**Nome del progetto:** ShareChef

**Funzionalità realizzate lato server:**

-Login.

-Creazione evento culinario (pranzo, cena, aperitivo).

-Applicazione filtri esclusivi per la creazione dell’evento (età, numero massimo di persone, nome evento, prezzo).

-Applicazione filtri non esclusivi per la creazione dell’evento (interessi).

-Partecipazione ad evento culinario.

-Sistema di recensione per il creatore dell’evento a cui si partecipa.

-Bacheca per singolo evento (un utente partecipante all’evento può scrivere cose).

-Lista degli eventi.

-Ricerca con filtri degli eventi.

-Suggerimenti eventi in base agli interessi ed alla distanza.

-Profilo utente.

-Elenco eventi passati e futuri (creati e a cui partecipi).

-Sistema di valutazione.

-Aggiornamento dati utente.

-Possibilità di visitare il profilo di un altro utente e i suoi eventi (creati e a cui partecipa).

**SPECIFICHE**

**Ognuna delle URL che seguono è preceduta dal prefisso rest/auth/{SID}/, in cui {SID} è un token di sessione restituito da rest/auth/login o rest/auth/utenti.**

**URL**: http://auth/login

**VERBO**: POST

**RISPOSTA**: Prende in input un oggetto contenente username e password di un utente e restituisce un oggetto contenente il relativo token di sessione/autenticazione.

**URL**: http://auth/logout

**VERBO**: POST

**RISPOSTA**: Prende in input un token di sessione/autenticazione e chiude la relativa sessione, invalidando il token.

**URL**: http://auth/utenti

**VERBO**: POST

**RISPOSTA**: Inserisce un nuovo utente nel sistema. L’oggetto passato come payload conterrà tutti i dati del profilo utente.

**URL**: http://auth/{SID}/utenti/{ID}

**VERBO**: GET

**RISPOSTA**: Restituisce il profilo dell’utente {ID} sotto forma dello stesso oggetto utilizzato come payload per l’inserimento (POST) tramite la URL auth/{SID}/utente, ad esclusione della password.

**URL**: http://auth/{SID}/utenti/{ID}

**VERBO**: PUT

**RISPOSTA**: Aggiorna il profilo dell’utente {ID}. Il payload è lo stesso oggetto utilizzato per l’inserimento tramite la URL auth/{SID}/utente. L’operazione viene eseguita nel contesto della sessione con token {SID} e solo se l’utente non ha effettuato l’accesso con Facebook.

**URL**: http://auth/{SID}/eventi

**VERBO**: POST

**RISPOSTA**: Inserisce un nuovo evento nel sistema. L’oggetto passato come payload conterrà tutti i dati dell’evento.

**URL**: http://auth/{SID}/eventi

**VERBO**: GET

**RISPOSTA**: Restituisce la lista degli eventi che verranno inseriti nella Home.

**URL**: http://auth/{SID}/eventi{ID}

**VERBO**: GET

**RISPOSTA**: Restituisce tutte le informazioni relative all’evento {ID}.

**URL**: http://auth/{SID}/eventi/{ID}

**VERBO**: PUT

**RISPOSTA**: Modifica dell’evento {ID}. Il payload è lo stesso oggetto utilizzato per l’inserimento tramite la URL applicazione/evento. L’operazione viene eseguita solo se l’utente ha mai creato un evento.

**URL**: http://auth/{SID}/eventi/{ID}/organizzatore

**VERBO**: GET

**RISPOSTA**: Restituisce le informazioni dell’utente che ha creato l’evento restituito dalla GET su eventi/{ID}. (Nella homepage, per ogni evento nella lista, mostreremo la foto del profilo, il nome e il cognome del creatore di quell’evento).

**URL**: http://auth/{SID}/eventi?{FILTRO}

Es. /eventi?tipo=cena, /eventi?prezzo=10, o anche composte come /eventi?prezzo=10&tipo=cena

**VERBO**: GET

**RISPOSTA**: Restituisce la lista delle URI che rappresentano gli eventi nel sistema filtrati tramite la query string {FILTRO}.

**URL**: http://auth/{SID}/utenti/{ID}/partecipazioni/{ID}/recensioni

**VERBO**: POST

**RISPOSTA**: Inserisce una nuova recensione nel sistema. L’oggetto passato come payload conterrà tutti i dati della recensione.

**URL**: http://auth/{SID}/utenti/{ID}/recensioni

**VERBO**: GET

**RISPOSTA**: Restituisce tutte le recensioni subite dall’utente {ID}.

**URL**: http://auth/{SID}/utenti/{ID}/recensioni/{ID}

**VERBO**: GET

**RISPOSTA**: Restituisce la recensione {ID} subita dall’utente {ID}.

**URL**: http://auth/{SID}/utenti/{ID}/messaggi

**VERBO**: POST

**RISPOSTA**: Inserisce un nuovo messaggio nel sistema. L’oggetto passato come payload conterrà tutti i dati del messaggio.

**URL**: http://auth/{SID}/eventi/{ID}/messaggi

**VERBO**: GET

**RISPOSTA**: Restituisce tutti i messaggi presenti nella bacheca dell’evento {ID}.

**URL**: http://auth/{SID}/utenti/{ID}/eventi

**VERBO**: GET

**RISPOSTA**: Restituisce la lista delle URI che rappresentano gli eventi nel sistema creati dall’utente {ID}.

**URL**: http://auth/{SID}/eventi/{ID}/partecipanti

**VERBO**: POST

**RISPOSTA**: Inserisce una nuova partecipazione ad un evento.

**URL**: http://auth/{SID}/eventi/{ID}/partecipanti

**VERBO**: GET

**RISPOSTA**: Restituisce la lista dei partecipanti a un evento.

**URL**: http://auth/{SID}/utenti/{ID}/partecipazioni

**VERBO**: GET

**RISPOSTA**: Restituisce la lista degli eventi a cui partecipa l’utente {ID}.

**Funzionalità realizzate lato client:**

-Login.

-Registrazione.

-Logout.

-Lista degli eventi futuri.

-Applicazioni filtri di tipo e prezzo agli eventi.

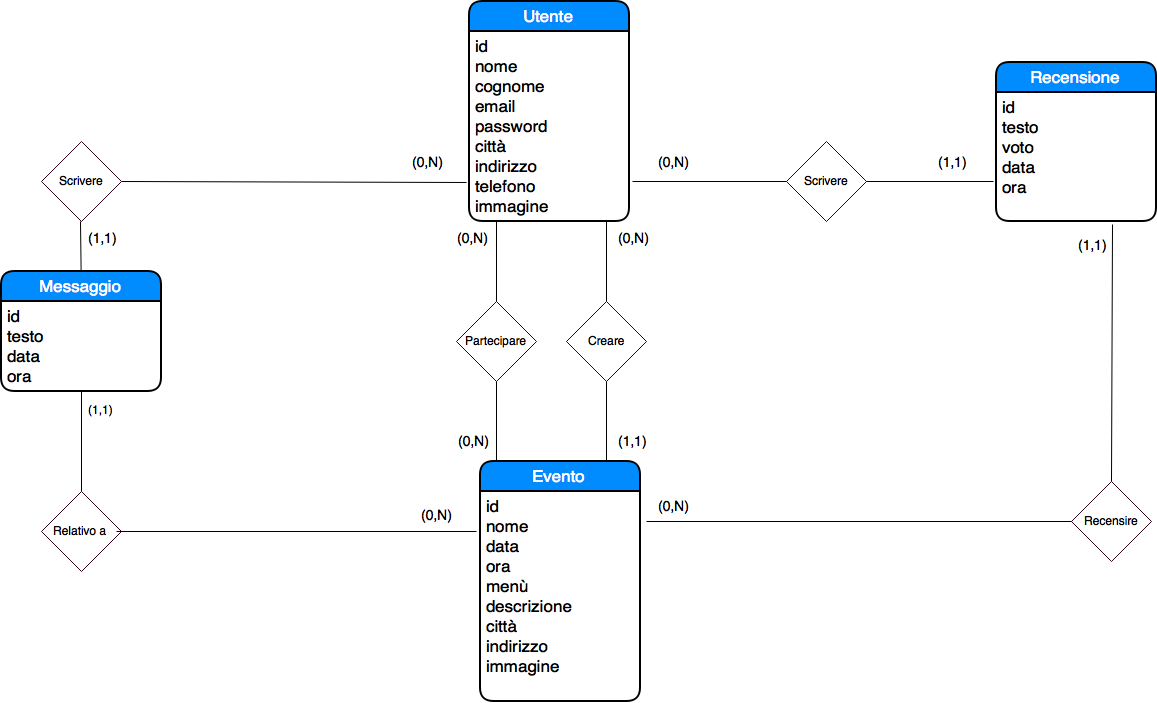
-Possibilità di partecipazione ad un evento.

-Lista di tutti i partecipanti ad un evento.

**Tecnologie/Librerie/Framework utilizzati lato client:** HTML5, CSS3, jQuery, FontAwesome.

**Tecnologie/Librerie/Framework utilizzati lato server:** JAX-RS, Jackson, MySQL.

**Diagramma ER:**

****