



# Documentazione per il progetto d'esame per il corso di Programmazione per il web



UNIVERSITÀ DEGLI STUDI DELL'AQUILA

**Facce Beve**

## ▼ Candidati:

Samuele Ramassone

Matteo Di Russo-Ciccarelli

Matteo Angelucci

**Anno Accademico 2021-2022**

---

## **Proposta Di Progetto**

Di seguito è esposta la documentazione della proposta di progetto eseguito dai candidati:

**Applicazione:** FacceBeve

**Descrizione:** L'applicazione web ha come obiettivo far interagire tra loro utenti e proprietari di locali, attraverso la visualizzazione e delle informazioni dei locali che si desidera visitare. Si presenta, quindi, come una vetrina dove sono catalogati locali dove andare a divertirsi di qualsiasi città e tipo.

Generalmente l'utente può effettuare la ricerca del locale di interesse ,e visualizzarne le recensioni, in base alle sue esigenze mediante: il nome, la selezione della città e/o il tipo di locale ricercato.

L'utente può o meno registrarsi, la registrazione porta vantaggi che comprendono: il poter scrivere una recensione ad un locale, salvare un determinato locale tra i preferiti e vedere gli eventi programmati dai locali.

Il proprietario è un tipo utente registrato, si dovrà sempre registrare, ma con un tipo di registrazione che è specializzata per questo tipo di utente, il proprietario potrà quindi

inserire uno o più locali che vuole pubblicizzare. In più gli è possibile programmare eventi (ad esempio: Festa di San Patrizio, serata Karaoke, ...) che risulteranno essere visibili solo all'utente registrato.

Il locale assume importanza in base alle recensioni lasciate dagli utenti registrati, in quanto, grazie ad esse, è possibile ordinare i locali dal “migliore” al “peggiore”, dandogli più o meno visibilità.

### **Tipologie Utenti:**

- *Utente non registrato;*
- *Utente registrato;*
- *Proprietario del locale;*
- *Amministratore dell'applicazione.*

### **Obiettivi Utenti:**

#### **→*Utente non registrato:***

- ◆ può registrarsi;
- ◆ può cercare locali, in base a nome, città, tipo di locale;
- ◆ può visualizzare le recensioni e le informazioni di ogni locale.

#### **→*Utente registrato:***

- ◆ ha tutte le funzionalità dell'utente non registrato;
- ◆ può scrivere recensioni del locale;
- ◆ può salvare i suoi locali preferiti;
- ◆ può vedere se ci sono eventi in un locale;
- ◆ può cercare di determinati eventi.

#### **→*Proprietario del locale:***

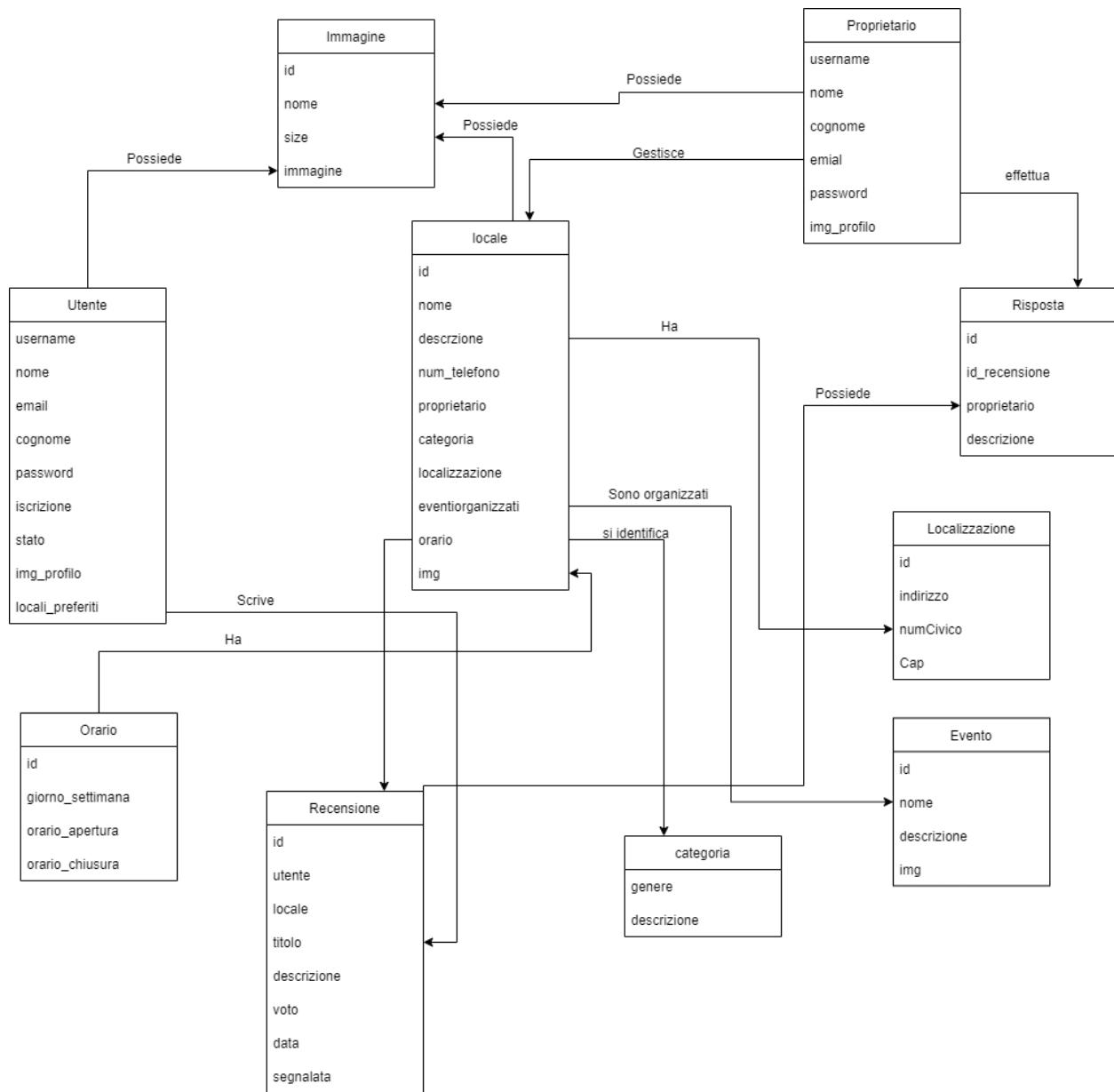
- ◆ deve effettuare la registrazione come “Proprietario di locali”;
- ◆ può registrare il suo locale
- ◆ può rispondere alle recensioni, eventualmente segnalare quelle inopportune;
- ◆ può programmare un evento in un determinato locale ;
- ◆ può rimuovere il locale dall'applicazione.

→*Amministratore dell'applicazione:*

- ◆può visualizzare gli utenti attivi e bannati;
  - ◆può aggiungere/eliminare delle categorie di locali
  - ◆può bannare/attivare gli utenti ;
  - ◆può eliminare recensioni ;
- 

## Analisi

Lo strumento utilizzato per analizzare la realtà d'interesse è quello del Modello Logico/Concettuale:



## Casi D'uso

Dopo aver analizzato il quadro generale attraverso il modello logico si passa ad analizzare i casi d'uso.

Sono stati individuati i seguenti casi d'uso:

## **1-RICERCA LOCALI:**

Precondizioni: nessuna, anche un utente non registrato può ricercare un locale.

Scenario:

1. L'utente riempie un campo della ricerca, o tutti e tre i campi di ricerca;
2. Il sistema ricerca e mostra i locali o il locale richiesto. Per ogni locale vengono fornite delle informazioni parziali
3. L'utente sceglie il locale per poi visualizzarlo in dettaglio
4. Il sistema restituisce la pagina di informazione del locale selezionato

## **2-RICERCA EVENTI:**

Precondizioni: L'utente deve essere loggato come user.

Scenario:

1. L'utente seleziona eventi come tipo di ricerca;
2. L'utente riempie un campo della ricerca, o tutti e tre i campi di ricerca;
3. Il sistema ricerca e mostra gli eventi o l'evento richiesto. Per ogni evento vengono fornite delle informazioni parziali;
4. L'utente sceglie l'evento per poi visualizzarlo in dettaglio
5. Il sistema restituisce la pagina di informazione del locale selezionato

## **3-AGGIUNTA LOCALE:**

Precondizioni: Precondizioni: L'utente deve essere loggato come proprietario.

1. Il proprietario accede alla propria area personale
2. Seleziona 'Aggiungi Locale'
3. Inserisce tutti i campi richiesti nella form e salva

## **4-CREAZIONE EVENTO:**

Precondizioni: L'utente deve essere loggato come proprietario.

1. Il proprietario accede alla propria area personale
2. Seleziona il locale per cui vuole creare un evento cliccando su ‘Gestisci Locale’
3. Clicca su ‘aggiungi evento’
4. Riempie la form e salva.

## **5-RECENSIRE UN LOCALE:**

Precondizioni: L’utente deve essere loggato come user.

1. L’utente ricerca il locale che vuole recensire;
2. L’utente accede alla pagina del locale cercato;
3. In fondo alla pagina l’utente scrive nell’ apposita form la sua recensione, dove è obbligatorio scrivere almeno il titolo e dare un voto;
4. Clicca sul pulsante aggiungi recensione.

## **6-RISPONDERE A UNA RECENSIONE:**

Precondizioni: L’utente deve essere loggato come proprietario.

1. Il proprietario dopo aver cercato il proprio locale ed essere entrato nella pagina del locale;
2. Nella sezione delle recensioni, sotto ad ogni recensione, il proprietario trova una area di testo dove può rispondere.

## **7-BANNARE UTENTI:**

Precondizioni: L’utente deve essere loggato come admin

1. Nella sezione ‘UTENTI ATTIVI’ si cerca l’utente da bannare e si clicca sul pulsante.

## **8-ATTIVARE UTENTI:**

Precondizioni: L’utente deve essere loggato come admin

1. Nella sezione ‘UTENTI BANNATI’ si cerca l’utente da bannare e si clicca sul pulsante.

## **9-AGGIUNGERE UNA CATEGORIA:**

Precondizioni: L’utente deve essere loggato come admin

1. Nella sezione categorie, scrivere il nome e descrizione
2. Cliccare su aggiungi

## **10-ELIMINARE UNA CATEGORIA:**

Precondizioni: L’utente deve essere loggato come admin

1. Nella sezione categorie, cercare la categoria
2. Cliccare su elimina

## **11-ELIMINARE UNA RECENSIONE:**

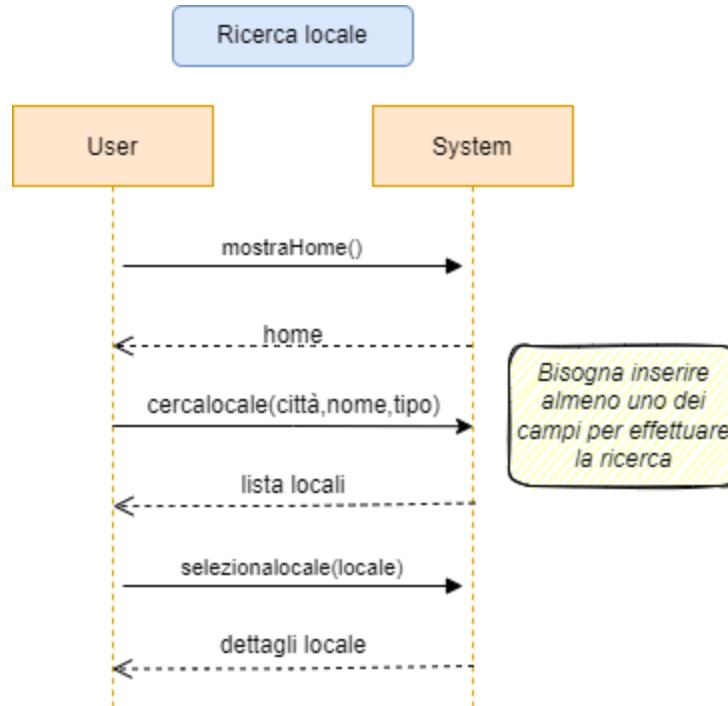
Precondizioni: L’utente deve essere loggato come admin.

1. Nella sezione recensioni, cercare la recensione desiderata;
2. Cliccare sul pulsante

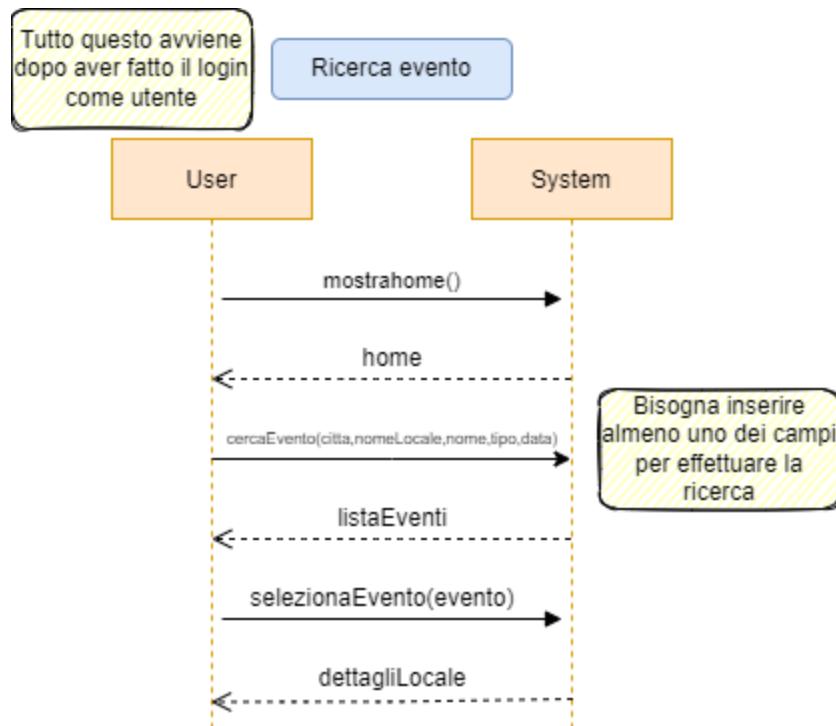
# **Diagrammi di sequenza**

Nei diagrammi di sequenza, per avere informazioni sul funzionamento dell’applicazione, abbiamo esploso alcuni casi d’uso.

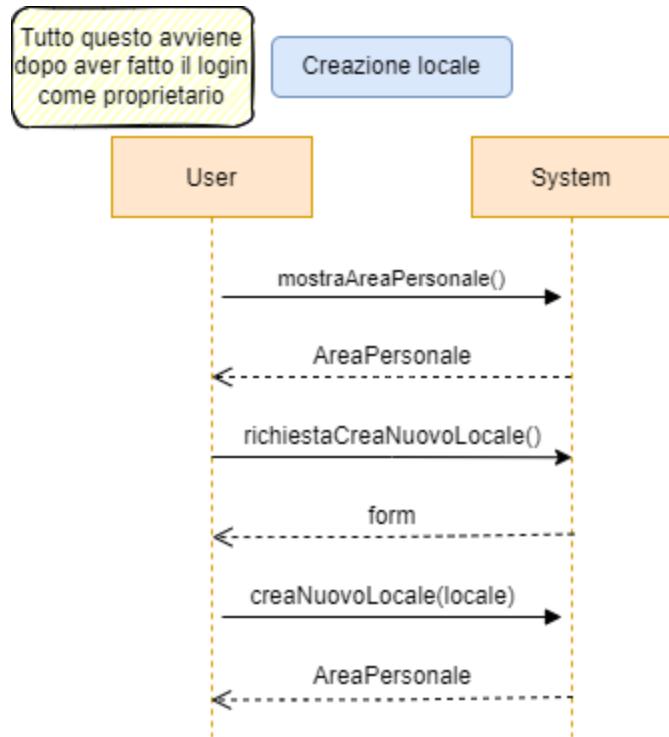
## **1-RICERCA LOCALI:**



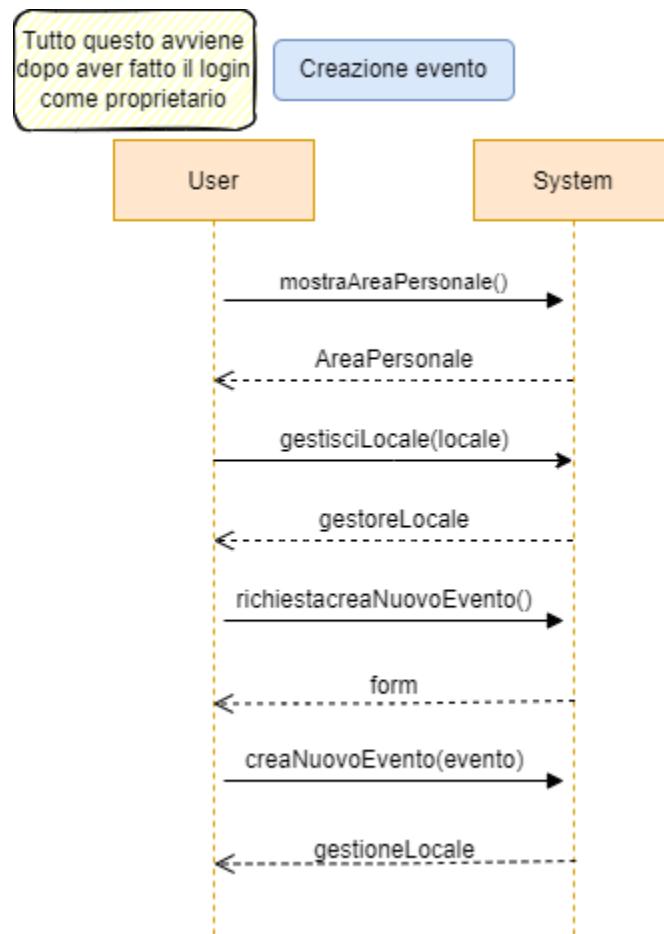
## 2-RICERCA EVENTI:



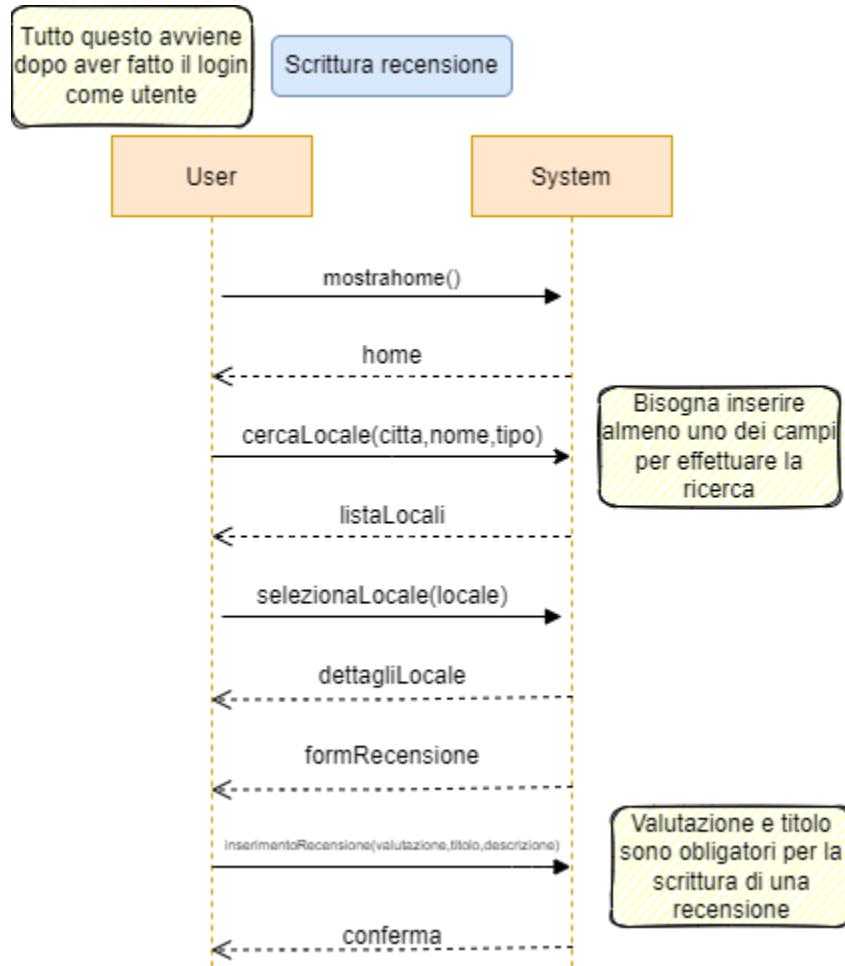
### 3-AGGIUNTA LOCALE:



### 4-CREAZIONE EVENTO:

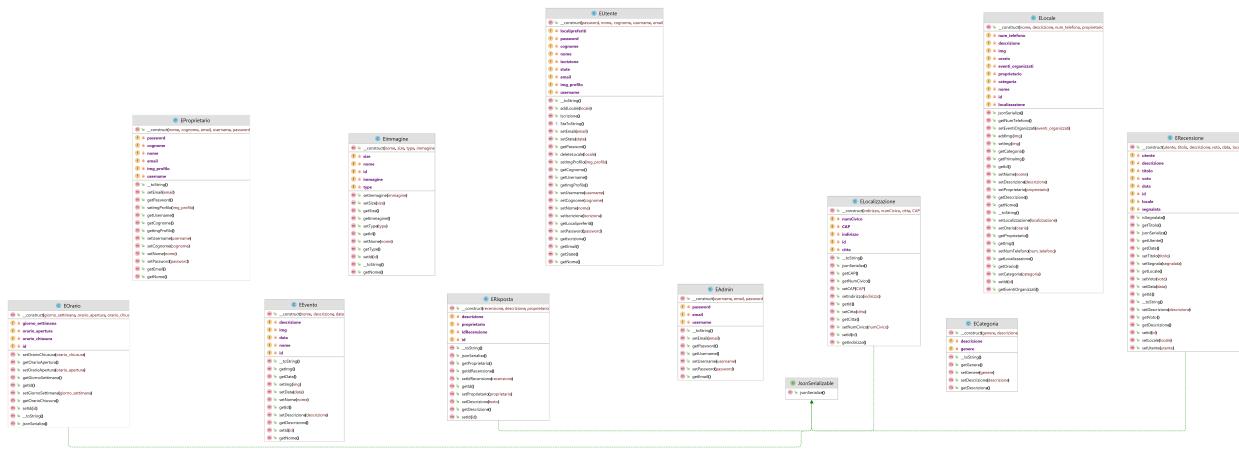


## 5-RECENSIRE UN LOCALE:

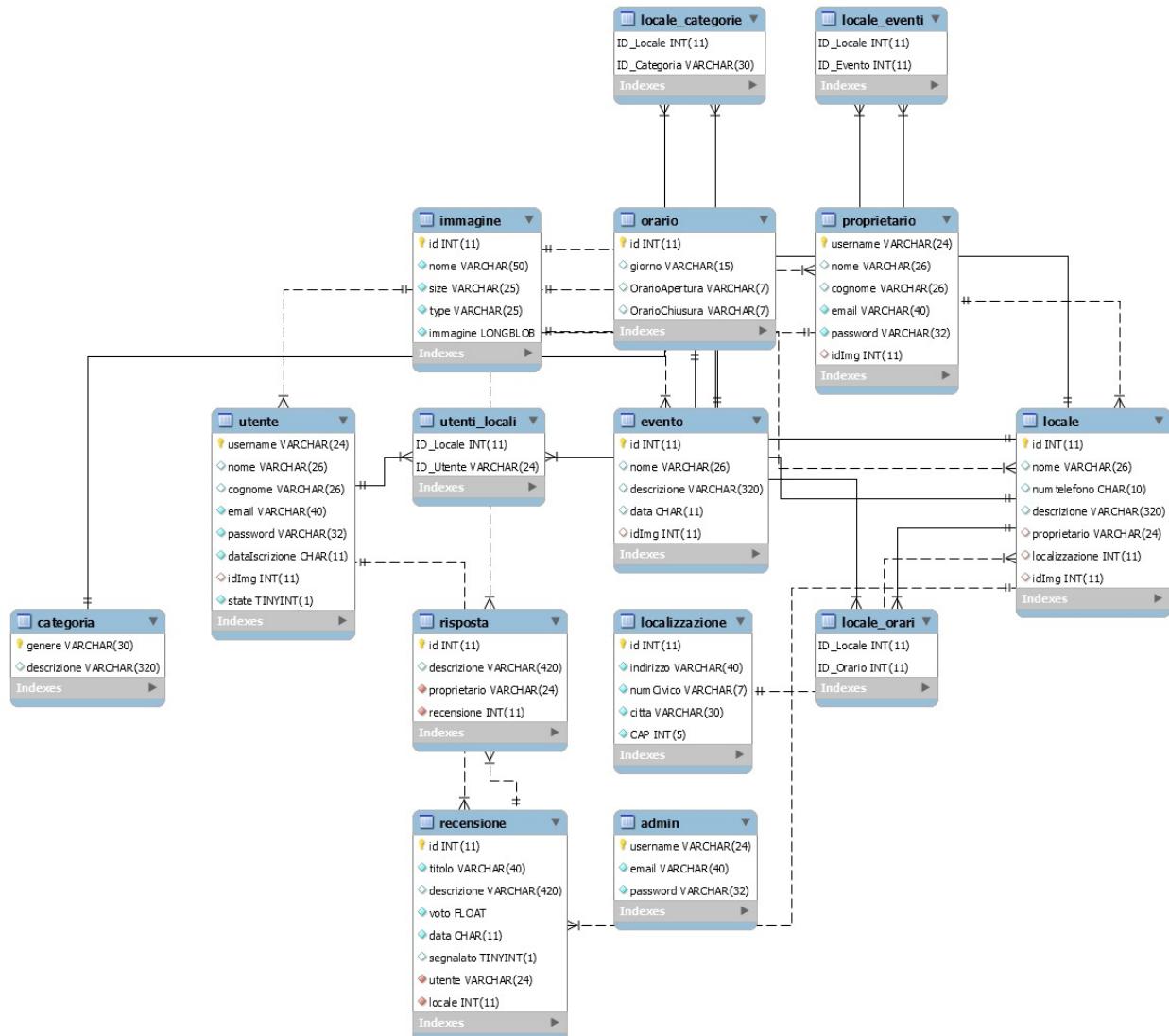


## Design del sistema

### Diagramma delle classi:



## Schema del database:



# Progettazione PCEF

Per la progettazione del codice, abbiamo utilizzato un **Layered Design** conosciuto come **PCEF**. Con questo pattern distinguiamo quattro **strati**, che sono:

- ▼ **Foundation**: responsabile della persistenza dei dati, conosce l'implementazione a livello fisico dei dati e si interfaccia con il DB.
- ▼ **Entity**: definisce gli oggetti specifici del domino applicativo e implementa i requisiti funzionali.
- ▼ **Controller**: definisce classi che disaccoppiano gli strati View ed Entity, conoscono quali classi implementano un determinato caso d'uso.
- ▼ **View**: contiene le classi che realizzano la GUI, permettendo le interazioni uomo-macchina.

Seguono le scelte fatte per ognuno di questi strati.

#### **FOUNDATION:**

Le classi Foundation condividono i metodi relativi alle cosiddette operazioni CRUD, oltre ad

avere metodi utili per facilitare il corretto funzionamento dell'applicazione. Per la progettazione dello strato 'F' abbiamo creato una superclasse **FDb** dalla quale tutte le classi di Foundation ereditano i metodi. In questa sono definiti gli attributi: una variabile che contiene l'oggetto connection verso il db, un attributo che rappresenta un array di istanze che viene utilizzato per l'approccio singleton, ed altri attributi che nelle singole classi che estendono FDb verranno impostati e saranno utili nell'interazione con il DB.

I metodi di questa superclasse corrispondono alle operazioni basilari e generiche che si possono fare

verso un db come **store**, **load**, **update**, **delete**, e **search**. Tutti questi restituiscono degli oggetti di

tipo statement, che poi verranno processati da altri metodi all'interno della classe, in base alle necessità che ci risultano necessarie.

Per quanto riguarda l'esecuzione di **query** sul db è stato progettato un apposito metodo che,

una volta passata una stringa che rappresenta una query, la esegue prima facendo un **prepare** dello

statement, operazione che serve per ottimizzare l'esecuzione della query stessa, e successivamente,

con il metodo **execute**, si verifica l'esecuzione vera e propria.

La classe FDb successivamente viene estesa da tutte le altre classi Foundation, nelle quali è

personalizzato il funzionamento dei metodi principali e, in base alle necessità, sono stati aggiunti

metodi opportuni.

### **ENTITY:**

Nel layer Entity abbiamo implementato le classi che modellano il dominio d'interesse.

Dove principalmente per ogni classe sono scritti metodi getter e setter.

### **CONTROLLER:**

I controllori propagano allo strato Entity o direttamente in Foundation lo stimolo ricevuto dalle

View da parte dell'utente finale. Essendo che questo stimolo rappresenta un'operazione che l'utente

richiede all'applicazione, i controllori vengono definiti per ogni caso d'uso. Il loro compito è quello

di coordinare il lavoro delle classi a livello sottostante per soddisfare lo stimolo ricevuto e di rendere il risultato alle classi View specifiche.

Abbiamo deciso anche qui di utilizzare un' approccio Singleton, visto che non servono oggetti

diversi dalle classi stesse di Controller.

Nella progettazione, si è deciso di inserire al loro interno l'interazione con gli oggetti delle classi

View, questo perché viene implementato il meccanismo del rewriting delle URL. A fronte di questa

implementazione si è dovuto sviluppare un controllore specifico chiamato Front Controller che,

richiamato esclusivamente dall'index.php, analizzando l'URL riesce a richiamare il giusto metodo

del giusto controllore; per questo motivo si è rivelato necessario avere dei controllori che utilizzassero oggetti delle classi View al loro interno e non il contrario.

Nel layer Controller è stata creata una classe per ogni caso d'uso ed un metodo per ogni evento

di interazione dell'utente con il sistema.

#### **VIEW:**

Le classi di questo layer implementano la GUI e realizzano l'interfaccia allo scopo di acquisire l'input

e visualizzare l'output. Per rispettare questi principi, abbiamo sviluppato una classe View per ogni

caso d'uso ed un metodo per ogni operazione di input ed output che si va ad eseguire.

---

## **Scelte progettuali**

### **Package ‘Template’:**

Per quanto riguarda le tecnologie web, abbiamo creato una cartella chiamata ‘Template’ dove abbiamo inserito tutto ciò che concerne quest’ambito.

Abbiamo cercato un template bootstrap che fosse il più possibile funzionale ai nostri scopi.

Una volta trovato il template giusto si è passati alla fase di personalizzazione di quest’ultimo,

creando tutte le pagine che sarebbero state necessarie, seguendo uno stile comune tra tutte e cercando di farle nostre con l’aggiunta di HTML e CSS. Inoltre sono state aggiunte diverse funzioni in JavaScript per la validazione dei form di registrazione,

controllo dei campi della ricerca e tutti i controlli sulle stringhe in input.

Per quanto riguarda la generazione delle pagine in base ai dati che vengono passati dai layer inferiori, abbiamo utilizzato il template-engine Smarty. La presentazione in ambito web è costituita da un testo contenente i dati da visualizzare e la presentazione dei dati fatta attraverso HTML.

Abbiamo creato tutti i template, file con estensione '.tpl', che ci sarebbero stati necessari, sfruttando le funzionalità di Smarty, e successivamente abbiamo fatto agire le classi View con degli appositi attributi \$smarty; questi, al momento della creazione di un oggetto delle classi View, vengono inizializzati e configurati utilizzando un'apposita classe che prende il nome di 'StartSmarty.php'.

## **Sessioni:**

Il meccanismo delle sessioni va in contro alle esigenze delle applicazioni che vogliono mantenere lo stato di una connessione. Le informazioni che si decidono di mettere in sessione saranno archiviate nel server e saranno accessibili solamente utilizzando un session id, cioè un identificativo univoco associato alla sessione.

Nella progettazione dell'applicazione web abbiamo deciso di utilizzare le sessioni nel caso in cui si accede, tramite login all'area personale, sia per un utente, un proprietario o un amministratore. Una volta che il login è avvenuto con successo vengono salvati sia l'id dell'utente (univoco), sia la tipologia di interlocutore. Quest'ultima viene salvata perché in base a chi sta interagendo con l'applicazione, si avranno interfaccia e funzionalità differenti.

Quando l'utente effettuerà il logout, ci sarà sia un unset della sessione, cioè verranno cancellate le informazioni della sessione, e sia un destroy per eliminare la sessione.

Per l'utilizzo delle sessioni abbiamo creato una classe apposita che prende il nome di 'USSession.php', con dei metodi che ci permettono di effettuare qualsiasi operazione con una sessione.

Una decisione progettuale che abbiamo preso è stata quella di modificare il tipo di cookie che

contiene il 'PHP\_SES\_ID', e impostargli una scadenza dopo 15 minuti, così da evitare che ci siano sessioni troppo prolungate nel tempo.

## Rewriting:

Il meccanismo di URL rewriting è quello che permette di avere URL descrittive rispetto alla situazione che sta accadendo e ha il compito di nascondere all'utente molte informazioni che non dovrebbe sapere.

Nel nostro caso abbiamo deciso di implementare il meccanismo di riscrittura delle URL. Abbiamo creato un file .htaccess, cioè un file di configurazione distribuita. Questo file contiene:

- le direttive per attivare il rewriting;
- condizioni che verificano se nel browser siano presenti la directory;
- condizione di base per la quale si inizia il rewriting dall'inizio del URL;
- la regola di rewriting che dice che tutte le URL devono essere indirizzate verso il file con nome 'index.php'.

Quest'ultimo è un file molto semplice, nel quale si richiama il controllore FrontController e si esegue il suo metodo 'run', al quale gli si passa il '\$\_SERVER['REQUEST\_URI']', cioè il path.

Il FrontController con il suo metodo 'run', analizza l'URL e indirizza la richiesta verso il giusto controllore.