Documento di Architettura

Edoardo Ghirardello, Giulio Cappelli, Elia Casotti

Gruppo T42

2022

Indice

1 Scopo del documento

Nel presente documento viene definita l'architettura del sistema "Fen Festa" utilizzando il Class Diagram UML e codice in OCL (Object Constraint Language).
Nel precedente documento sono stati rappresentati i componenti del sistema che ora verranno tradotti in classi attraverso il diagramma UML indicato.

2 Diagramma delle Classi

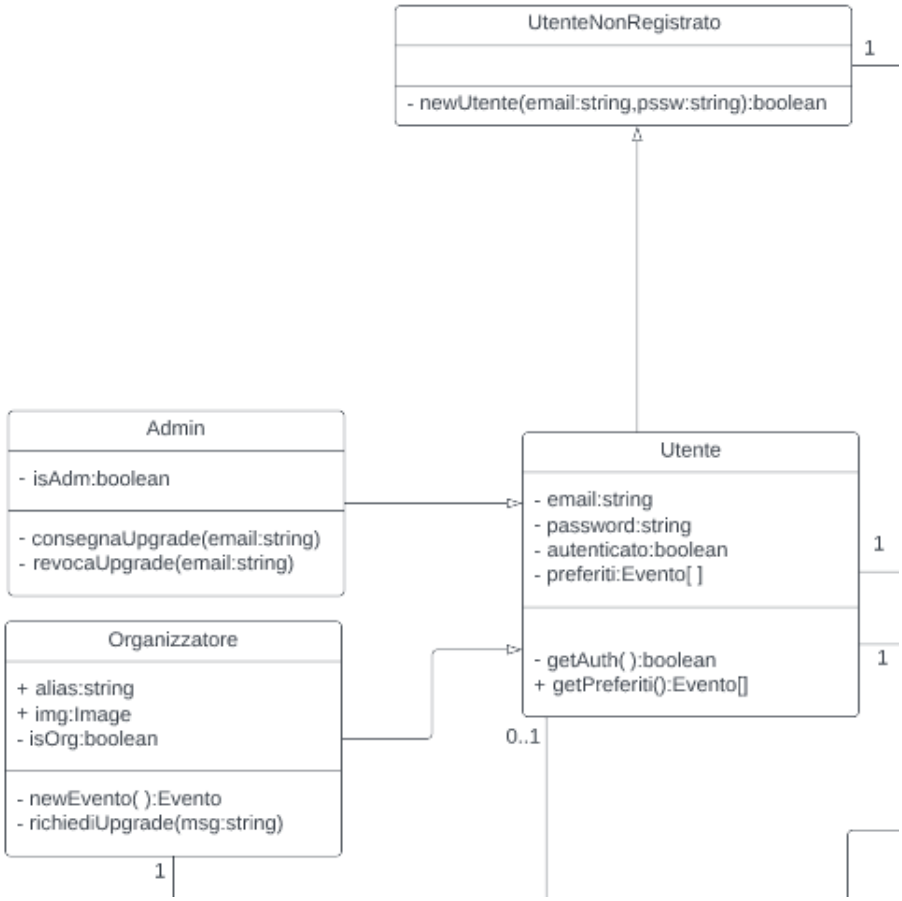
Nel presente capitolo vengono rappresentate le classi previste nell'ambito del progetto "Fen Festa". Ogni componente presente nel diagramma dei componenti diventa una classe. Tutte le classi individuate sono caratterizzate da un nome, una lista di attributi e una lista di metodi che definiscono le operazioni previste all'interno della classe. Associazioni tra le diverse classi forniscono informazioni sulle relazioni tra esse. Riportiamo di seguito le classi individuate.

2.1 Utenti

Analizzando il diagramma di contesto e quello dei componenti si individua la presenza di diversi attori per definire diversi lielli di "User". In particolare si identificano lo user non autenticato **UtenteNonRegistrato** da cui deriva quello autenticato **Utente** e i due da esso derivati **Organizzatore** e **Admin**.

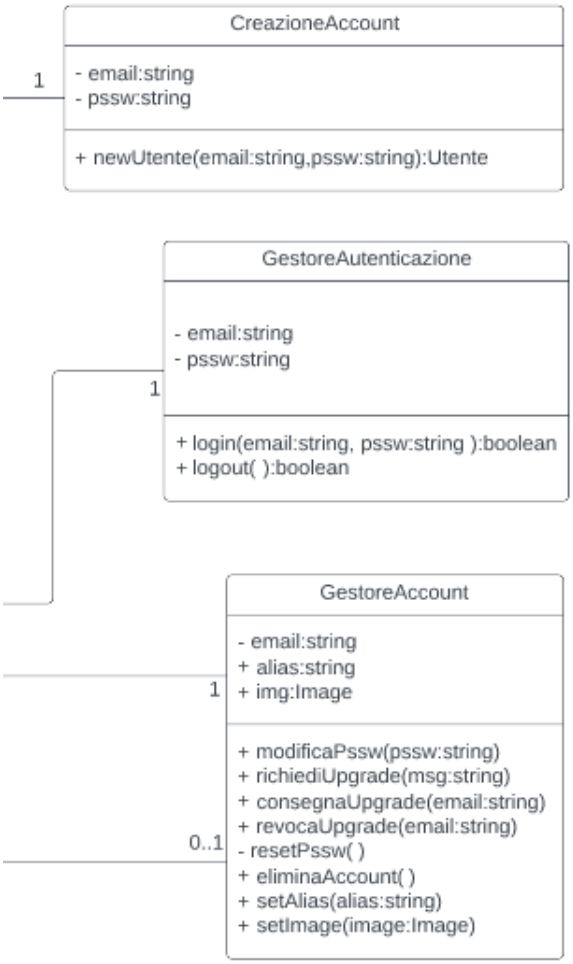
Utente è colui che utlizza l'app per vedere gli eventi disponibili, abbia esso effettuato il login o meno (il login offre più funzionalità specificate in documenti precedenti che verranno meglio identificate successivamente nella sezione inerente all'OCL).

Organizzatore e **Admin** sono due tipi specifici di utente che dispongono di autorizzazioni extra, sono quindi classi derivate tramite generalizzazione dalla classe **Utente**



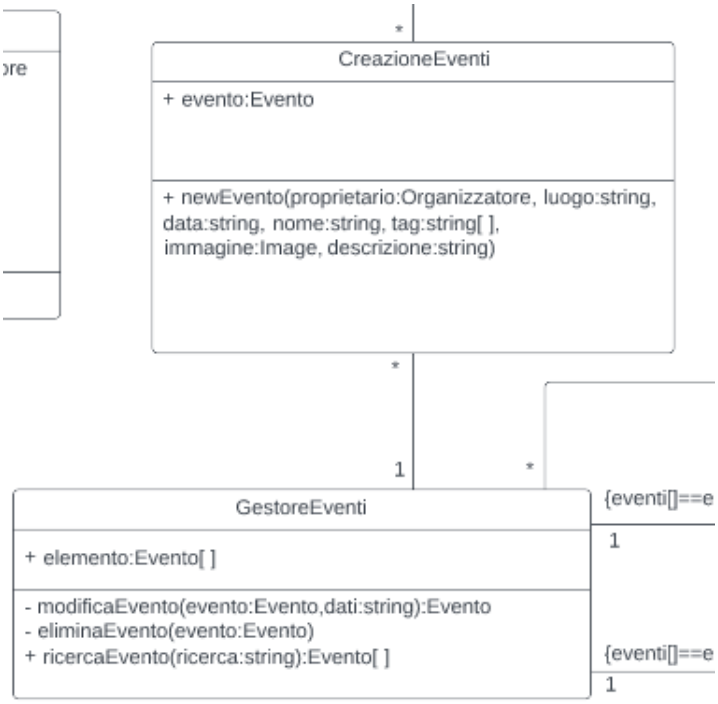
2.2 Gestione Autenticazione e Account

Il diagramma dei componenti include diversi componenti per la gestione degli account a partire dalla creazione di un nuovo account, passando dall'autenticazione e permettendo la gestione del proprio account. Definiamo quindi le classi **CreazioneAccount**, **GestoreAutenticazione** e **GestoreAccount**. La classe **CreazioneAccount** viene utilizzato unicamente dall'utente non ancora registrato, permettendo poi all'utente di autenticarsi. Il **GestoreAccount** presenta diversi metodi usati sia da utenti normali che da Admin e Organizzatori con l'autorizzazione specifica.



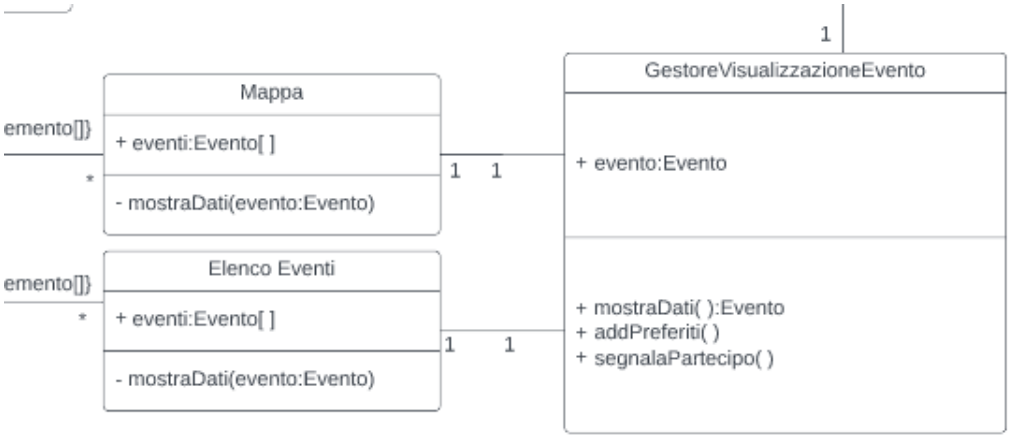
2.3 Creazione e Gestione Eventi

Il diagramma dei componenti identifica due componenti per la gestione degli eventi, dividendo il processo in una prima parte di creazione e una seconda di gestione. Avendo accesso all'intero elenco di eventi la gestione si occupa anche di effettuare la ricerca di un evento. Vengono quindi identificate le rispettive classi **CreazioneEventi** e **GestoreEventi**.



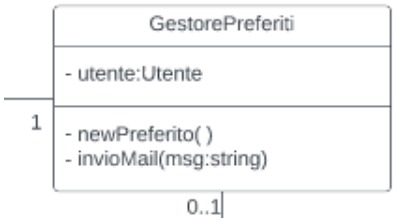
2.4 Visualizzazione degli eventi

Il diagramma dei componenti identifica i componenti **Gestore Visualizzazione Evento**, **Mappa** e **Elenco Eventi** che assieme si occupano di mostrare agli utenti gli eventi. Creando una classe per ogni componente identifichiamo **Mappa**, **ElencoEventi** e **GestoreVisualizzazioneEvento**. L'utente si interfaccia con queste classi per ricercare e visualizzare gli eventi salvati in app. Può visualizzare i dati del singolo evento mostrati dal **GestoreVisualizzazioneEvento**, il quale interagendo con altre classi permette l'aggiunta dell'evento ai preferiti.



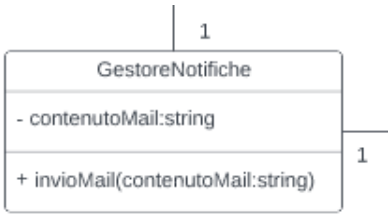
2.5 Gestione Preferiti

Il diagramma dei componenti identifica un gestore dei preferiti per permettere all'utente autenticato di salvarsi gli eventi a cui è interessato. Identifichiamo quindi la classe **GestorePreferiti**.



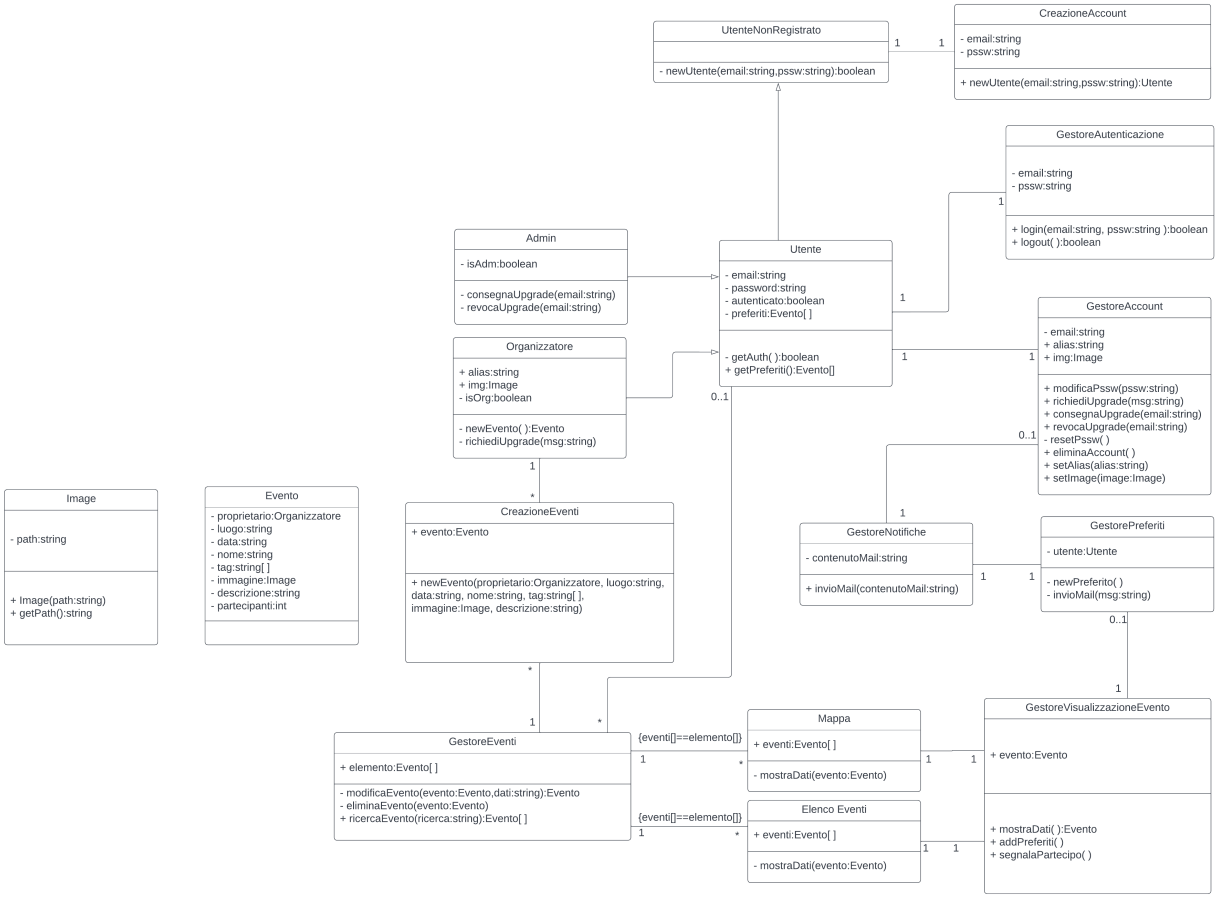
2.6 Gestione Notifiche

Il diagramma dei componenti identifica un gestore delle notifiche che si occupa dell'invio di email per il recupero della password e per notificare l'utente autenticato dei suoi eventi preferiti. Identifichiamo la classe **GestoreNotifiche** per gestire questi messaggi in base alle richieste che riceve dal **GestorePreferiti** e dal **GestoreAccount**.



2.7 Diagramma delle Classi Complessivo

Riportiamo di seguito il diagramma delle classi con tutte le classi presentate. È presente inoltre una classe ausiliaria, "Evento", per descrivere il tipo di dato "Evento" utilizzato da altre classi nel diagramma.



3 Codice in Object Constraint Language

In questo capitolo è descritta in modo formale la logica prevista nell'ambito di alcune operazioni di alcune classi. Tale logica viene descritta in Object Constraint Language (OCL).

3.1 Operazioni Account Utente

È presente una serie di restrizioni in base a se l'utente ha già effettuato il login oppure no e in base al tipo di utente. In particolare non è possibile creare un utente se ne egiste già uno con la stessa mail; la modifica della password, la richiesta di upgrade, l'eliminazione dell'account e il logout richiedono di aver già effettuato l'accesso; la richiesta di upgrade richiede inoltre che l'utente non sia già organizzatore mentre il logout cambia lo stato da autenticato a non autenticato; il reset della password e il login richiedono che l'utente non sia già autenticato (e il login cambia lo stato da non autenticato ad autenticato).

```
context CreazioneAccount.newUtente( )
pre Utente.email=null
context GestoreAccount::modificaPssw( )
pre Utente.autenticato=1
context GestoreAccount::richiediUpgrade( )
pre Utente.autenticato=1
pre Organizzatore.isOrg=0
context GestoreAccount::resetPssw( )
pre Utente.autenticato=0
context GestoreAccount.eliminaAccount( )
pre Utente.autenticato=1
post Utente=null
context GestoreAutenticazione::login( )
pre Utente.autenticato=0
post  Utente.autenticato=1
context GestoreAutenticazione::logout( )
pre Utente.autenticato=1
post Utente.autenticato=0
```

3.2 Autorizzazioni Admin

Solo l'admin ha accesso alla consegna degli upgrade degli account e alla revoca degli stessi, previa sua autenticazione.

```
context GestoreAccount::consegnaUpgrade( )
pre: Utente.autenticato=1
pre: Admin.isAdm=1
context GestoreAccount::revocaUpgrade( )
pre: Utente.autenticato=1
pre: Admin.isAdm=1
```

3.3 Autorizzazione Creazione Nuovo Evento

Per poter creare un nuovo evento è necessario che l'utente sia autenticato e sia un organizzatore.

```
context Organizzatore::newEvento( )
pre: Utente.autenticato=1
pre: Organizzatore.isOrg=1
```

3.4 Autorizzazioni Gestione Evento

Per poter modificare o eliminare un evento è necessario essere admin oppure essere l'organizzatore proprietario dell'evento che si intende eliminare/modificare.

```
context GestoreEventi::modificaEvento( )
pre: Utente.autenticato=1
pre: isAdm=1 OR (isOrg=1 AND Evento.proprietario=Organizzatore)
context GestoreEventi::eliminaEvento( )
pre: Utente.autenticato=1
pre: isAdm=1 OR (isOrg=1 AND Evento.proprietario=Organizzatore)
```

3.5 Gestione Preferiti

Il gestore preferiti permette unicamente all'utente autenticato di aggiungere un evento alla lista degli eventi preferiti se questo non è già presente e di rimuoverlo in caso contrario.

```
context GestorePreferiti::newPreferito( )
pre Utente.autenticato=1
pre Evento not in Utente.preferiti[ ]
post Evento in Utente.preferiti[ ]
pre Evento in Utente.preferiti[ ]
post Evento not in Utente.preferiti[ ]
```

4 Diagramma delle Classi con codice OCL

Riportiamo infine il diagramma delle classi con tutte le classi fino ad ora presentate ed il codice OCL individuato.