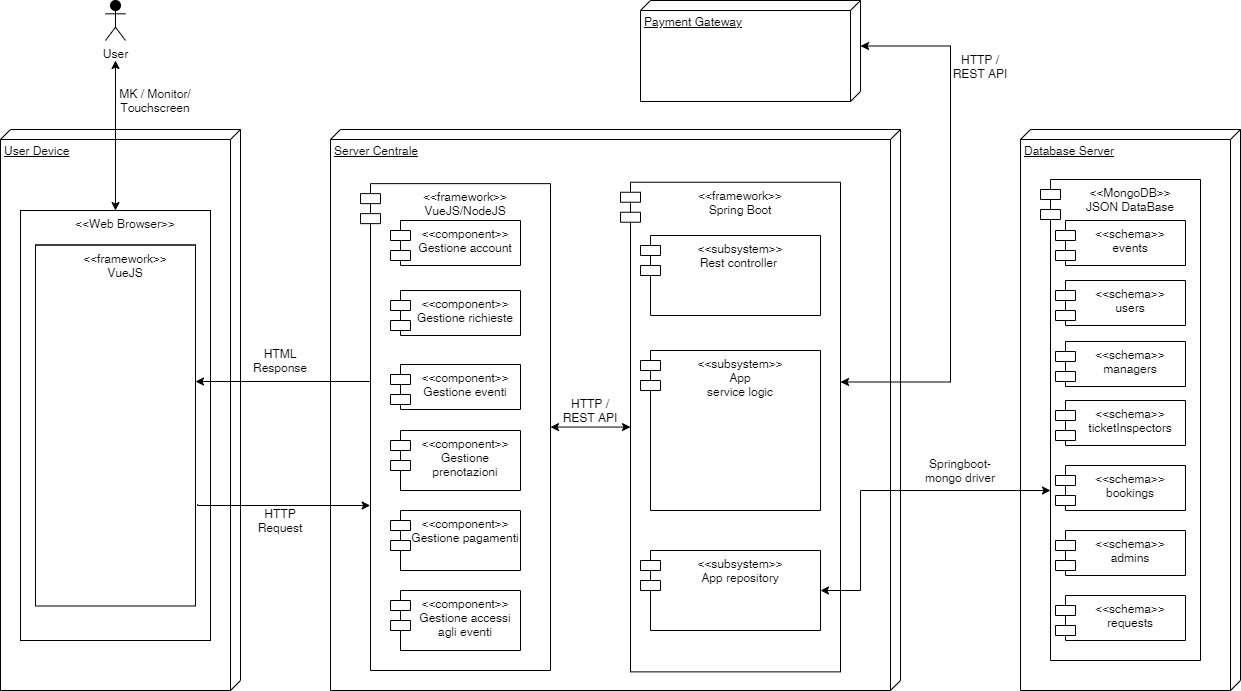


Modello 5.1 Use Case Diagram

## Architettura

### Deployment diagram

Per quanto concerne il deployment diagram, l’introduzione di nuove funzionalità ha richiesto l’aggiunta di nuovi componenti interni alla parte web visuale del sistema e l’introduzione di nuove collezioni di dati.

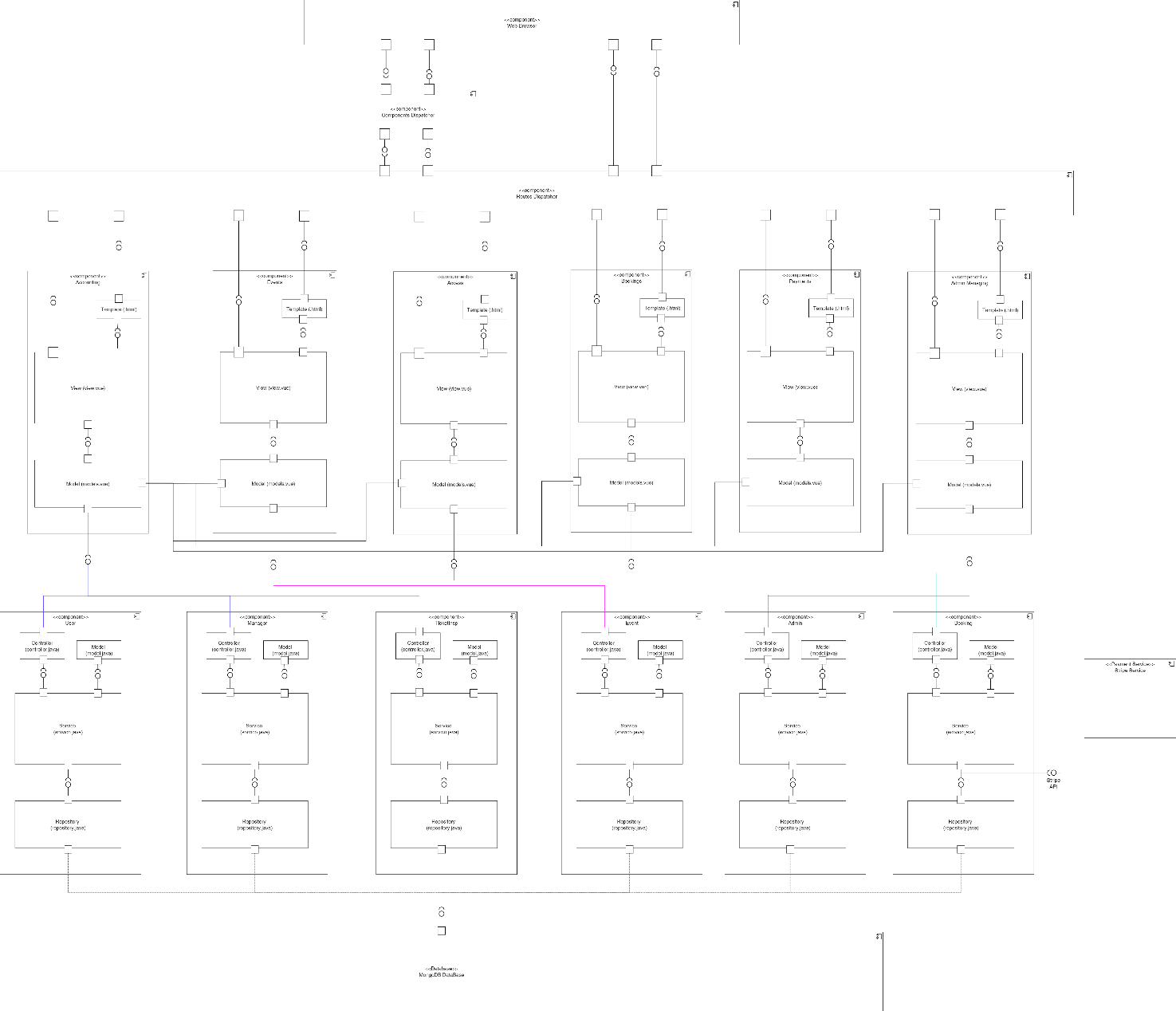


Modello 5.2 Deployment Diagram

### Component diagram

La modifica del deployment diagram si ripercuote strettamente nel component diagram, in cui le iterazioni tra i vari model in Vue si incrementano e in cui viene aggiunto un nuovo macrocomponente per la gestione degli amministratori. La parte di gestione degli amministratori a livello di API viene elaborata in un suo componente che comunica con altri componenti.

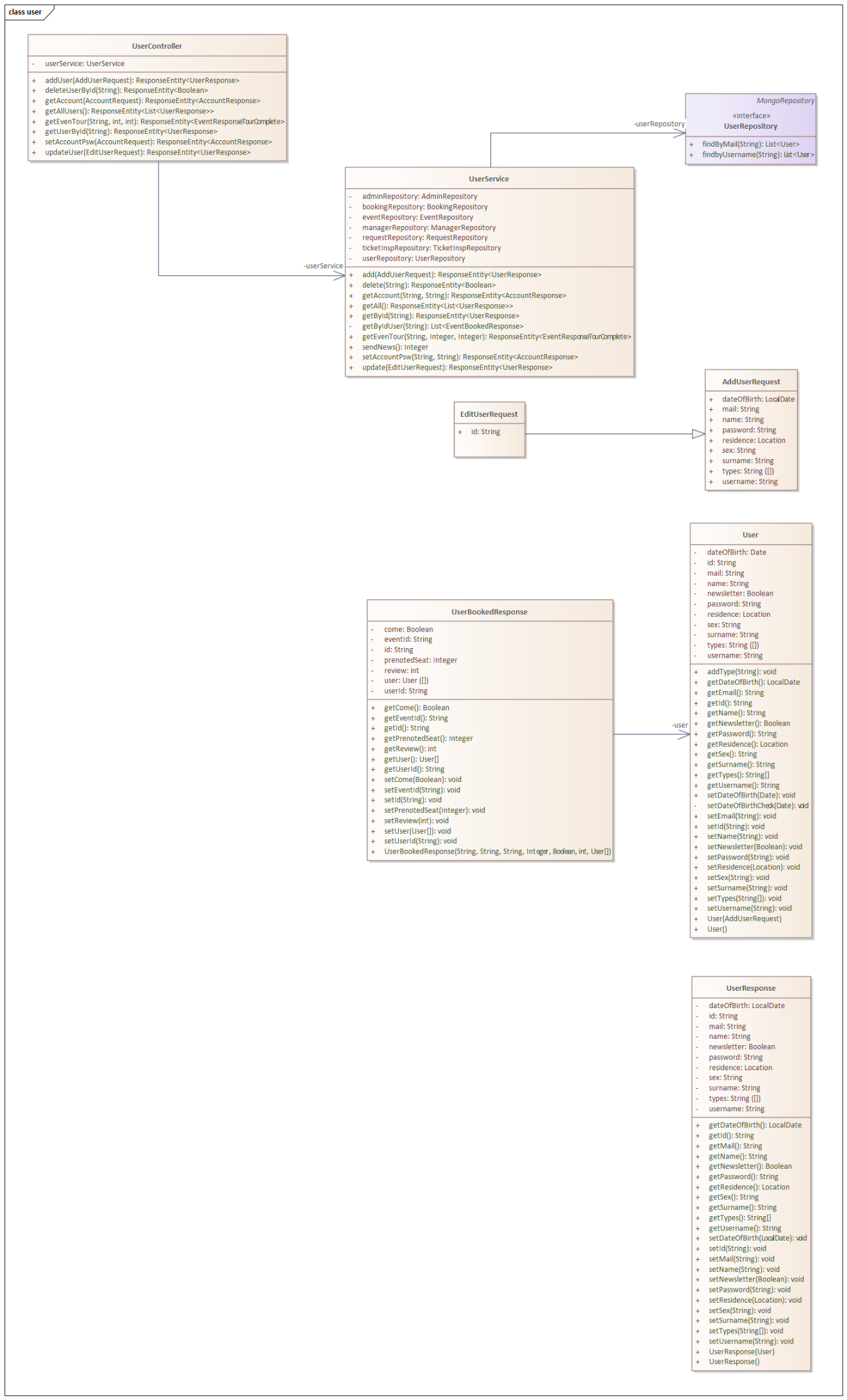
Si riportano le aggiunte di seguito.



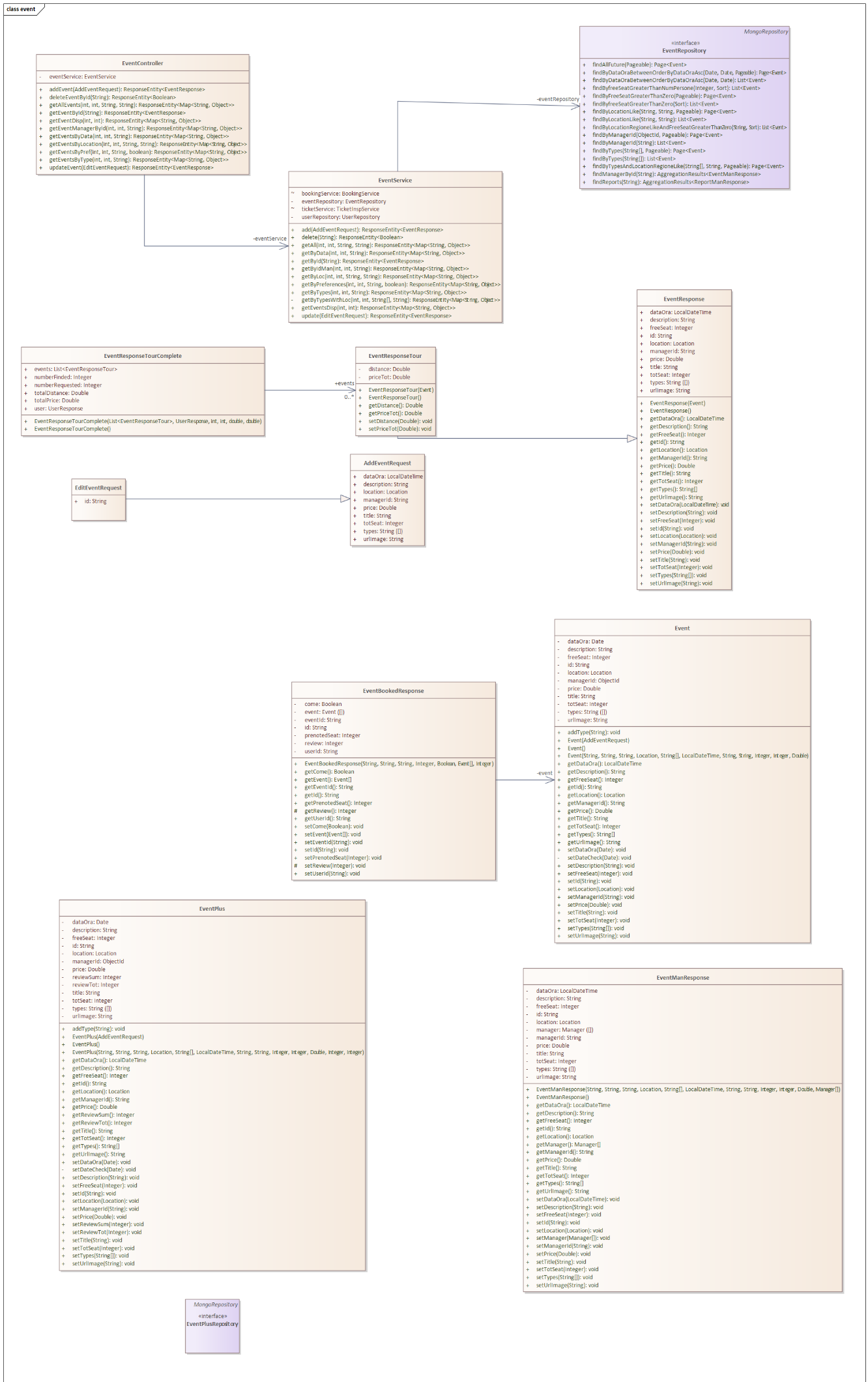
Modello 5.3 Component Diagram

## Class diagram

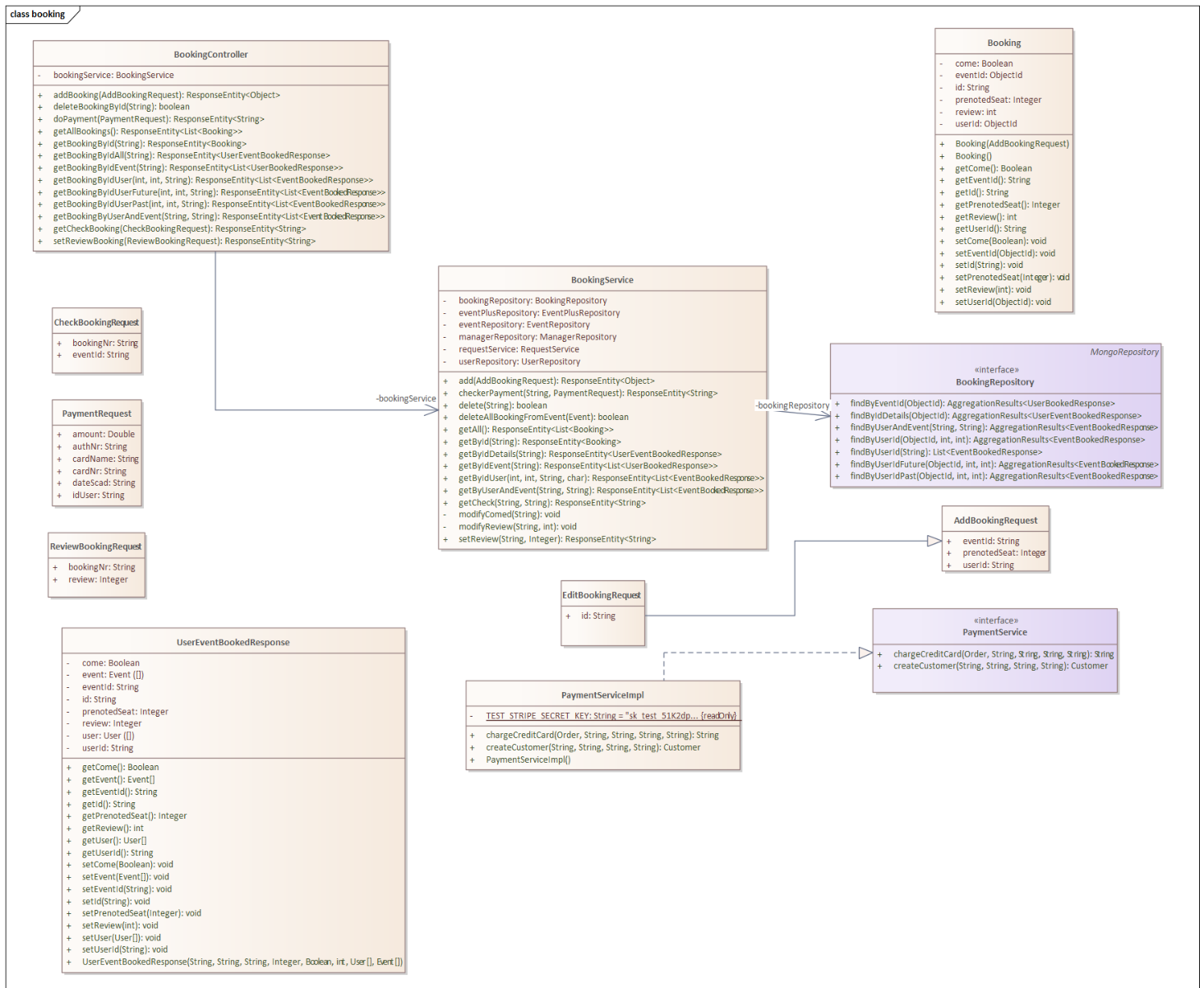
Di seguito sono mostrati i class diagram relativi alle classi utilizzate per l’implementazione dell’iterazione corrente.



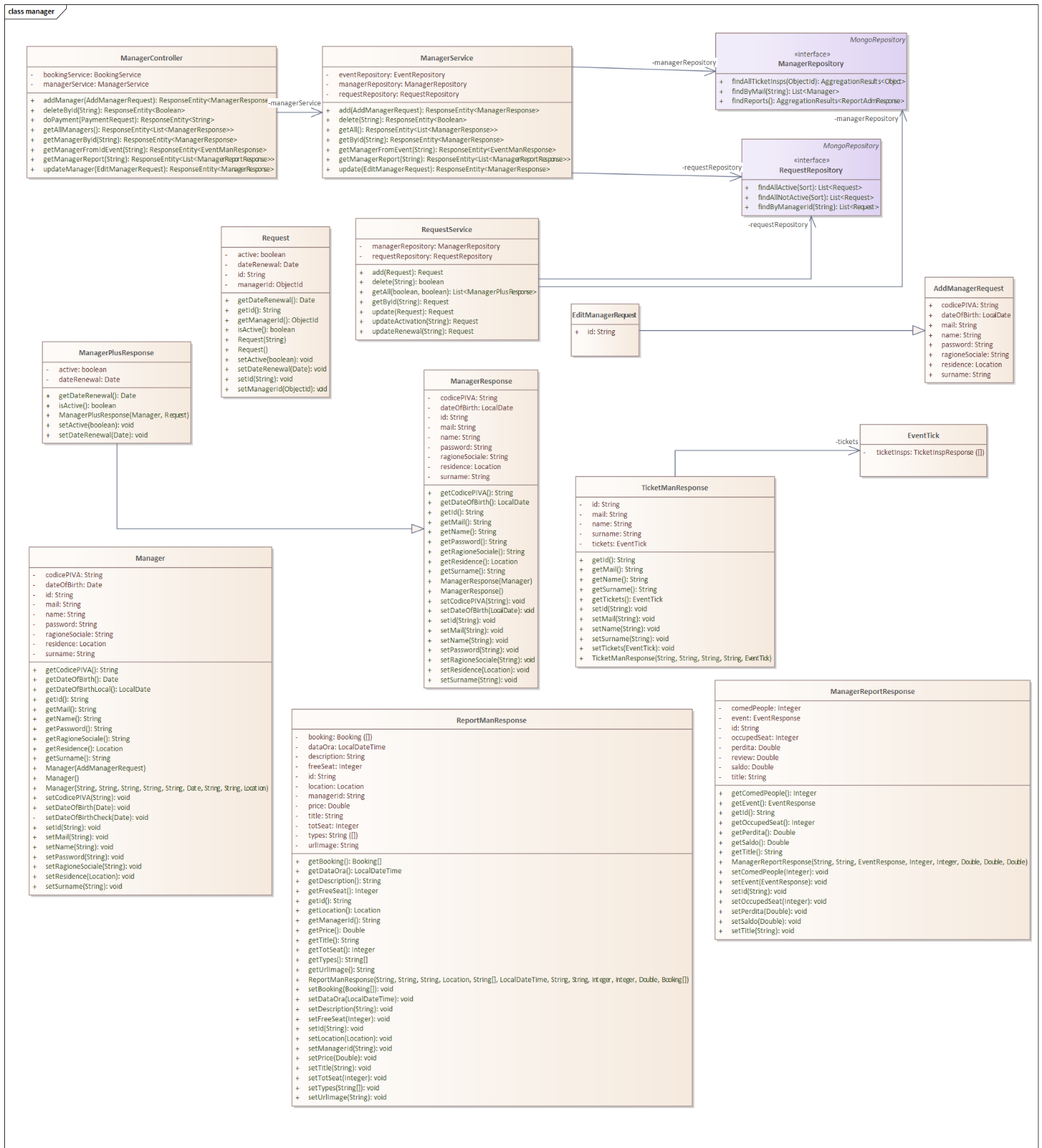
Modello 5.4 User Class Diagram



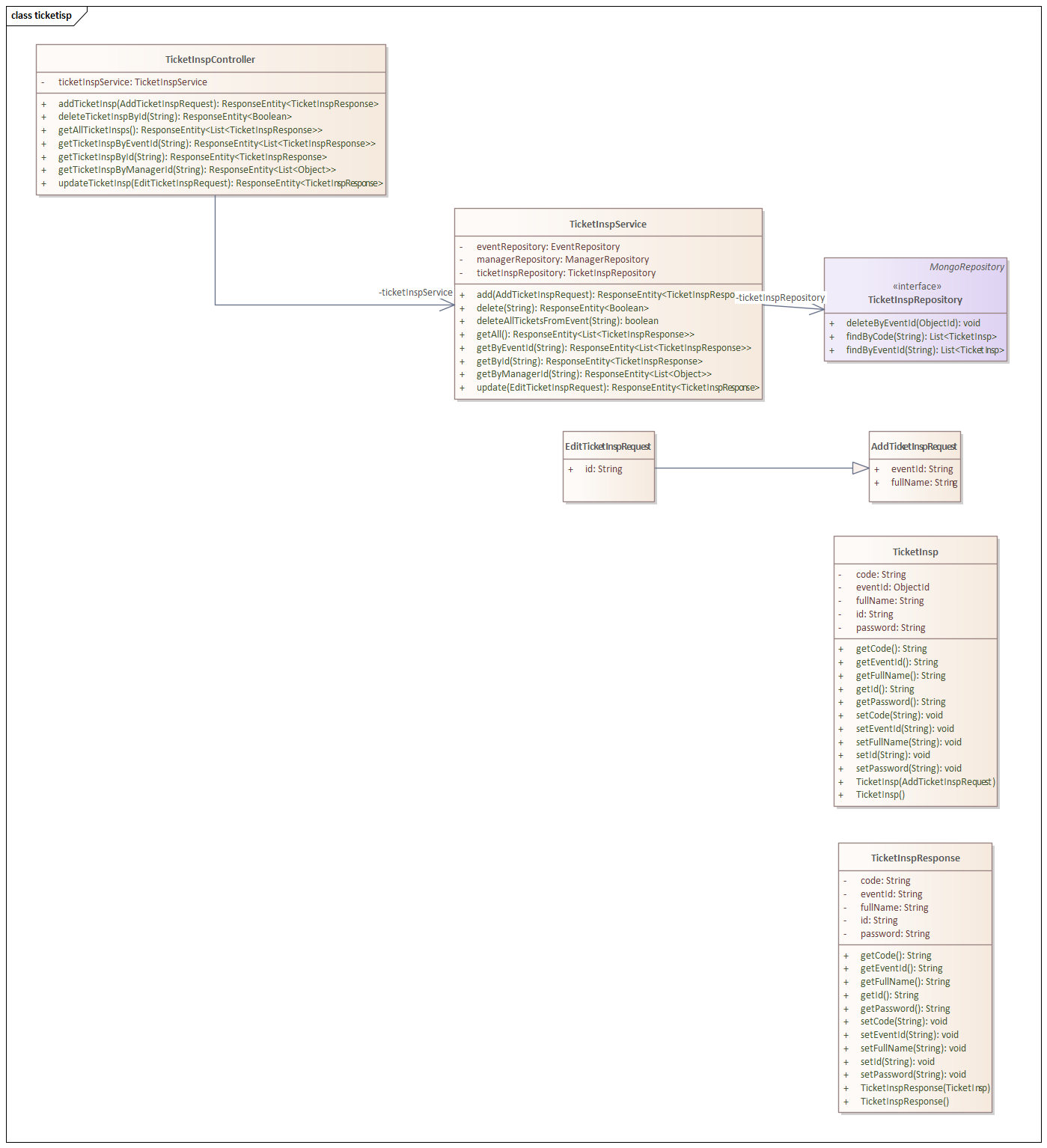
Modello 5.5 Event Class Diagram



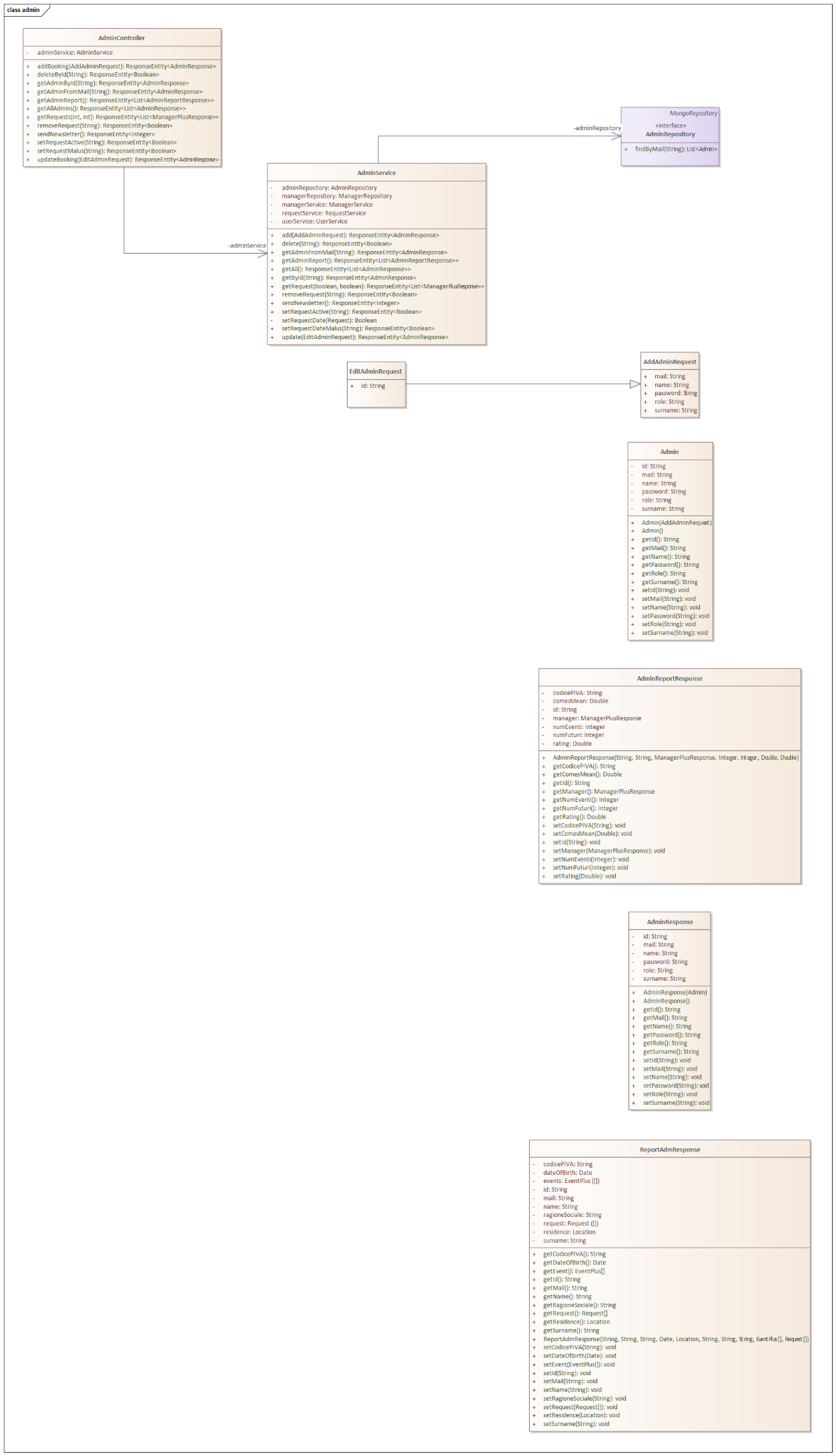
Modello 5.6 Booking Class Diagram



Modello 5.7 Manager Class Diagram



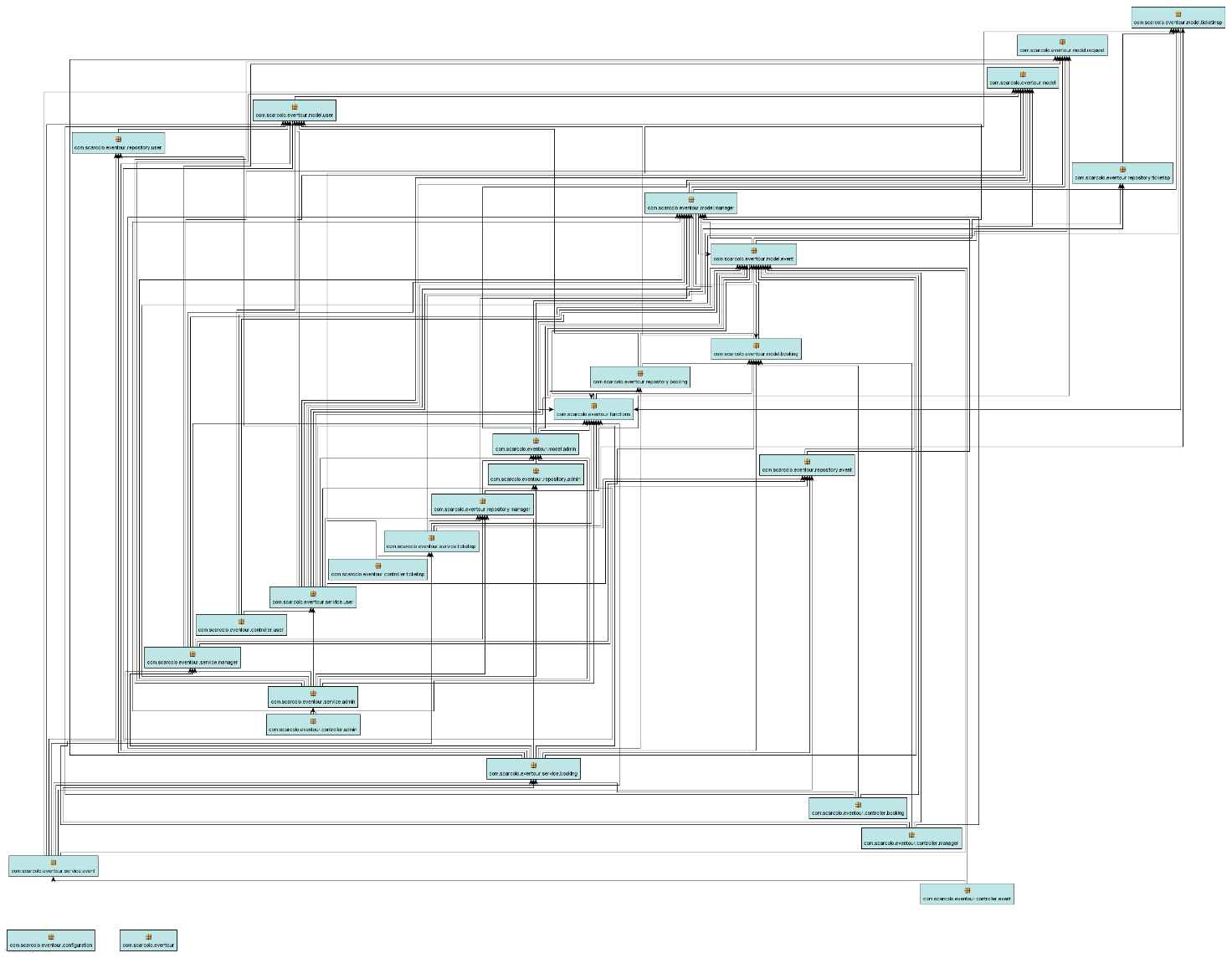
Modello 5.8 Ticket Inspector Class Diagram



Modello 5.9 Admin Class Diagram

A causa dell’aumento di complessità dei casi d’uso introdotti, anche le classi hanno avuto un incremento di metodi e funzioni introdotte, o addirittura l’introduzione di nuovi modelli e strutture applicanti il pattern architetturale CSR.

Si riporta di seguito il package diagram, dove si nota l’aumento di complessità.



Modello 5.10 Package Diagram

## Algoritmo evenTour

Lo scopo della versione corrente dell’algoritmo progettato è quella di trovare ancora un numero di eventi definito dalla invocazione della API, che siano disponibili per un numero definito di persone e che creino un tour di eventi, che l’utente può fare.

Per prima cosa, l’algoritmo ricerca l’utente in base al suo id (ipotizzando d utenti presenti e ipotizzando che MongoDB sfrutti l’indice come chiave, portando la ricerca a O(1)). Nel caso in cui l’utente non esista viene segnalato l’errore direttamente.

Vengono poi ricercati tutti gli eventi che hanno almeno nPersone posti disponibili, successivi alla data odierna e ordinati per data. Esso ha complessità tempo pari a O(Ntot) per la selezione degli eventi compatibili e O(n\*log n) nell’ordinamento di tutti i dati selezionati. Ipotizziamo che tutti gli eventi siano corrispondenti alla richiesta (n=Ntot)

Vengono poi selezionate tutte le prenotazioni fatte dall’utente (con complessità tempo O(M)).

L’algoritmo esclude poi, come nella iterazione precedente, tutti gli eventi già prenotati o gli eventi che hanno date uguali a essi. Ciò richiederà O(m\*n).





L’algoritmo si divide poi in due macroblocchi:

* Selezione dell’evento con distanza dall’utente minore di una distanza fissata e che sia ottimale per lui
* Selezione, successiva, degli eventi ottimali che siano a una distanza minore dall’evento precedentemente selezionato nella soluzione, con limitazioni in un range di date.

Per il primo blocco, l’algoritmo scorre tutti gli eventi filtrati, per ognuno calcola la distanza tra utente ed evento sfruttando i valori di latitudine e longitudine secondo la formula:

La funzione sottostante è l’implementazione di essa in codice dopo previa conversione delle latitudini e longitudini da gradi a radianti.



Dopo ciò, l’algoritmo, nel caso l’evento rispetti le condizioni precedentemente imposte di tipologia di evento, attribuisce un peso all’evento sulla base di uguaglianza o similitudine di tipi (in questo caso il peso è moltiplicato per 1.25 volte), distanza dall’utente e prezzo rispetto all’evento selezionato come ottimale fino a quel momento.

In questo modo l’algoritmo andrà a minimizzare questo peso per trovare l’evento che fornisce l’evento iniziale ottimale per l’utente. Nel caso in cui nessun evento venga selezionato, l’algoritmo restituisce un errore.

In maniera grafica si può vedere la scelta fatta dall’algoritmo di seguito:

Immagine che contiene testo, orologio, grafica vettoriale

Descrizione generata automaticamente

Figura 5.1 Scelta del miglior evento iniziale

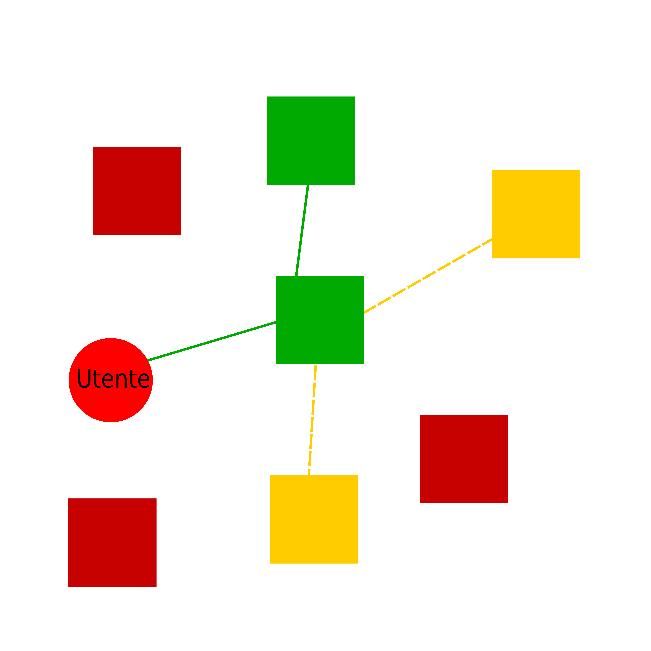




Per il secondo blocco, l’algoritmo scorre tutti gli eventi selezionati prima del filtro sulla distanza e inizia a riscorrerli tutti (tranne quello selezionato). Per ogni evento da selezionare successivamente per raggiungere il numero di eventi richiesti, l’algoritmo esclude gli eventi superiori a una distanza fissata dall’ultimo evento selezionato e che siano compresi tra la data dell’evento selezionato e le due settimane successive a esso. Per ognuno, scorre gli eventi filtrati e calcola la distanza tra evento selezionato precedentemente ed evento attuale sfruttando i valori di latitudine e longitudine secondo la formula espressa prima.

Dopo ciò, l’algoritmo, nel caso l’evento rispetti le condizioni precedentemente imposte di tipologia di evento, attribuisce un peso all’evento sulla base di uguaglianza o similitudine di tipi (in questo caso il peso è moltiplicato per 1.25 volte), distanza dall’utente e prezzo rispetto all’evento selezionato come ottimale fino a quel momento.

Si mostra quindi la selezione del secondo evento e del tour di eventi completo di seguito:

 Immagine che contiene testo, orologio

Descrizione generata automaticamente

Figura 5.2 Selezione secondo evento Figura 5.3 Selezione di tutti gli eventi

Dopo aver selezionato un numero di eventi richiesto dall’utente, l’algoritmo calcola la distanza totale e la spesa totale raggiunte con il tour, comprendendo anche il ritorno al comune di residenza come distanza, visibile in questa immagine:

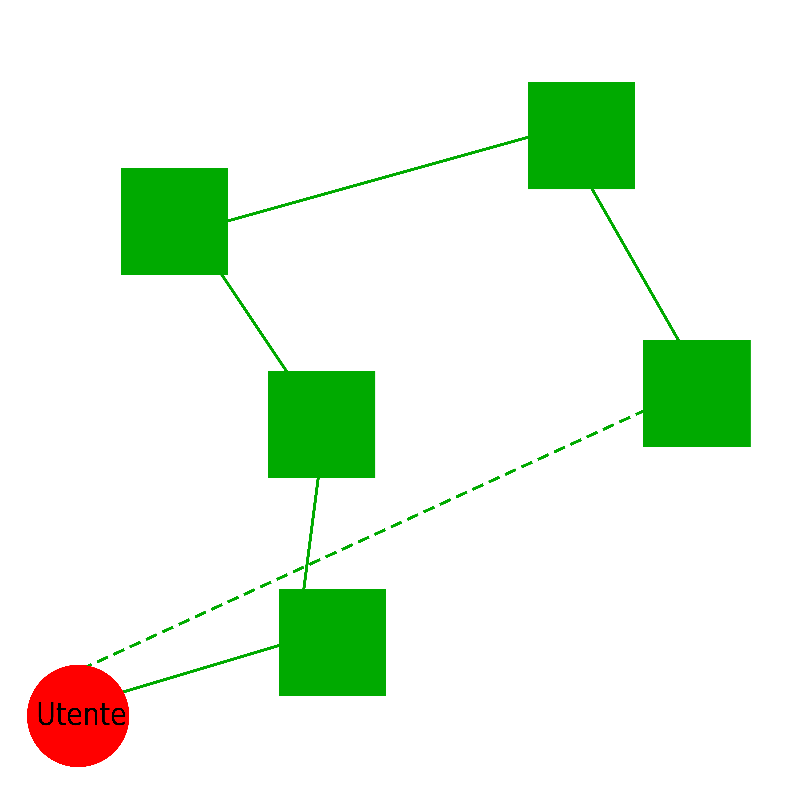


Figura 5.4 Vista della distanza percorsa

Si riporta lo pseudocodice dell’algoritmo rimanente comprendente di complessità tempo:





## Testing interfaccia grafica

### Analisi statica

* EsLint

### Analisi dinamica

* NightWatch



Casi di test User: iterazione 2



Casi di test Manager: iterazione 2



Casi di test Admin: iterazione 2

## Testing API

### Analisi statica

### Analisi dinamica

# Sviluppi futuri

Il progetto, per come è stato sviluppato, si trova in una fase avanzata se si guarda alle caratteristiche su cui si basa. Possibili sviluppi futuri riguardano senz’altro l’algoritmo, il quale può essere ulteriormente raffinato e ottimizzato, introducendo differenti parametri che permetteranno la generazione di molteplici soluzioni con la conseguente scelta della migliore secondo i gusti personali dell’utente.

Un ulteriore miglioramento può essere svolto sulla sicurezza dei dati salvati nel database, introducendo un sistema di crittografia che permetta il salvataggio delle password non in chiaro.

Un ulteriore sviluppo è il completamento delle funzionalità “mock” quali registrazioni reali di utenti e manager e pagamenti realmente attivi e l’invio di newsletter programmate, senza necessità da parte dell’amministratore di inviarla manualmente.

1. <https://www.7-zip.org/download.html> [↑](#footnote-ref-1)
2. <https://www.eclipse.org/downloads/packages/release/2021-06/r> [↑](#footnote-ref-2)
3. <https://drawio-app.com/> [↑](#footnote-ref-3)
4. <https://sparxsystems.com/> [↑](#footnote-ref-4)
5. <https://www.uml-lab.com/en/uml-lab/> [↑](#footnote-ref-5)
6. <http://www.dependency-analyzer.org/> [↑](#footnote-ref-6)