Relazione progetto
Database

Sito web personale

Cartalemi Simone
X81000684
Anno Accademico 2018/2019

Indice

Introduzione

Si vuole progettare un sistema informativo per la gestione di un blog personale attraverso un'interfaccia web, nel nostro caso su opere d'arte ed eventi legati al cliente che ne sta facendo richiesta.

Essendo un blog personale sarà il cliente, o un ristretto numero di utenti, ad utilizzare il sistema, che è progettato su pagine web in PHP e HTML. L'utente ha una minima competenza della gestione di infrastrutture informatiche, pertanto l'applicazione web non dovrà tener conto di alcune caratteristiche per utenti più esigenti. Ciò nonostante, il database è progettato per essere versatile e di facile manutenzione, soprattutto qualora l'utente decidesse di aggiungere o rimuovere caratteristiche significative dell'intera applicazione.

Esiste già la quasi totalità strutturale del sito web, come viene naturale immaginare manca tutta la parte correlata ai dati. Esiste un gruppo di requisiti minimi che il database deve possedere, ma saranno aggiunte caratteristiche utili al fine di mantenere efficiente l'intera applicazione.

In questa relazione si discuterà la progettazione, dal suo concepimento, del database che è necessario per ottenere i risultati desiderati del prodotto finale. Ovviamente si discuteranno eventuali scelte "anomale" e tutte le strategie applicate, al fine di avere una piena e assoluta conoscenza del funzionamento della suddetta base di dati.

Si partirà da una completa analisi della situazione sulla quale si sta operando, per poi via via costruire l'intero sistema, dalla carta alla macchina. Il sistema è testato e montato su server web in *localhost*, ma non si esclude il fatto che sarà presto online, che poi è la sua destinazione finale.

Specifiche sui dati

Come già annunciato, esiste già la struttura di base della nostra web app. Esistono diverse pagine nel quale il visitatore può liberamente scorrazzare, alcune di queste sono statiche, ma la maggior parte sono dinamiche e dovranno interfacciarsi con una base di dati.

Il cliente deve principalmente pubblicare **Articoli** sulle proprie opere e sugli eventi o mostre che lo riguardano, con una certa frequenza. Come viene facile pensare, ogni articolo deve contenere delle **Foto** che mostrino effettivamente di che cosa si sta parlando nel suddetto articolo. Inoltre, le opere di cui parlano gli articoli sono di varie **Categorie**, che variano dalla scultura ai gioielli.

Questo è quello che riguarda la primissima parte della base di dati, ovvero ciò che necessariamente deve contenere (in effetti è il motivo per il quale è nato il bisogno di avere una base per i nostri dati).

In un mondo che diventa sempre più digitale, chiunque, da qualunque parte del mondo, può navigare in rete e cercare qualsiasi cosa gli passi per la mente; nel nostro caso, chiunque può entrare all'interno del sito web, motivo per il quale quest'ultimo è disponibile in varie **Lingue**. Fatta quest'ultima considerazione diventa evidente il bisogno di rendere gli articoli disponibili in vari linguaggi, o almeno quelli di base: italiano e inglese.

Molto importante considerare l'aspetto della gestione interna del sito, pertanto, solo pochi **Utenti** possono creare, modificare ed eliminare articoli o altri dati presenti all'interno del sistema.

Il sito è provvisto di una sezione dedicata alla **Biografia**, che, in base alle esigenze del cliente, deve essere modificabile in qualsiasi momento e quindi diventare quasi un curriculum informale. Pur essendo ancora un giovane artista, sono in molti i giornali, libri e notiziari che hanno parlato di lui e ci auguriamo che saranno ancora in molti a parlarne in futuro. Anche la biografia dovrà essere scritta in più lingue. Da qui l'idea di raccogliere i **Riferimenti** in una lista e renderli disponibili in una parte della sezione di cui abbiamo discusso pocanzi.

Ultima considerazione da fare riguarda la presenza di una traccia delle **Visite** al sito web, ovvero registrare quanti visitatori accedono al sito durante il corso della giornata. Non sarà qualcosa di raffinato come Google Analytics, ma potrebbe tornare utile se disponibile immediatamente all'accesso del sito e magari fare qualche considerazione in più sull'andamento generale.

Queste sono le richieste del cliente, analizziamo e approfondiamo i concetti, quindi formalizziamo e costruiamo la nostra base di dati. Alcune scelte verranno fatte in base ad ulteriori esigenze del cliente che verranno descritte più avanti.

Analisi

Nella raccolta dei dati si individuano facilmente delle entità:

1) Articoli
2) Foto
3) Categorie
4) Lingue
5) Utenti
6) Biografia
7) Riferimenti
8) Visite

A questo punto costruiamo un *glossario dei termini*, molto utile per approfondire la struttura concettuale.

Termine	Descrizione	Sinonimo	Legame
Articoli	Tutte le informazioni sugli articoli, anche quelle opzionali	Opere	Foto, Categorie, Lingue
Foto	Nome della foto	Immagini	Articoli
Categorie	Nome della categoria		Articoli
Lingue	Per rendere il sito multilingua		Biografia, Articoli
Utenti	Informazioni sugli utenti che accedono e operano sui dati		
Biografia	Descrizione dell'omonima pagina	Curriculum	Lingue
Riferimenti	Collegamenti o citazioni dell'artista	Citazioni	
Visite	Tabella di traccia delle visite giornaliere		

Per il momento ci limitiamo a stabilire quali sono le entità del nostro database. Non sono state descritte in maniera accurata i requisiti e proprietà di cui queste entità hanno bisogno. Infatti, utilizzando una strategia di progettazione adeguata, esamineremo tali requisiti mentre costruiremo il modello concettuale, quindi il modello *ER*.

Modello Concettuale

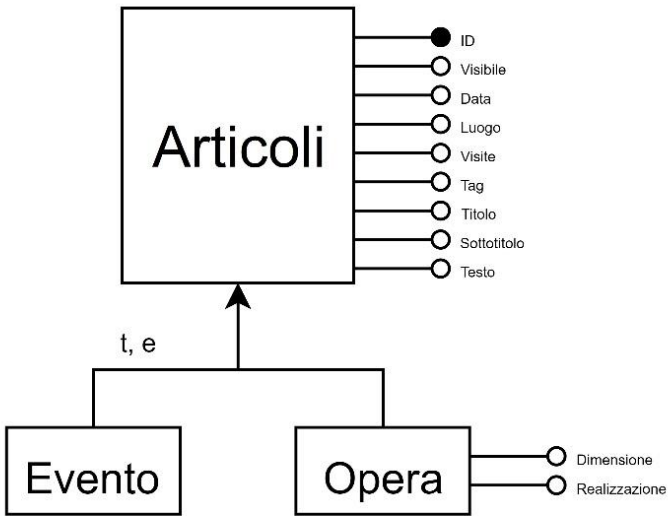
Costruiamo a poco a poco il nostro modello ER che rappresenterà il nostro schema più nel dettaglio. Utilizziamo una strategia **Bottom-Up**: introdurremo le entità, i loro attributi e le associazioni tra di esse, così da ottenere uno schema grafico.

Articoli

La prima entità che salta subito all'occhio (nonché concetto di base) è **Articoli**. Un articolo deve possedere le informazioni necessarie, oltre quelle richieste dal cliente. Nel nostro caso avremo:

- ID: chiave primaria dell'entità attraverso la quale, per una questione di eleganza e semplicità, distingueremo gli articoli tra di loro. Poiché è solo una comodità, andranno bene valori auto incrementanti;
- Titolo: frase di lunghezza massima pari a 250 caratteri, necessaria;
- Sottotitolo: frase di lunghezza massima di 250 caratteri, opzionale;
- Testo: qui sarà contenuto il testo in formato HTML (ovvero con tutti i tag necessari) perché l'applicazione è in grado di convertire il testo con tutti gli stili di formattazione disponibili (grassetto, corsivo, ecc.) e quindi è così che i testi saranno conservati. Deve anche poter conservare immagini nella codifica base64. Lunghezza massima di 16 milioni di caratteri;
- Visibilità: consente la creazione di bozze o nascondere articoli per revisionarli. Sarà un valore booleano e avrà il valore "1" se è visibile dagli utenti, "0" altrimenti e di default;
- Data: terrà le informazioni sulla data e sull'ora di creazione o pubblicazione dell'articolo. Obbligatorio ma modificabile;
- Luogo: dov'è possibile trovare l'opera o dove si tiene l'evento, pertanto è opzionale;
- Tag: è una stringa che contiene i tag per l'indicizzazione SEO. Ogni parola o frase sono separati da ", " perché la piattaforma li gestisce così. Al più 30 tag;
- Visite: un intero che per ogni articolo memorizza il numero di visitatori e ne fa una classifica;
- Dimensione: è una stringa che contiene le misure dell'opera (se materiale). Opzionale;
- Realizzazione: anno di realizzazione dell'opera, opzionale.

Si nota una netta distinzione tra ciò che è "opera" e ciò che è "evento", pertanto conviene distinguere le due cose creando una gerarchia. Tutti gli articoli saranno opere oppure eventi ed entrambe le categorie si escludono a vicenda. Facendo questa distinzione ci si accorge facilmente che la Dimensione e la Realizzazione hanno poco a che vedere con gli eventi, ragione per qui sembra più idoneo lasciare questi attributi alle opere piuttosto che a tutta la gerarchia. Il risultato che ne viene fuori è il seguente:



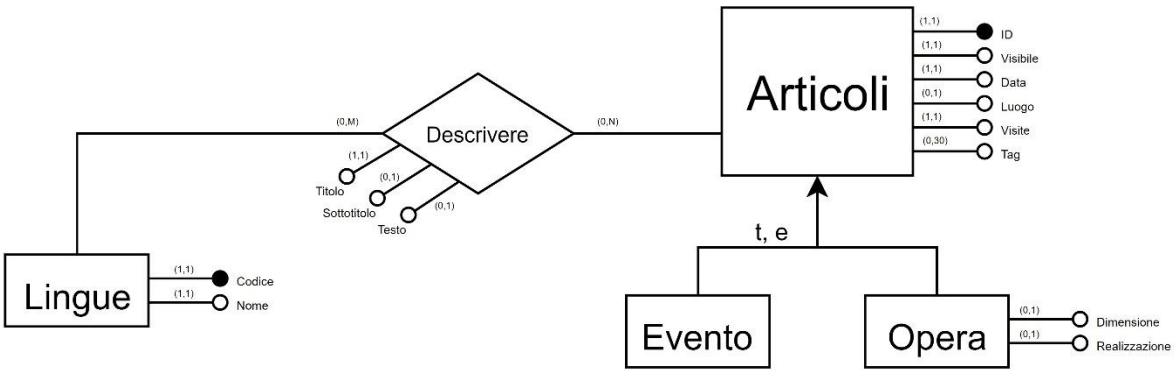
Lingue

Come precedentemente accennato, ogni articolo sarà descritto in una o più lingue, a loro volta ogni lingua può descrivere più articoli. Si evidenzia un'associazione molti a molti. L'entità **Lingue** avrà come attributi:

- Codice: che sarà la sigla della lingua in base allo standard ISO 639 (es: it) che tornerà utile nelle funzioni di gestione del sito e sarà formato da due caratteri e costituirà la chiave dell'entità.
- Nome: stringa che definisce il nome completo della lingua, ad esempio "Italiano". Massimo 20 caratteri;

Entrambi gli attributi sono univoci e obbligatori.

Nel creare l'associazione *Descrivere* ci si accorge che alcuni campi di Articoli sono per lo più correlati all'associazione: Titolo, Sottotitolo e Testo sono relativi al singolo articolo, tuttavia sono soggetti al cambio della lingua. Questo ci spinge a staccare questi attributi da Articoli per metterli all'associazione, creando così il seguente risultato:

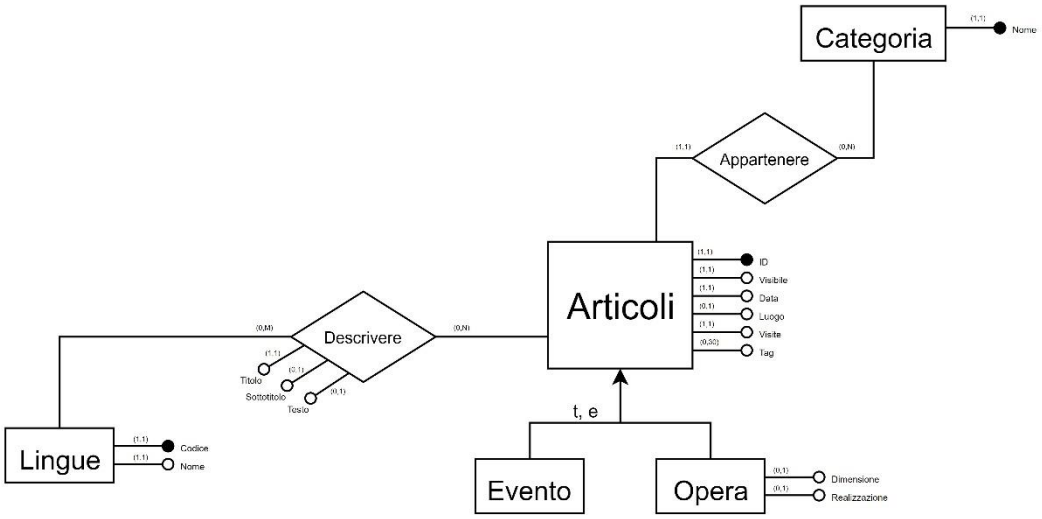


Categorie

Passiamo ora ad analizzare l'entità **Categorie**, che, molto semplicemente, dovrà possedere solo l'attributo che dovrà distinguere la classe alla quale appartengono gli articoli:

- Nome: stringa di massimo 30 caratteri, costituirà la chiave.

Questo perché sarà sempre unica e distinta e non ha bisogno di altre informazioni. Si potrebbe pensare di unificare la categoria agli articoli, ma si preferisce non fare questa unione poiché in futuro potrebbe verificarsi l'eventualità di creare o eliminare alcune classi di articoli. Ogni



Articolo appartiene ad una e una sola categoria, ad ogni Categoria appartengono più articoli: si presenta un'associazione una a molti e la figura seguente ne rappresenta le caratteristiche in forma grafica.

Foto

Ultimo stretto collegamento degli articoli riguarda le **Foto**, che non saranno realmente conservate all'interno del database, ma se ne terrà solo conservato solo il nome:

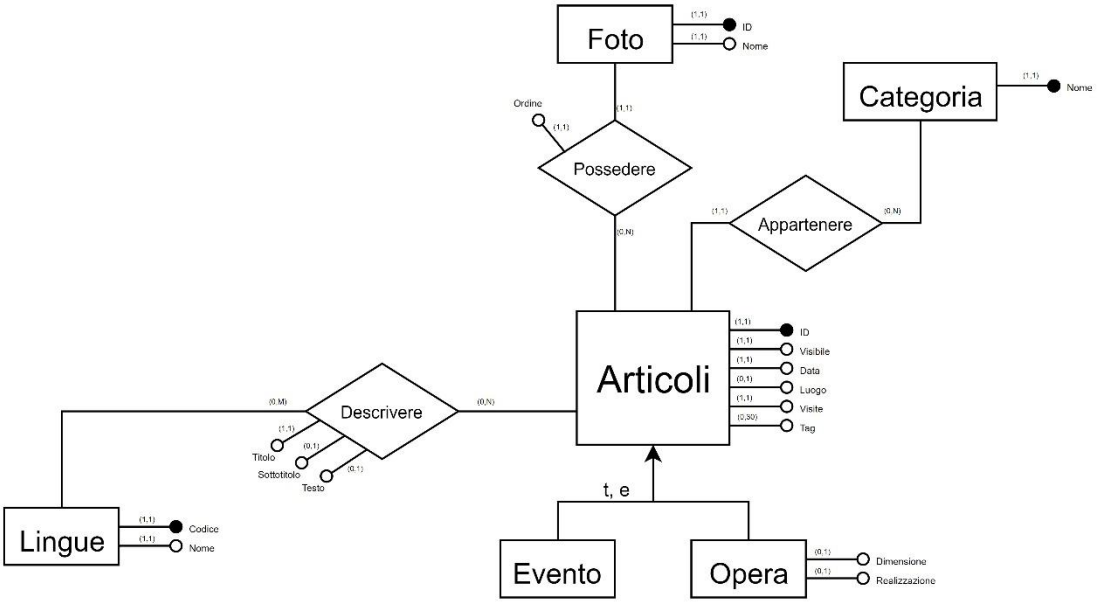
- ID: chiave primaria dell'entità che renderà molto più facili future operazioni che dovranno essere effettuate sulla web app;
- Nome: stringa della lunghezza massima di 300 caratteri.

Chiaramente sia l'ID che il Nome sono necessari e unici. Ogni Articolo possiede delle foto (opzionalmente) e ogni Foto deve necessariamente essere posseduta da un articolo, perché non esistono foto che non appartengono a nessun articolo. Si prevede che non possa verificarsi che una stessa immagine possa appartenere a due o più articoli contemporaneamente. Se dovesse verificarsi sarà molto più semplice caricarla più volte (uno per articolo), ma come già detto, questo è molto difficile che si riscontri.

L'associazione avrà un ulteriore attributo:

- Ordine: è un intero, obbligatorio, che registra l'ordine con il quale le foto dovranno apparire all'interno del sito web. Un caso particolare è la foto che avrà il valore 1, infatti questa sarà l'anteprima dell'articolo al quale è associata.

Collegando questa entità al tutto si ottiene il seguente schema:



In questo modo abbiamo completato il pezzo fondamentale del nostro puzzle.

Biografia

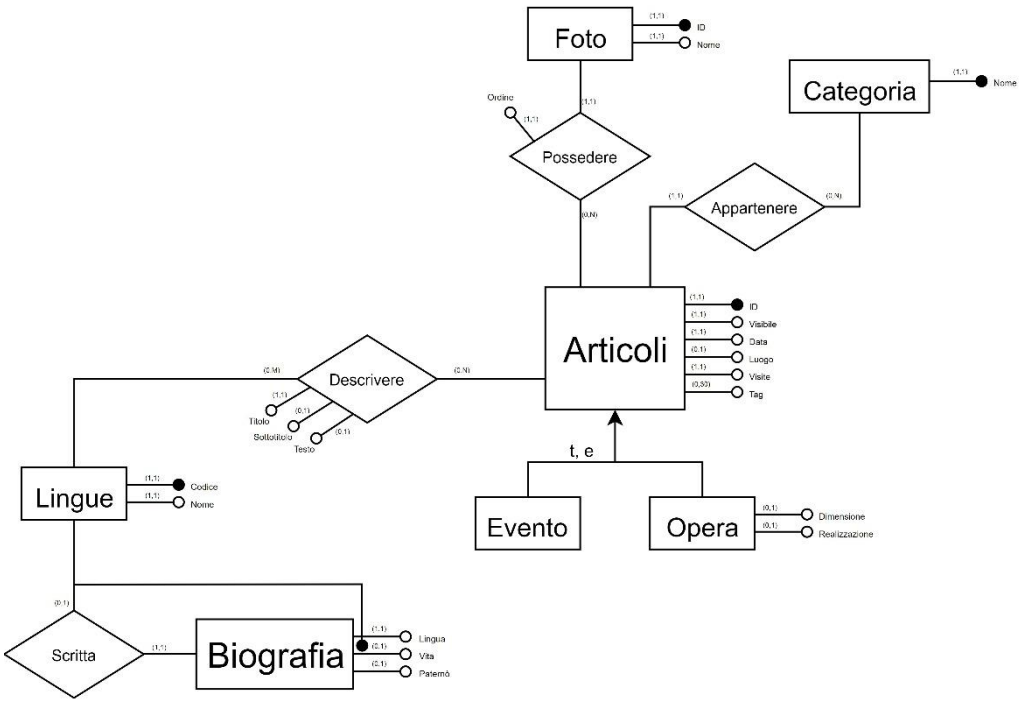
Un altro cardine del nostro prodotto è dato dalla **Biografia**, che sarà composta dai seguenti attributi:

- Lingua: chiave della relazione che coincide con la chiave dell'entità Lingue, identifica la particolare tupla e ne definisce la versione internazionale;

- Vita: testo di massimo 65 000 caratteri, sarà proprio la descrizione della vita dell'artista e alcuni tratti importanti della sua vita;
- Paternò: cenni storici e informazioni sulla città di Paternò, tematica che sta a cuore del richiedente. Anche in questo caso 65 000 caratteri sono più che sufficienti.

La sezione biografica non subirà numerosissime modifiche nel corso del tempo, ma ha senso inserirla all'interno del nostro schema poiché ha uno stretto rapporto con la lingua del nostro sistema, infatti ogni Biografia è scritta in una e una sola Lingua e, ovviamente, ogni Lingua può avere una Biografia.

Gli attributi di questa entità saranno dei testi, probabilmente anche abbastanza lunghi (ma opzionali, così da consentire delle bozze), proprio per questo motivo è necessario specificarli in un'altra entità piuttosto che fonderli con l'entità Lingue, in questo modo non dovremo scomodare questi campi durante le procedure che avranno a che fare con la lingua. Ecco che il nostro modello si presenta come segue.



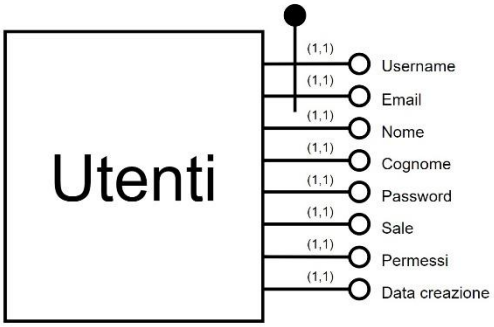
Quello descritto finora manca di un'importante caratteristica, ovvero la sicurezza del sito web stesso. Difatti deve esistere un muro tra utente che si limita solo a visitare il sito web e amministratore che crea, modifica ed eventualmente elimina tutte le informazioni contenute all'interno del database.

Utenti

Una tabella degli **Utenti** è la soluzione al nostro caso. Ogni istanza d'utenza deve avere le informazioni minime e necessarie per l'identificazione:

- E-mail: necessaria sin dalla registrazione dell'account, è anche utile per contattare l'utente in caso di necessità;
- Username: anch'esso utile ad indentificare univocamente l'utente, bastano 30 caratteri. Insieme ad e-mail costituiscono la chiave;
- Nome: identifica il nome dell'utente con un massimo di 30 caratteri;
- Cognome: identifica il cognome dell'utente con un massimo di 30 caratteri;

- Password: identifica la password utilizzata dall'utente nella fase di autenticazione. Il valore di tale attributo viene generato casualmente e memorizzato con un algoritmo di crittografia. Bastano 255 caratteri;
- Sale: generato insieme alla password. Combinati insieme consentono l'accesso, ma ne descriveremo meglio il senso più avanti. Anche qui 255 caratteri sono sufficienti.



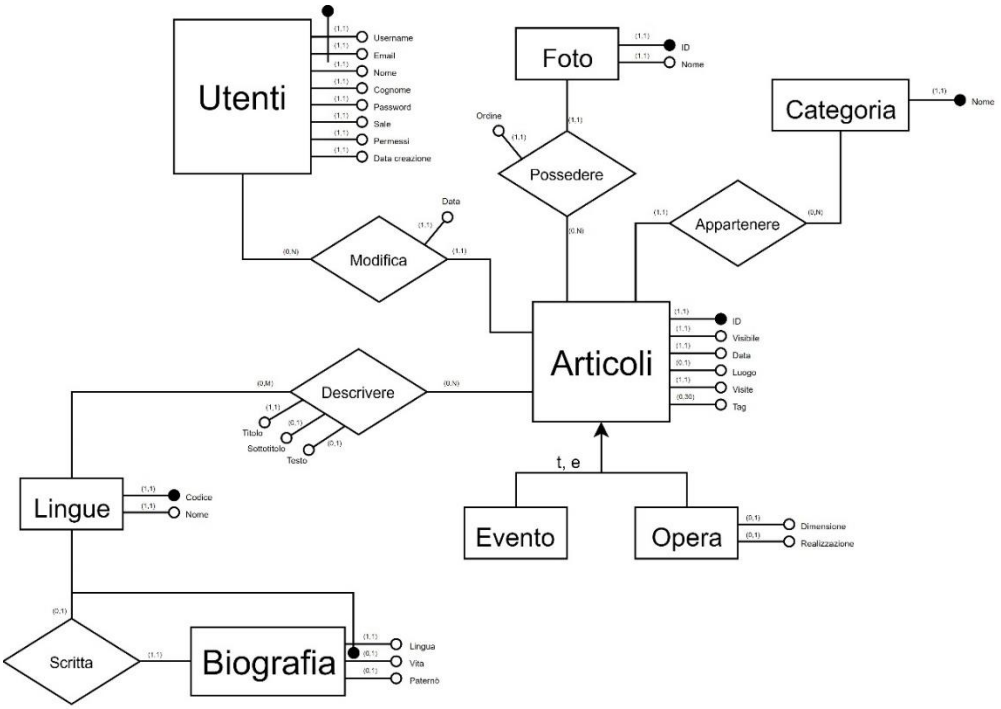
- Permessi: attributo che informa se consentire o no all'app delle operazioni sui dati e non solo, valore intero che va da 0 (nessun permesso consentito) a 7 (tutti i permessi, quindi amministratore). L'utente di livello 7 NON può essere eliminato o declassato in base ad una specifica richiesta del cliente;
- Data Creazione: data della registrazione dell'account.

Non sono necessarie altre informazioni allo scopo dell'applicazione, ma com'è facile capire ognuna di queste informazioni è indispensabile. Caso particolare e unico sarà un utente chiamato Admin e con accesso a pieni permessi, al fine di eseguire alcune operazioni di base del sito. Accanto la loro rappresentazione.

Sempre per ragioni di sicurezza, è necessario assicurarsi che possano essere solo gli utenti a eseguire le operazioni sui dati. Per fare ciò inseriamo un'associazione tra Utenti e Articoli, così da impedire modifiche se non si effettua prima l'accesso. Ovviamente non possiamo assicurarci che il nostro database non sarà mai violato, ma è un po' come rimuovere le chiavi dal cruscotto e chiudere la propria auto quando si lascia in un parcheggio. Quindi tra Utenti e Articoli introduciamo l'associazione Modifica: Ogni Utente può modificare gli articoli e ogni articolo deve necessariamente essere modificato da un solo utente alla volta. L'associazione conterrà:

- Data: attributo che si assicurerà di conservare il momento in cui verrà effettuata una modifica. Per evitare confusione specifichiamo che la modifica che registreremo nel database sarà solo l'ultima effettuata da un utente (o la creazione dell'articolo la prima volta).

Questo è lo schema che ne viene fuori:



Visite

Rimangono solo due entità da aggiungere allo schema, ma entrambe non sono connesse dal complesso costruito finora, inoltre sono distinte tra di loro.

La prima è l'entità **Visite**: questa entità si distingue dalle visite al singolo articolo poiché non è detto che chi visita il sito poi visiti almeno un articolo e per di più lo scopo di questa entità è molto differente da quello già esaminato prima. Gli attributi che ci servono sono:

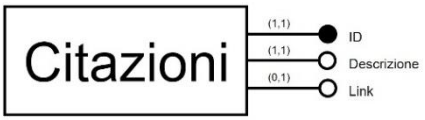
- Data: che, oltre a registrare esattamente la data, sarà la nostra chiave;
- Totale: intero che conta le visite per quel determinato giorno.

Entrambi sono obbligatori, ma non è detto che per ogni giorno dell'anno esisterà il record affiliato e questo a noi sta bene perché sarà compito della funzione in PHP a sapere cosa fare con questi dati.

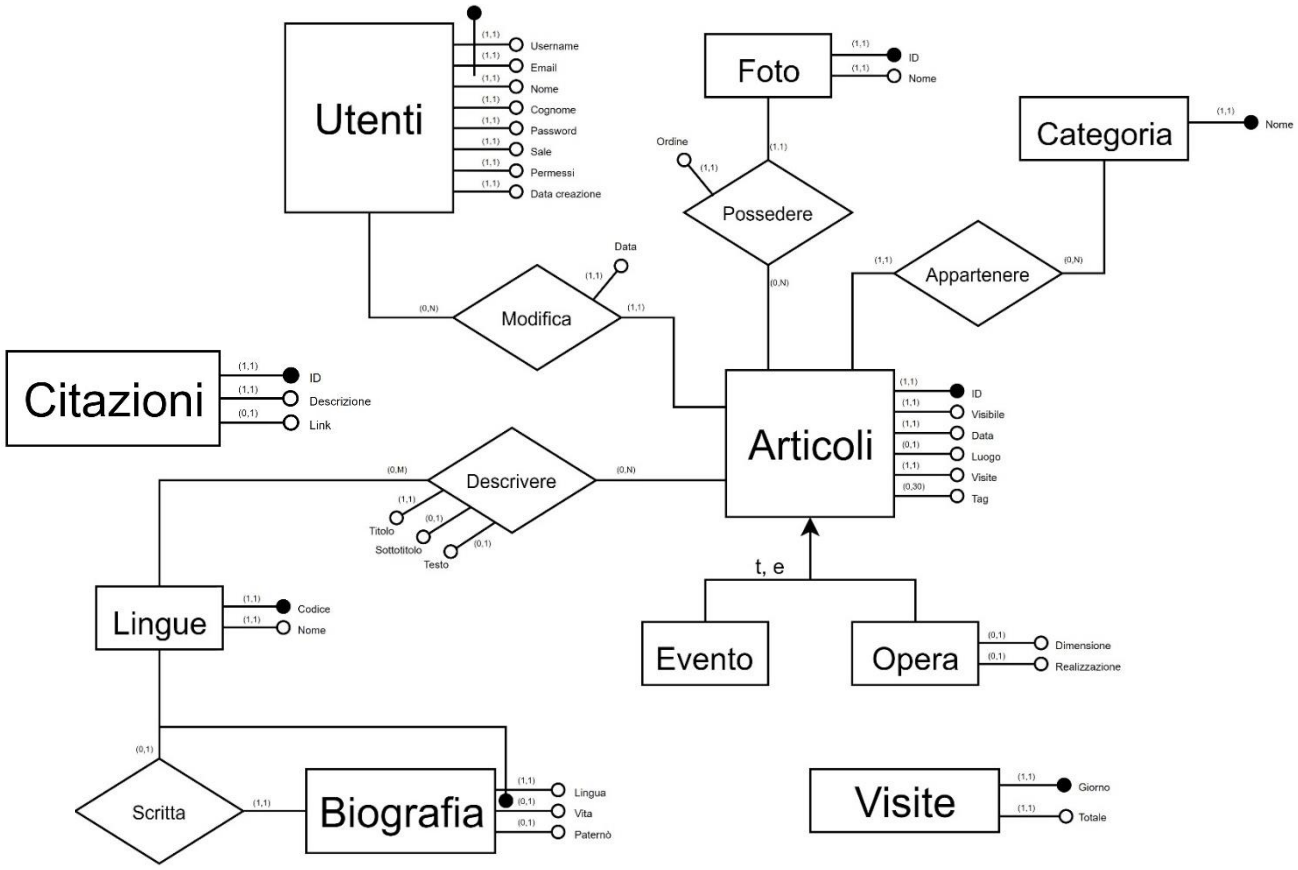
Citazioni

L'ultima entità riguarda le **Citazioni** (o riferimenti), che altro non è che una lista di libri, giornali o altri materiali online e non che parlano del nostro artista o di opere da lui create. Questi dati saranno visibili nella sezione biografica del sito e arricchiranno il curriculum informale. L'entità è così costituita:

- ID: sarà la chiave di ogni record;
- Descrizione: breve frase che esprimerà un riferimento;
- Link: opzionale. Se il materiale, o maggiori informazioni, sono disponibili online.



Abbiamo costruito il nostro modello ER con tutte le entità e le associazioni.



Progettazione Logica

Portata

Iniziamo il nostro cammino verso la traduzione dello schema concettuale in modello logico.

La prima cosa da fare è aggiungere una *tavola dei volumi*, così da avere una stima massima (un periodo di circa dieci anni) della mole di dati con il quale avremo a che fare.

Concetto	Tipo	Volume
Articoli	Entità	500
Foto	Entità	2000
Categorie	Entità	10
Lingue	Entità	10
Utenti	Entità	5
Biografia	Entità	10
Riferimenti	Entità	200
Visite	Entità	3650

Concetto	Tipo	Volume
Descrivere (articoli)	Relazione	5000
Appartenere (categoria)	Relazione	500
Possedere (foto)	Relazione	500
Modifica (articolo)	Relazione	500
Scritta (Biografia)	Relazione	10

Come è facile notare, ci sono entità e associazioni strettamente collegate tra di loro. Ora passiamo ad elencare le operazioni da effettuare su questo database, tenendo conto delle specifiche e della frequenza con il quale esse saranno richiamate.

Operazione	Frequenza media
Dato l'ID, restituire i dati relativi all'articolo corrispondente	400/giorno
Restituire gli articoli in ordine di visibilità e pubblicazione	50/mese
Data la categoria, restituire gli articoli visibili in ordine di pubblicazione	20/giorno
Dato l'ID, restituire le sezioni dell'articolo corrispondente	1000/giorno
Restituire gli articoli più letti che non sono eventi	1200/giorno
Restituire alcuni articoli visibili di una categoria*	100/giorno
Restituire alcuni articoli visibili che non sono eventi*	1800/giorno
Dato l'ID, restituire i dati relativi all'articolo precedente	1000/giorno
Dato l'ID, restituire i dati relativi all'articolo successivo	1000/giorno
Dato l'ID, Incrementare le visite dell'articolo corrispondente	100/giorno
Eventi passati	10/giorno
Eventi futuri	10/giorno
Restituire la lista degli articoli che hanno portato più visite in 7 giorni dalla loro data di pubblicazione	1/settimana
Restituire la classifica di categorie con media di visualizzazioni, articolo più letto e media delle foto rispetto agli articoli	1/settimana
Restituire la lista di articoli che possiedono più di 10 foto	1/settimana
Restituire la lista degli ultimi 10 articoli modificati	1/settimana
Restituire la lista di tutte le lingue e il totale degli articoli che ne possiedono una versione	1/settimana
Restituire la lista degli articoli modificati da un utente	1/settimana
Dati alcuni parametri, restituire tutti gli articoli che li rispettano	50/mese
Restituisci la lista delle lingue	1000/giorno
Crea articolo	20/anno

Crea contenuto di un articolo in una determinata lingua	40/anno
Modificare articolo	50/anno
Modificare contenuto di un articolo in una determinata lingua	100/anno
Elimina contenuto di un articolo in una determinata lingua	1/10 anni
Elimina articolo e tutto il suo contenuto	1/5 anni
Restituisci la lista delle categorie	2000/giorno
Restituisci il numero di articoli di ogni categoria	50/anno
Aggiungi categoria	1/3 anni
Elimina categoria	1/3 anni
Dato l'ID, restituire i dati relativi a una foto	500/giorno
Dato l'ID, restituire i dati relativi a tutte le foto dell'articolo corrispondente	1000/giorno
Restituire alcune foto di tutti gli articoli*	3/anno
Restituire alcune foto di tutti gli articoli visibili e che non sono eventi*	50/giorno
Dato l'ID, restituire il massimo attributo di ordinamento di foto dell'articolo corrispondente	50/anno
Aggiungere foto	200/anno
Dato l'ID, Rinominare la foto corrispondente	300/anno
Dato l'ID, spostare la foto corrispondente a sinistra, nell'ordinamento con tutte le altre dell'articolo a cui appartiene	1000/anno
Dato l'ID, spostare la foto corrispondente a destra, nell'ordinamento con tutte le altre dell'articolo a cui appartiene	400/anno
Elimina foto	1/mese
Ricerca utente tramite username	50/mese
Ricerca utente tramite e-mail	50/mese
Restituisci la lista degli utenti, ordinata rispetto ai permessi	10/anno
Inserire nuovo utente	1/5 anni
Modificare le informazioni utente	1/5 anni
Modificare password utente	1/10 anni
Modificare permessi di un utente	1/3 anni
Elimina utente	1/10 anni
Dato l'ID, restituire la citazione corrispondente	20/mese
Restituisci la lista delle citazioni	20/giorno
Dato il codice della lingua, restituire la biografia corrispondente	20/anno
Dato il codice della lingua, modificare la biografia corrispondente	20/anno
Creare citazione	20/anno
Modificare citazione	5/anno
Eliminare citazione	1/10 anni
Restituire le visite di oggi	5000/giorno
Restituire il totale delle visite	5000/giorno
Inserire visita	1/giorno
Incrementare visita di oggi	40/giorno
Restituire la lista delle visite di un determinato periodo	1/settimana

*con "alcuni" si intende un ristretto numero in maniera da migliorare la visualizzazione dei risultati

Analisi delle ridondanze

Controllando accuratamente il database non risultano esserci ridondanze.

Simulazione dell'aggiunta di una ridondanza

Supponiamo di voler aggiungere l'attributo "Numero di articoli" all'entità **Categorie**. Le operazioni coinvolte sarebbero le seguenti:

1) Restituisci il numero di articoli di ogni categoria	50/anno
2) Crea articolo	20/anno
3) Modificare articolo	50/anno
4) Elimina articolo e tutto il suo contenuto	1/5 anni

Esaminiamo il costo di queste operazioni, nel caso in cui mantenessimo questo dato.

1) 1L = 1 × 50/anno
2) (50L + 1L + 1S) = 53 × 20/anno
3) (50L + 1L + 1S) = 53 × 50/anno
4) (50L + 1L + 1S) = 53 × 1/5 anni

Esaminiamo ora il costo delle operazioni senza la ridondanza.

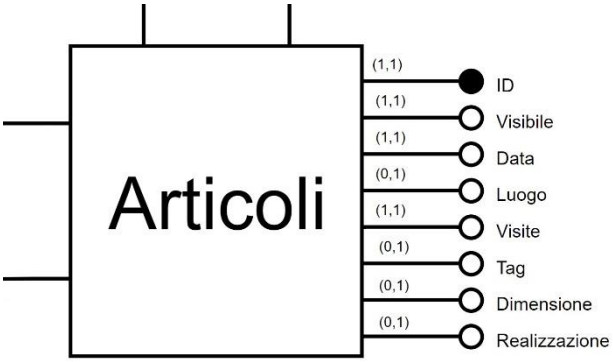
1) 50L = 50 × 50/anno
2) 0 × 20/anno
3) 0 × 50/anno
4) 0 × 1/ 5 anni

Risulta evidente come questa ridondanza convenga solo nella prima operazione, ma non nelle altre, dove aumenterebbe solo il costo di operazioni alla quale non sono interessati.

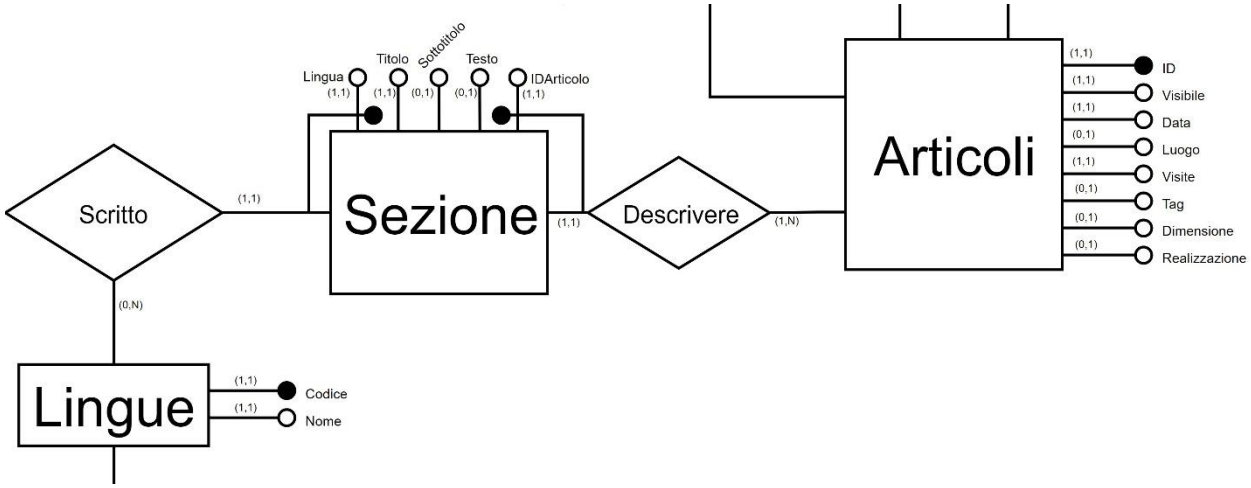
Ristrutturazione del modello ER

È arrivato il momento della ristrutturazione del database, visto che alcuni concetti astratti devono essere scomposti oppure combinati al fine di poterlo realizzare.

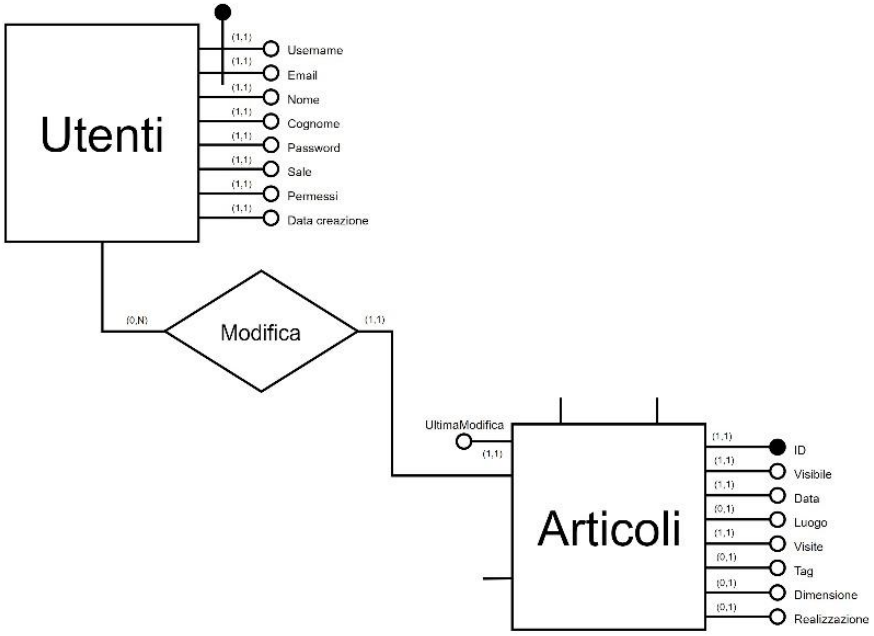
L'entità **Articoli** si può scomporre in *Evento* e *Opera*. Ci accorgiamo però che entrambe le categorie hanno la maggior parte delle caratteristiche in comune, infatti conviene collassare verso l'alto questa gerarchia. Così facendo non dovremo duplicare associazioni e attributi e la distinzione tra le varie gerarchie sarà gestita dalla tabella *Categoria*. Ristrutturando l'entità, questo è il risultato:



L'associazione **Descrivere** è di tipo molti a molti e deve quindi essere convertita in Entità. Chiameremo questa nuova tabella *Sezioni*, perché in effetti sarà la relazione che conterrà le sezioni degli articoli, ovviamente in lingua. Rimane tutto inalterato, quindi l'unica cosa da fare è importare su questa nuova entità le chiavi con le quali è messa in comunicazione.

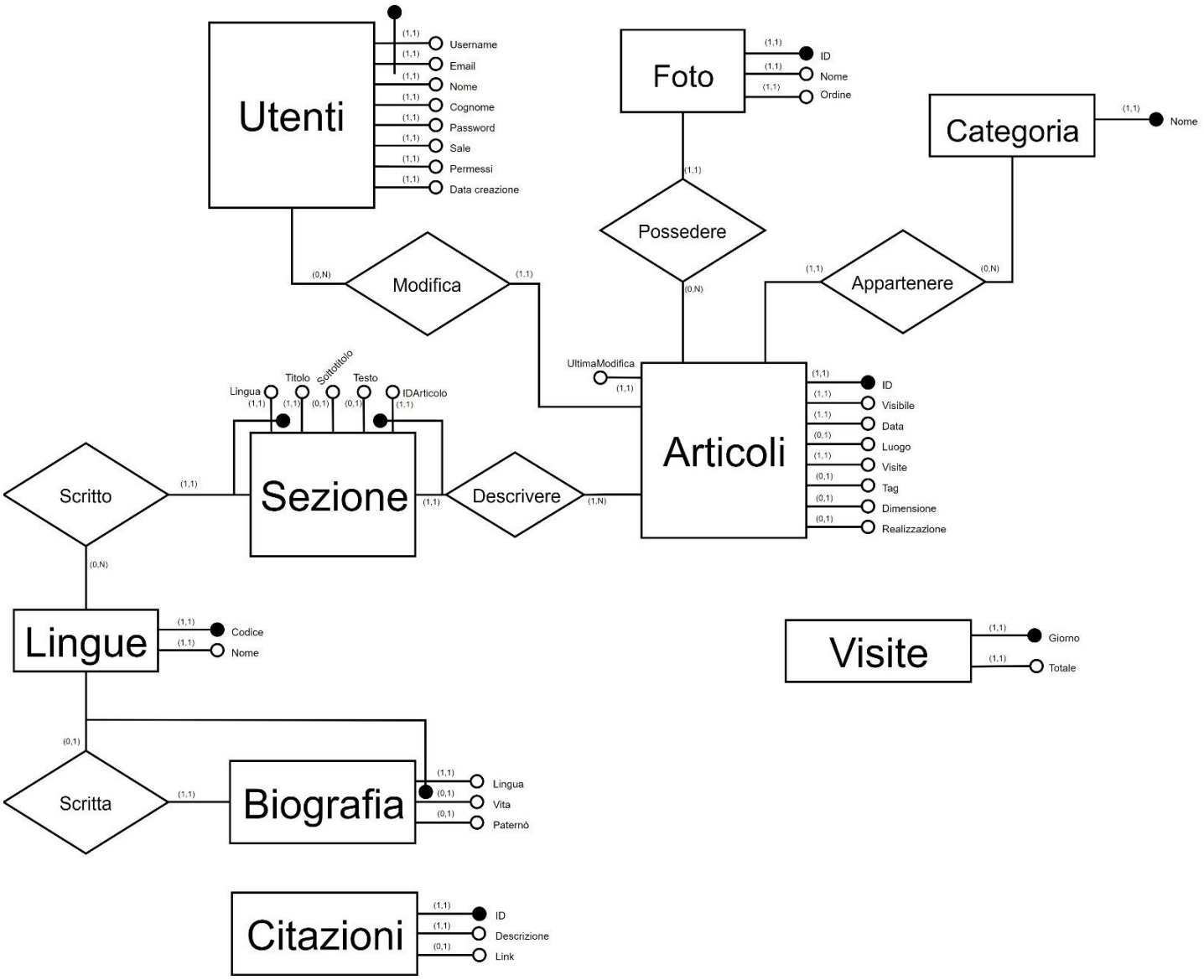


L'associazione **Modifica** tra *Articoli* e *Utenti* viene dislocata verso l'entità *Articoli*: qui si terrà traccia delle modifiche ad ogni singolo articolo. Rinomineremo il l'attributo *Data* in *Ultima Modifica*.



Analogamente, anche l'associazione **Possedere** tra *Articoli* e *Foto* subirà la medesima modifica: l'*Ordine* delle foto relativo all'articolo viene trasferito a Foto.

Ecco quindi il nostro schema ER ristrutturato:



Prendiamo in esame le singole relazioni e i loro attributi e ne deriviamo il modello logico (chiave primaria, chiave esterna, chiave primaria ed esterna):

Categoria: (Nome)

Articoli: (ID, Visibile, Data, Luogo, Visite, Tag, Dimensione, Realizzazione, UltimaModifica, Categoria, Utente)

Foto: (ID, Nome, Ordine, IDArticolo)

Lingue: (Codice, Nome)

Sezioni: (Lingua, IDArticolo, Titolo, Sottotitolo, Testo)

Utenti: (Username, Email, Nome, Cognome, Password, Sale, Permessi, DataCreazione)

Biografia: (Lingua, Vita, Paternò)

Citazioni: (ID, Descrizione, Link)

Visite: (Giorno, Totale)

Normalizzazione

La normalizzazione è un procedimento che ha lo scopo di eliminare la ridondanza ed eventuali incoerenze del database. Il processo di normalizzazione sottopone uno schema di relazione a una serie di test per certificare se soddisfa una data forma normale. Per verificare che il nostro modello logico sia normalizzato dobbiamo assicurarci che rispetti le forme normali.

Dipendenze funzionali

Dallo schema si evincono le seguenti DF:

Articoli:

ID → Visibile, Data, Luogo, Visite, Tag, Dimensione, Realizzazione, UltimaModifica, Categoria, Utente)

Foto:

ID → Nome, Ordine, IDArticolo

Lingue:

Codice → Nome

Sezioni:

Lingua, IDArticolo → Titolo, Sottotitolo, Testo

Utenti:

Username, Email → Nome, Cognome, Password, Sale, Permessi, DataCreazione

Biografia:

Lingua → Vita, Paternò

Citazioni:

ID → Descrizione, Link

Visite:

Giorno → Totale

Prima forma normale (1NF)

Richiede che il dominio di un attributo comprenda valori atomici e che il valore di un qualsiasi attributo in una tupla sia un valore singolo del dominio, ovvero:

- Tutte le righe della tabella contengono lo stesso numero di colonne;
- Gli attributi rappresentano informazioni elementari;
- I valori che compaiono in una colonna appartengono allo stesso dominio;
- Ogni riga è diversa da tutte le altre
- L'ordine con il quale le righe compaiono nella tabella è irrilevante.

La prima forma normale è già parte integrante della definizione formale di relazione nel modello logico, quindi il nostro schema rispetta la 1NF.

Seconda forma normale (2NF)

Uno schema relazionale è in seconda forma normale (2NF) quando è in prima forma normale (1NF) e tutti i suoi attributi non-chiave dipendono dall'intera chiave, cioè non possiede attributi che dipendono soltanto da una parte della chiave.

Esaminando la relazione **Articoli** si nota che essa risulta essere già in seconda forma normale (2NF); infatti essendoci un solo attributo chiave (ID) tutti gli altri attributi dipendono funzionalmente da quest'ultimo. Inoltre, non risultano esserci anomalie di inserimento, modifica e cancellazione: nel momento in cui una qualsiasi di questa operazione viene effettuata non si presentano anomalie di nessun tipo.

Le relazioni che presentano due attributi come chiave rispettano la 2NF, infatti se prendiamo (ad esempio) in esame la relazione **Utenti**, tutti gli attributi non-chiave non dipendono solo da *Username* o da *Email*, ma dalla coppia *Username - Email*.

Lo stesso ragionamento può essere fatto per tutte le altre relazioni dello schema senza incontrare anomalie o altro.

Terza forma normale (3NF)

Una base dati è in terza forma normale (3NF) se è in seconda forma normale e se tutti gli attributi non-chiave dipendono dalla chiave soltanto, ossia non esistono attributi non-chiave che dipendono da altri attributi non-chiave. Tale normalizzazione elimina la dipendenza transitiva degli attributi dalla chiave.

Sempre prendendo in esame la relazione **Articoli** si nota che essa risulta essere anche in terza forma normale, ma anche le altre relazioni rispettano la 3NF.

Forma normale di Boyce-Codd (BCNF)

Uno schema è in forma normale di Boyce e Codd se e solo se, per ogni dipendenza funzionale non banale $X \rightarrow Z$ definita su di esso, X è una superchiave della relazione.

Valutando ogni relazione ci si accorge che anche questo vincolo è rispettato.

In conclusione, il nostro schema è normalizzato secondo la BCNF. Abbiamo quindi il nostro modello relazionale completo, adesso ne esamineremo le caratteristiche nel dettaglio, pronti per avvicinarci al modello fisico.

Progettazione Fisica

Tabelle

In questa sezione rivedremo alcuni concetti importanti e porremo le basi per la creazione del database vero e proprio. Il DMBS scelto per questo base di dati è **MySQL**, poiché, essendo tra i più diffusi, è proprio quello che è installato sul server che ospiterà il sito web, ma anche in localhost. Rivediamo in una tabella riassuntiva le informazioni necessarie per la creazione delle tabelle e delle relazioni.

Tabella	Attributo	Chiave	Formato	Indicizzato	NULL
Categoria	Nome	PK	Testo (30)	Sì	No
Articoli	ID	PK	Intero autoincrementato	Sì	No
	Visibile		Booleano	No	No
	Data		Data e ora	No	No
	Luogo		Testo (255)	No	Sì
	Visite		Intero	No	No
	Tag		Testo (500)	No	Sì
	Dimensione		Testo (100)	No	Sì
	Realizzazione		Data	No	Sì
	Categoria	FK	Testo (30)	Sì	No
	UltimaModifica		Data e ora	No	No
	Utente	FK	Testo (30)	No	No
Foto	ID	PK	Intero autoincrementato	Sì	No
	Nome		Testo (100)	No	No
	Ordine		Intero	Sì	No
	IDArticolo	FK	Intero	Sì	No
Lingue	Codice	PK	Testo (2)	Sì	No
	Nome		Testo (20)	No	No
Sezioni	Lingua	FK, PK	Testo (2)	Sì	No
	IDArticolo	FK, PK	Intero	Sì	No
	Titolo		Testo (250)	No	No
	Sottotitolo		Testo (250)	No	Sì
	Testo		Testo (16 000 000)	No	Sì
Utenti	Username	PK	Testo (30)	Sì	No
	Email	PK	Testo (30)	Sì	No
	Nome		Testo (30)	No	No
	Cognome		Testo (30)	No	No
	Password		Testo (255)	No	No
	Sale		Testo (255)	No	No
	Permessi		Intero	No	No
	DataCreazione		Data e ora	No	No
Biografia	Lingua	FK, PK	Testo (2)	Sì	No
	Vita		Testo (65 000)	No	Sì
	Paternò		Testo (65 000)	No	Sì
Citazioni	ID	PK	Intero autoincrementato	Sì	No
	Descrizione		Testo (1000)	No	No
	Link		Testo (1000)	No	Sì
Visite	Giorno	PK	Data	Sì	No
	Totale		Intero	No	No

Grazie a questa tabella possiamo facilmente scrivere i comandi SQL per creare tale schema. Solo per conformità e maggiore leggibilità, il database sarà creato in lingua inglese e nella versione 5.0 di MySQL.

```
CREATE TABLE `category` (
  `Name` varchar(30) NOT NULL COMMENT 'Name of category' PRIMARY KEY
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
CREATE TABLE `users` (
  `Username` varchar(30) NOT NULL COMMENT 'User name',
  `Email` varchar(30) NOT NULL,
  `Name` varchar(30) NOT NULL,
  `Lastname` varchar(30) NOT NULL,
  `Password` tinytext NOT NULL,
  `Salt` tinytext NOT NULL COMMENT 'Salt of password',
  `Permission` int(11) NOT NULL DEFAULT '0' COMMENT 'permission of user',
  `Datecreate` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT 'Date of create account',
  PRIMARY KEY (`Username`,`Email`),
  UNIQUE KEY `Username` (`Username`),
  UNIQUE KEY `Email` (`Email`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
CREATE TABLE `articles` (
  `ID` int(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,
  `Category` varchar(30) NOT NULL COMMENT 'Category of article',
  `Visible` tinyint(1) NOT NULL DEFAULT '1' COMMENT 'Visibility of article',
  `Data` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT 'Date of the opera or the event',
  `Place` tinytext COMMENT 'Location',
  `Dimensions` varchar(100) DEFAULT NULL COMMENT 'Short description for the size of the opera',
  `Realization` year(4) DEFAULT NULL COMMENT 'Date of realization',
  `Tags` varchar(500) DEFAULT NULL,
  `Visits` int(11) NOT NULL DEFAULT '0' COMMENT 'Number of visits',
  `LastEdit` datetime on update CURRENT_TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `User` varchar(30) NOT NULL,
  KEY `Category` (`Category`),
  KEY `User` (`User`),
  CONSTRAINT `belong` FOREIGN KEY (`Category`) REFERENCES `category` (`Name`) ON DELETE RESTRICT ON UPDATE
CASCADE,
  CONSTRAINT `modify` FOREIGN KEY (`User`) REFERENCES `users` (`Username`) ON DELETE RESTRICT ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
CREATE TABLE `photos` (
  `ID` int(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,
  `Name` varchar(100) NOT NULL COMMENT 'Name or URL of the photo',
  `Sorting` int(11) NOT NULL COMMENT 'Order of the photos',
  `Article` int(11) NOT NULL,
  KEY `Article` (`Article`),
  KEY ` Sorting` (`Sorting`),
  CONSTRAINT `possess` FOREIGN KEY (`Article`) REFERENCES `articles` (`ID`) ON DELETE RESTRICT ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
CREATE TABLE `languages` (
  `Code` varchar(2) NOT NULL PRIMARY KEY COMMENT 'Code composed by 2 char',
  `Name` varchar(20) NOT NULL COMMENT 'Name of language'
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
CREATE TABLE `sections` (
  `Article` int(11) NOT NULL,
  `Language` varchar(2) NOT NULL,
  `Title` varchar(250) NOT NULL COMMENT 'Title of article in language',
  `Subtitle` varchar(250) COMMENT 'Optional subtitle of article',
  `Text` mediumtext NOT NULL COMMENT 'Text of article',
  PRIMARY KEY (`Language`,`Article`),
  KEY `Article` (`Article`),
  KEY `Language` (`Language`),
  CONSTRAINT `describe` FOREIGN KEY (`Article`) REFERENCES `articles` (`ID`) ON DELETE RESTRICT ON UPDATE
CASCADE,
  CONSTRAINT `written` FOREIGN KEY (`Language`) REFERENCES `languages` (`Code`) ON DELETE CASCADE ON UPDATE
CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
CREATE TABLE `biography` (
  `Language` varchar(2) NOT NULL PRIMARY KEY,
  `Life` text NOT NULL COMMENT 'life of Pier Manuel',
  `Paternò` text NOT NULL COMMENT 'hints on Paternò',
  KEY `Language` (`Language`),
  CONSTRAINT `drafted` FOREIGN KEY (`Language`) REFERENCES `languages` (`Code`) ON DELETE CASCADE ON UPDATE
CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
CREATE TABLE `citations` (
  `ID` int(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,
  `Description` varchar(1000) NOT NULL COMMENT 'description of reference',
  `LinkURL` varchar(1000) NULL COMMENT 'url to reference'
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
CREATE TABLE `visits` (
  `Day` date NOT NULL PRIMARY KEY,
  `Visits` int(11) NOT NULL COMMENT 'Number of visits'
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Vincoli

Prima di passare all'analisi delle funzioni per la quale il database è stato creato, andiamo ad analizzare alcuni vincoli necessari per mantenere integro il database.

Innanzi tutto, dobbiamo creare (nella maniera apposita e con le funzioni interne alla web app) l'utente fondamentale che permetterà l'inserimento e altre operazioni. L'utente sarà chiamato Admin, avrà i permessi massimi, mail del dominio del sito stesso e password (che per motivi ovvi non scriveremo in questa documentazione, ma è recuperabile).

```
INSERT INTO `users`
(`Username`, `Name`, `Lastname`, `Email`, `Password`, `Salt`, `Permission`, `Datecreate`)
VALUES
('Admin', 'Amministratore', 'sito', 'admin@piermanuel.it',
'[redacted]', '[redacted]', 7, '2018-10-06
02:34:00');
```

Creiamo un trigger che all'eliminazione di un utente si assicuri che questo non sia Admin e che non abbia i permessi massimi e, in tal caso, cambiare l'attributo *User* all'articolo modificato proprio da quell'utente. Bloccare l'operazione altrimenti.

```
CREATE TRIGGER `Deleting_User`
  BEFORE DELETE ON `users`
  FOR EACH ROW
  IF (OLD.`Permission` >= 7
  OR OLD.`Username` = 'Admin') THEN
    CALL cannot_delete_error;
  ELSE
    UPDATE `articles`
    SET `User` = 'Admin'
    WHERE `User` = OLD.`Username`;
  END IF;
```

Questo per far sì che l'articolo, la cui ultima modifica è stata effettuata da uno specifico utente, non venga eliminato insieme a quest'ultimo. Questa azione è necessaria perché ricordiamo che è obbligatorio avere un utente per aggiungere o modificare articoli. Inoltre, creiamo un trigger che impedisca la modifica dei permessi di Admin e username, così da impedirne l'eliminazione ed errori nel sistema.

```
CREATE TRIGGER `Admin_Privilegis`
  AFTER UPDATE ON `users`
  FOR EACH ROW
  IF (OLD.`Username` = 'Admin'
  AND (NEW.`Username` <> OLD.`Username`
  OR NEW.`Permission` <> OLD.`Permission`)) THEN
    UPDATE `users`
    SET NEW.`Username` = OLD.`Username`, NEW.`Permission` = OLD.`Permission`;
  END IF;
```

Infine, aggiungiamo un ulteriore trigger che creerà un record nella tabella Biografia non appena inseriamo una nuova lingua, così da trovarcelo già pronto nel momento in cui andremo ad aggiungere del testo. Questo ci facilita di molto il lavoro perché ci risparmia molto codice PHP.

```
CREATE TRIGGER `Add_Biography`
  AFTER INSERT ON `languages`
  FOR EACH ROW
  INSERT INTO `biography` (`Life`, `Paternò`, `Language`)
  VALUES ('New life', 'New Paternò', NEW.`Code`);
```

Adesso il nostro database è inviolabile e pronto ad essere popolato e interrogato, quindi elenchiamo tutte le interrogazioni necessarie contenute nelle varie classi PHP (essendo l'applicazione progettata ad oggetti). D'ora in poi tutte le variabili saranno precedute dal carattere '**$**', per mantenere compatibilità con il resto del codice. Si sottolinea che le query saranno richiamate dalle funzioni del PHP, noi in questa documentazione ci limitiamo ad agevolare la lettura e renderle facilmente leggibili.

Operazioni

La classe in questione si occuperà della gestione dei record delle tabelle Articoli, Categoria, lingua e sezioni. Ci serviranno due viste, che renderanno le query future più snelle.

Vista che visualizzi tutti gli articoli visibili ordinati per data, dalla più recente

```
CREATE VIEW `visible_articles` AS
  SELECT *
  FROM `articles`
  WHERE `Visible` = 1
  ORDER BY `Data` DESC;
```

Vista che tutte le anteprime

```
CREATE VIEW `previews` AS
SELECT `ID`, `Name`, `Article`
FROM `photos`
WHERE `Sorting` = 1;
```

Di seguito la lista di tutte le query con breve descrizione

Dato l'ID, restituire i dati relativi all'articolo corrispondente

```
SELECT *
FROM `articles`
WHERE `ID` = "$ID";
```

Restituire gli articoli in ordine di visibilità e pubblicazione

```
SELECT *
FROM `articles`
ORDER BY `Visible`, `Data` DESC;
```

Data la categoria, restituire gli articoli visibili in ordine di pubblicazione

```
SELECT *
FROM `visible_articles`
WHERE `Category` = "$category";
```

Dato l'ID, restituire le sezioni dell'articolo corrispondente

```
SELECT `Language`, `Title`, `Text`, `Subtitle`
FROM `sections`
WHERE `Article` = "$ID";
```

Restituire gli articoli più letti che non sono eventi

```
SELECT *
FROM `articles`
WHERE `Visible` = 1
AND `Category` != "Eventi"
ORDER BY `Visits` DESC, `Data` DESC
```

Restituire alcuni articoli visibili di una categoria

```
SELECT *
FROM `visible_articles`
WHERE `Category` = "$category"
LIMIT $limit
OFFSET $offset;
```

Restituire alcuni articoli visibili che non sono eventi

```
SELECT *
FROM `visible_articles`
WHERE `Category` != "Eventi"
LIMIT $limit
OFFSET $offset;
```

Dato l'ID, restituire i dati relativi all'articolo precedente

```
SELECT *
FROM `visible_articles`
WHERE `Data` < "$data"
AND `Category` = "$category"
LIMIT 1;
```

Dato l'ID, restituire i dati relativi all'articolo successivo

```
SELECT *
FROM `visible_articles`
WHERE `Data` > "$data"
AND `Category` = "$category"
LIMIT 1;
```

Dato l'ID, Incrementare le visite dell'articolo corrispondente

```
UPDATE `articles`
SET `Visits` = `Visits` + 1
WHERE `articles`.`ID` = "$ID";
```

Eventi passati

```
SELECT *
FROM `visible_articles`
WHERE `Category` = "Eventi"
AND `Data` < CURRENT_TIME();
```

Eventi futuri

```
SELECT *
FROM `visible_articles`
WHERE `Category` = "Eventi"
AND `Data` > CURRENT_TIME();
```

Restituire la lista degli articoli che hanno portato più visite in 7 giorni dalla loro data di pubblicazione

```
SELECT `a`.`ID`, SUM(pc.`Visits`) AS Visite
FROM `articles` AS `a`, (SELECT `ID`, `visits`.* FROM `visits`, `articles`) AS pc
WHERE `a`.`ID` = pc.`ID`
AND pc.`Day` BETWEEN `a`.`Data` AND `a`.`Data` + INTERVAL 7 DAY
GROUP BY `a`.`ID`
ORDER BY Visite DESC;
```

Restituire la classifica di categorie con media di visualizzazioni, articolo più letto e media delle foto rispetto agli articoli

```
SELECT `a`.`Category`, ROUND(AVG(Foto), 2) AS Foto, ROUND(AVG(`a`.`Visits`), 2) AS Visite, COUNT(DISTINCT
T1.`ID`) AS Articoli, `a1`.`ID`
FROM `articles` AS `a` LEFT JOIN (
  SELECT `Category`, `articles`.`ID`, COUNT(*) AS Foto
  FROM `articles` LEFT JOIN `photos` ON `articles`.`ID` = `Article`
  GROUP BY `Category`, `articles`.`ID`
) AS T1 ON `a`.`Category` = T1.`Category` JOIN `articles` AS `a1` ON `a`.`Category` = `a1`.`Category`
WHERE `a1`.`Visits` >=ALL(
  SELECT `Visits`
  FROM `articles`
  WHERE `a`.`Category` = `Category`
)
GROUP BY `Category`
ORDER BY Visite DESC;
```

Restituire la lista di articoli che possiedono più di 10 foto

```
SELECT `Article`, COUNT(*) AS Foto
FROM `photos`
GROUP BY `Article`
HAVING Foto > 10
ORDER BY Foto DESC;
```

Restituire la lista degli ultimi 10 articoli modificati

```
SELECT *
FROM articles JOIN `sections` ON `ID` = `Article`
ORDER BY `LastEdit` DESC
LIMIT 10;
```

Restituire la lista di tutte le lingue e il totale degli articoli che ne possiedono una versione

```
SELECT `l`.`Name`, IFNULL(Totale, 0) AS Totale
FROM `languages` AS `l` LEFT JOIN (
  SELECT `Name`, COUNT(`s`.`Title`) AS Totale
  FROM (`languages` LEFT JOIN `sections` AS `s` ON `Code` = `s`.`Language`) LEFT JOIN `articles` AS `a` ON
`s`.`Article` = `a`.`ID`
  WHERE `Visible` = 1
  GROUP BY `Name`
  ORDER BY Totale) AS T1 ON `l`.`Name` = T1.`Name`;
```

Restituire la lista degli articoli modificati da un utente

```
SELECT *
FROM `articles`
WHERE `User` = "$user"
ORDER BY `LastEdit` DESC
LIMIT 10;
```

Dati alcuni parametri, restituisci tutti gli articoli che li rispettano

```
SELECT `a`.*
FROM (`articles` AS `a` LEFT JOIN `sections` AS `s` ON `a`.`ID` = `s`.`Article`) LEFT JOIN `photos` AS `p` ON
`a`.`ID` = `p`.`Article`
WHERE (
  CONVERT(`s`.`Title` USING utf8) LIKE "%$searching%"
  OR CONVERT(`s`.`Subtitle` USING utf8) LIKE "%$searching%"
  OR CONVERT(`a`.`Place` USING utf8) LIKE "%$searching%"
  OR CONVERT(`a`.`Dimensions` USING utf8) LIKE "%$searching%"
  OR CONVERT(`a`.`ID` USING utf8) LIKE "%$searching%"
  OR CONVERT(`a`.`Data` USING utf8) LIKE "%$searching%"
  OR CONVERT(`a`.`Realization` USING utf8) LIKE "%$searching%"
  OR CONVERT(`a`.`Tags` USING utf8) LIKE "%$searching%"
  OR CONVERT(`s`.`Text` USING utf8) LIKE "%$searching%"
  OR CONVERT(`s`.`Title` USING utf8) LIKE "%$searching%"
  OR CONVERT(`p`.`Name` USING utf8) LIKE "%$searching%"
) AND `a`.`Category` = "$category"
AND `a`.`User` = "$user"
AND `a`.`ID` [NOT] IN (
  SELECT `Article`
  FROM `sections`
  WHERE `Language` = "$language"
)
GROUP BY `a`.`ID`
ORDER BY `a`.`Visible`, `a`.`$sort` [DESC];
```

Restituisci la lista delle lingue

```
SELECT *
FROM `languages`;
```

Crea articolo

```
INSERT INTO `articles`
(`Category`, `Visible`, `Data`, `Place`, `Dimensions`, `Realization`, `Tags`, `User`)
VALUES
("$category", $visible, $data, "$place", "$dimension", $realization, "$tags", "$user");
```

Crea contenuto di un articolo in una determinata lingua

```
INSERT INTO `sections`
(`Language`, `Title`, `Subtitle`, `Text`, `Article`)
VALUES
("$language", "$title", "$subtitle", "$text", $IDArticle);
```

Modificare articolo

```
UPDATE `articles`
SET `Category` = "$category", `Visible` = $visible, `Data` = "$dataOra", `Place` = "$place", `Dimensions` =
"$dimension", `Realization` = $realization, `Tags` = "$tags", `User` = "$user", LastEdit = CURRENT_TIME-
STAMP()
WHERE `ID` = $IDArticle;
```

Modificare contenuto di un articolo in una determinata lingua

```
UPDATE `sections`
SET `Title` = "$title", `Subtitle` = "$subtitle", `Text` = "$text"
WHERE `Article` = $IDArticle
AND `Language` = "$language";
```

Elimina contenuto di un articolo in una determinata lingua

```
DELETE FROM `sections`
WHERE `Article` = $ID
AND `Language` = "$language";
```

Elimina articolo e tutto il suo contenuto

```
START TRANSACTION;
  DELETE FROM `photos` WHERE `Article` = $ID;
  DELETE FROM `sections` WHERE `Article` = $ID;
  DELETE FROM `articles` WHERE `ID` = $ID;
COMMIT;
```

Restituisci la lista delle categorie

```
SELECT `Name`
FROM `category`;
```

Aggiungi categoria

```
INSERT INTO `category`
(`Name`)
VALUES
("$newCategory");
```

Elimina categoria

```
DELETE FROM `category`
WHERE `Name` = "$category";
```

Dato l'ID, restituire i dati relativi a una foto

```
SELECT *
FROM `photos`
WHERE `ID` = $ID;
```

Dato l'ID, restituire i dati relativi a tutte le foto dell'articolo corrispondente

```
SELECT *
FROM `photos`
WHERE `Article` = $IDArticle
ORDER BY `Sorting` ASC;
```

Restituire alcune foto di tutti gli articoli

```
SELECT *
FROM `photos`
ORDER BY `Article` DESC, `Sorting` ASC
LIMIT $limit OFFSET $offset;
```

Restituire alcune foto di tutti gli articoli visibili e che non sono eventi

```
SELECT *
FROM `previews` AS `p` JOIN `visible_articles` AS `a` ON `a`.`ID` = `p`.`Article`
WHERE `Category` != "Eventi"
LIMIT $limit OFFSET $offset;
```

Dato l'ID, restituire il massimo attributo di ordinamento di foto dell'articolo corrispondente

```
SELECT MAX(`Sorting`) AS `max`
FROM `photos`
WHERE `Article` = $IDArticle;
```

Aggiungere foto

```
INSERT INTO `photos`
(`Name`, `Sorting`, `Article`)
VALUES
("$name", $sorting, $IDArticle);
```

Dato l'ID, rinominare la foto corrispondente

```
UPDATE `photos`
SET `Name` = "$newName"
WHERE `ID` = $ID;
```

Dato l'ID, spostare la foto corrispondente a sinistra, nell'ordinamento con tutte le altre dell'articolo a cui appartiene

```
SELECT *
FROM `photos`
WHERE `Sorting` = $Sorting - 1
AND `Article` = $Article;

//successivamente applicare alle due foto

UPDATE `photos`
SET `Sorting` = $newsort
WHERE `ID` = $ID;
```

Dato l'ID, spostare la foto corrispondente a destra, nell'ordinamento con tutte le altre dell'articolo a cui appartiene

```
SELECT *
FROM `photos`
WHERE `Sorting` = $Sorting + 1
AND `Article` = $Article;

//ad entrambe le foto

UPDATE `photos`
SET `Sorting` = $newsort
WHERE `ID` = $ID;
```

Elimina foto

```
//da richiamare con l'istruzione
SET @ID = $ID;
CALL `decrement_sort`(@ID);

DELIMITER $$
CREATE PROCEDURE decrement_sort(IN `IDDeleting` INT)
  BEGIN
    DECLARE `ArticleID` INT;
    DECLARE `Sort` INT;
    SELECT `Article` INTO `ArticleID`
    FROM `photos`
    WHERE `ID` = `IDDeleting`;
    SELECT `Sorting` INTO `Sort`
    FROM `photos`
    WHERE `ID` = `IDDeleting`;
    DELETE FROM `photos`
    WHERE `ID` = `IDDeleting`
    AND `Article` = `ArticleID`;
    UPDATE `photos`
    SET `Sorting` = `Sorting` - 1
    WHERE `Article` = `ArticleID`
    AND `Sorting` > `Sort`;
  END$$
DELIMITER ;
```

Ricerca utente tramite username

```
SELECT *
FROM `users`
WHERE `Username` = "$username";
```

Ricerca utente tramite e-mail

```
SELECT *
FROM `users`
WHERE `Email` = "$email";
```

Restituisci la lista degli utenti, ordinata rispetto ai permessi

```
SELECT *
FROM `users`
ORDER BY `Permission` DESC;
```

Inserisci nuovo utente

```
INSERT INTO `users`
(`Username`, `Email`, `Name`, `Lastname`, `Password`, `Salt`, `Permission`)
VALUES
("$username", "$email", "$name", "$lastname", "$hash", "$salt", 0);
```

Modifica le informazioni utente

```
UPDATE `users`
SET `Name` = "$name", `Lastname` = "$lastname", `Username` = "$username", `Email` = "$email"
WHERE `Username` = "$username";
```

Modificare password utente

```
UPDATE `users`
SET `Password` = "$hash", `Salt` = "$salt"
WHERE `Username` = "$username";
```

Modificare permessi utente

```
UPDATE `users`
SET `Permission` = $permission + 1
WHERE `Username` = "$username";
```

Elimina utente

```
DELETE FROM `users`
WHERE `Username` = "$username";
```

Dato l'ID, restituire la citazione corrispondente

```
SELECT *
FROM `citations`
WHERE `ID` = $ID;
```

Restituisci la lista delle citazioni

```
SELECT *
FROM `citations`
ORDER BY `ID` DESC;
```

Dato il codice della lingua, restituire la biografia corrispondente

```
SELECT *
FROM `biography`
WHERE `Language` = "$language";
```

Dato il codice della lingua, modificare la biografia corrispondente

```
UPDATE `biography`
SET `Life` = "$bio", `Paternò` = "$paterno"
WHERE `Language` = "$language";
```

Creare citazione

```
INSERT INTO `citations`
(`Description`, `LinkURL`)
VALUES
("$description", "$url");
```

Modificare citazione

```
UPDATE `citations`
SET `Description` = "$description", `LinkURL` = "$url"
WHERE `ID` = $ID;
```

Eliminare citazione

```
DELETE FROM `citations`
WHERE `ID` = $ID;
```

Restituire le visite di oggi

```
SELECT *
FROM `visits`
WHERE `Day` = CURRENT_DATE;
```

Restituire il totale delle visite

```
SELECT SUM(`Visits`) AS `total`
FROM `visits`;
```

Inserire visita

```
INSERT INTO `visits`
(`Day`, `Visits`)
VALUES
(CURRENT_DATE, 1);
```

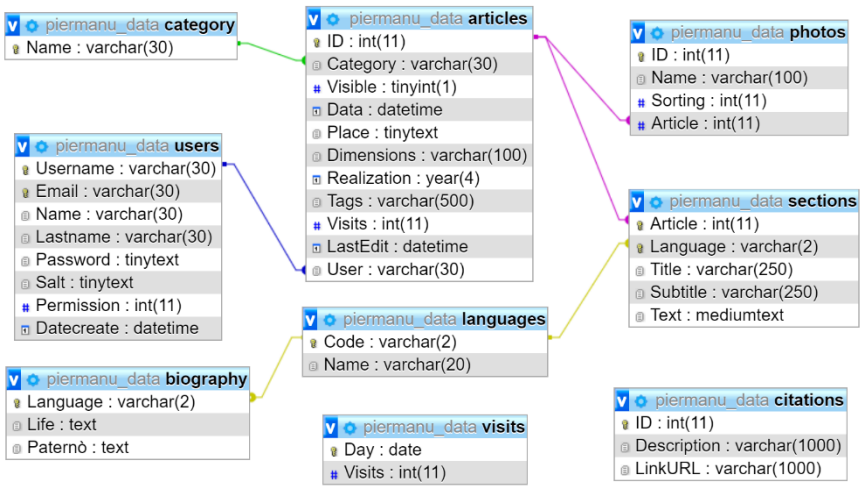
Incrementare visita di oggi

```
UPDATE `visits`
SET `Visits` = `Visits` + 1
WHERE `Day` = CURRENT_DATE;
```

Restituire la lista delle visite di un determinato periodo

```
SELECT *
FROM `visits`
WHERE `Day` BETWEEN "$dal" AND "$al"
ORDER BY `Day` ASC;
```

Dopo aver implementato lo schema, le procedure, i trigger e le viste possiamo ottenere lo schema grafico definitivo.



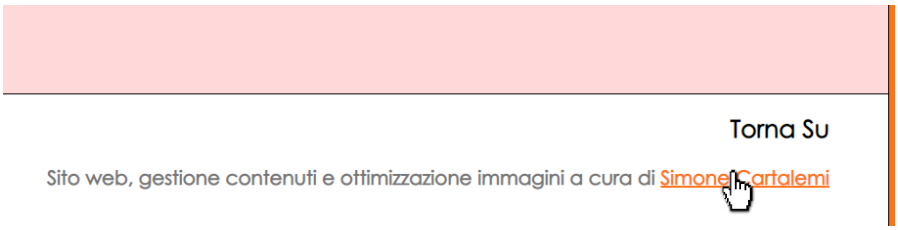
Guida per l'utente

Per chiudere questa documentazione mostreremo brevemente le funzionalità applicate alla web app descrivendo i passaggi.

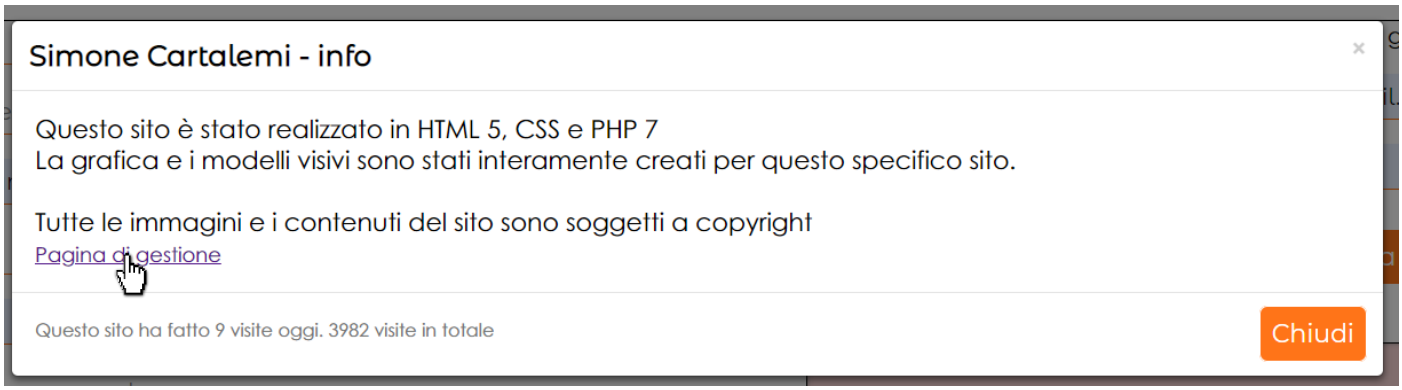
La pagina principale del sito è http://www.piermanuelcartalemi.it e una volta aperta è possibile consultare gli articoli e tutte le informazioni sull'artista in maniera abbastanza intuitiva.

Login

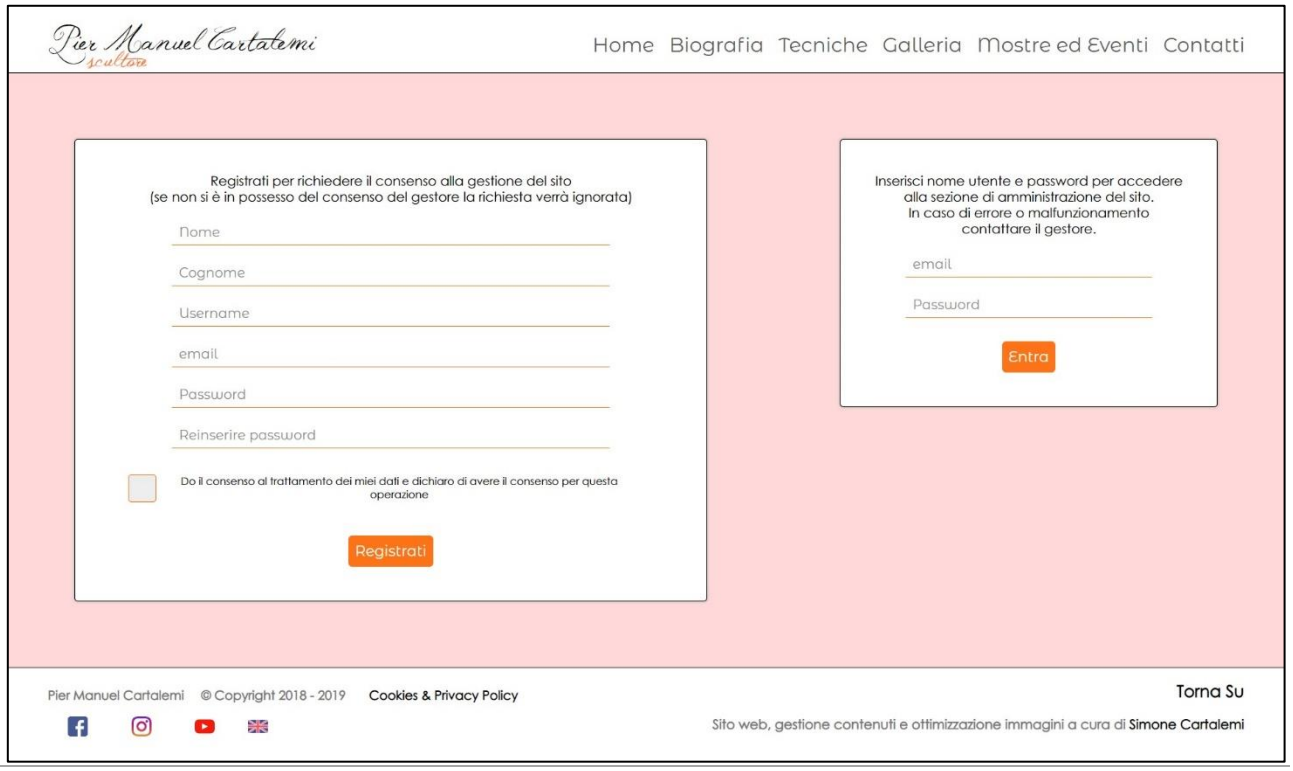
Per gestire il sito è necessario effettuare il login. Da qualsiasi pagina scendere in basso nel footer e cliccare sul link come nell'immagine seguente.



Si aprirà una finestra pop-up e apparirà il link per essere indirizzati alla pagina di login.

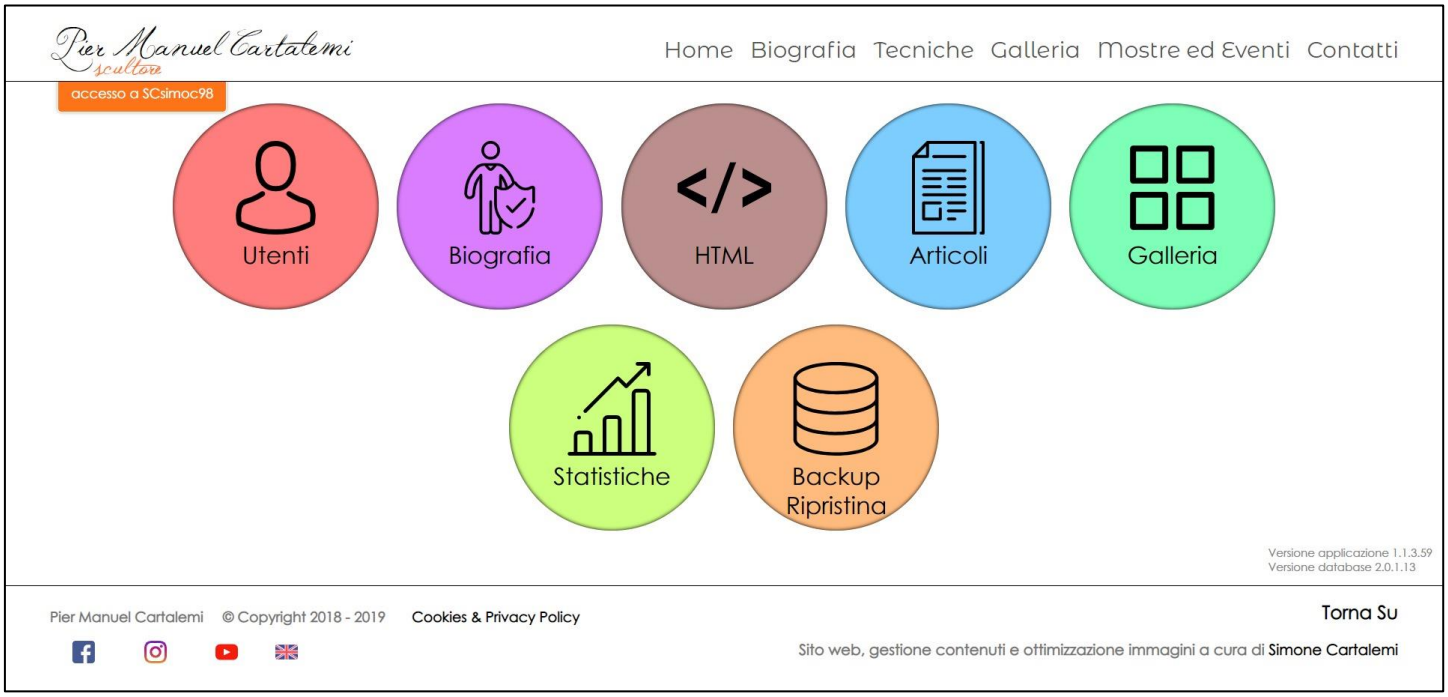


A questo punto arriverete alla pagina di login che dovrebbe avere più o meno questo aspetto.



Come si nota facilmente, a sinistra è presente un link per iscriversi al sito. Se è la prima volta che si tenta di entrare nel sito bisogna utilizzare le credenziali di Admin, così da avere il pieno controllo della piattaforma. Utilizzare il form di destra per loggarsi.

Una volta loggati dovrebbe apparire una schermata molto simile a questa

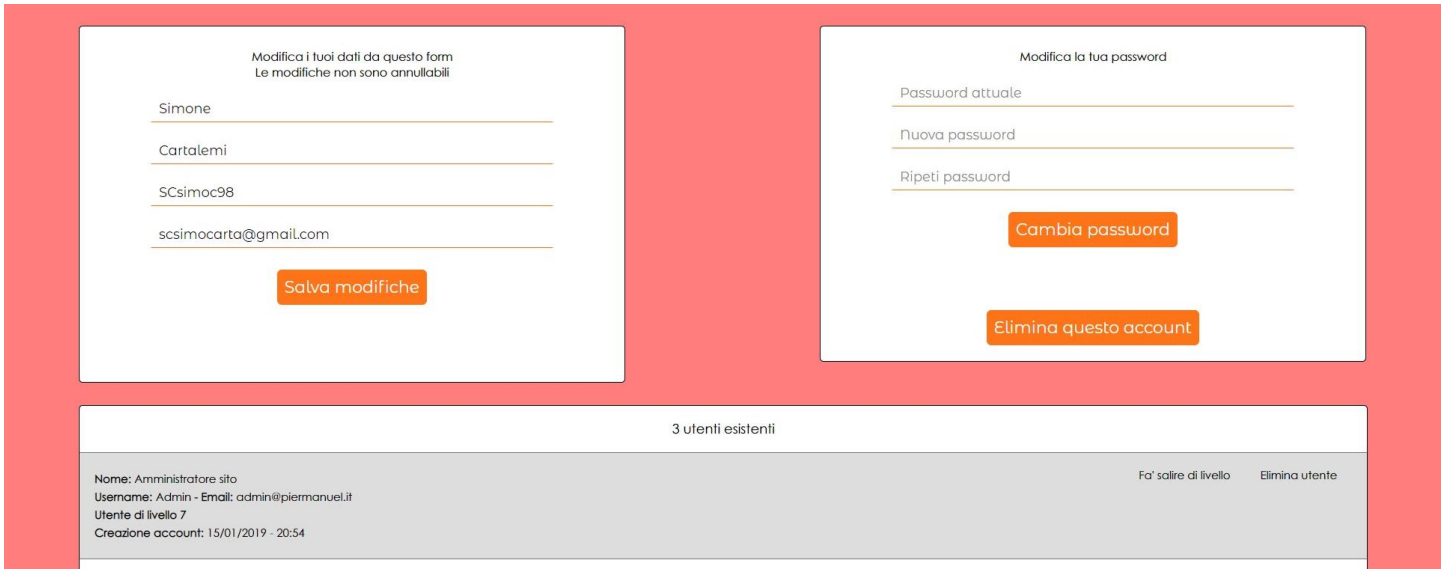


Da questa pagina è possibile gestire tutte le funzionalità del sito e consultarne lo stato.

Gestione utenti

La prima sezione è dedicata alla gestione dell'utente, dove quest'ultimo potrà aggiornare i propri dati, la propria password ed eliminare l'account. Se si dispone dei permessi necessari sarà anche possibile consultare lo stato degli altri utenti iscritti sulla piattaforma.

Di seguito la schermata della pagina

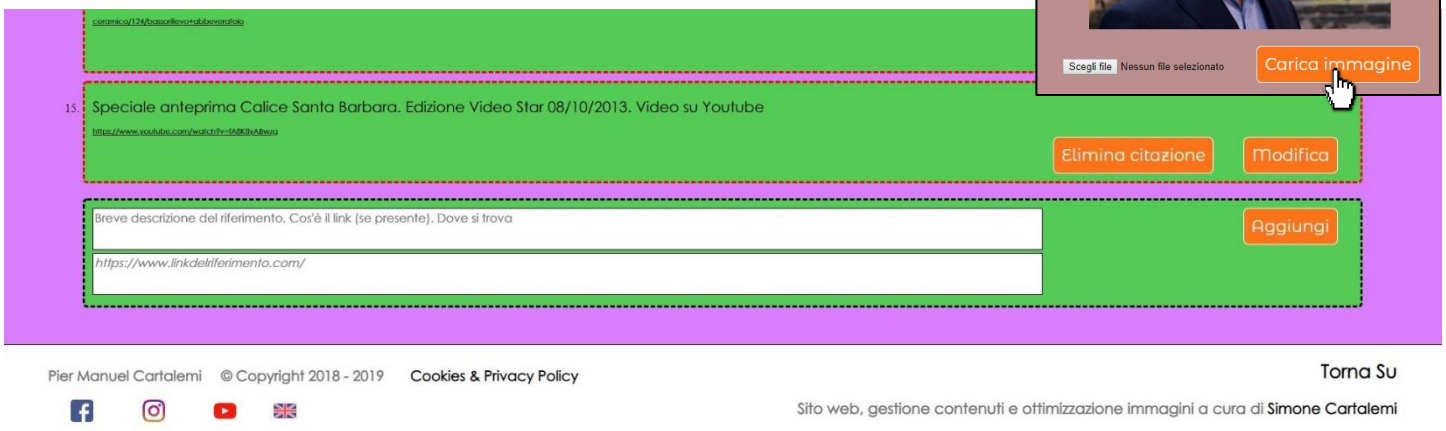


Biografia

È facilmente gestibile la sezione che tratta la biografia. Per modificare una versione biografica all'interno del database è sufficiente selezionare la lingua da modificare dall'apposito tasto e salvare le modifiche. È inoltre possibile aggiornare i file PDF del curriculum (non devono superare i 4MB). **I file devono chiamarsi esattamente come indicato** dal sito, altrimenti si riceverà un errore del caricamento. Infine è facilmente consultabile la parte che riguarda le citazioni, quindi aggiungerle, modificarle o eliminarle. Le schermate:

A sinistra abbiamo la parte che riguarda le citazioni, mentre sotto abbiamo la parte che riguarda l'aggiornamento della biografia.

Nella sezione "HTML" (a destra) è possibile aggiornare la foto della home, scegliendo un file jpg non superiore ai 4MB.



Articoli

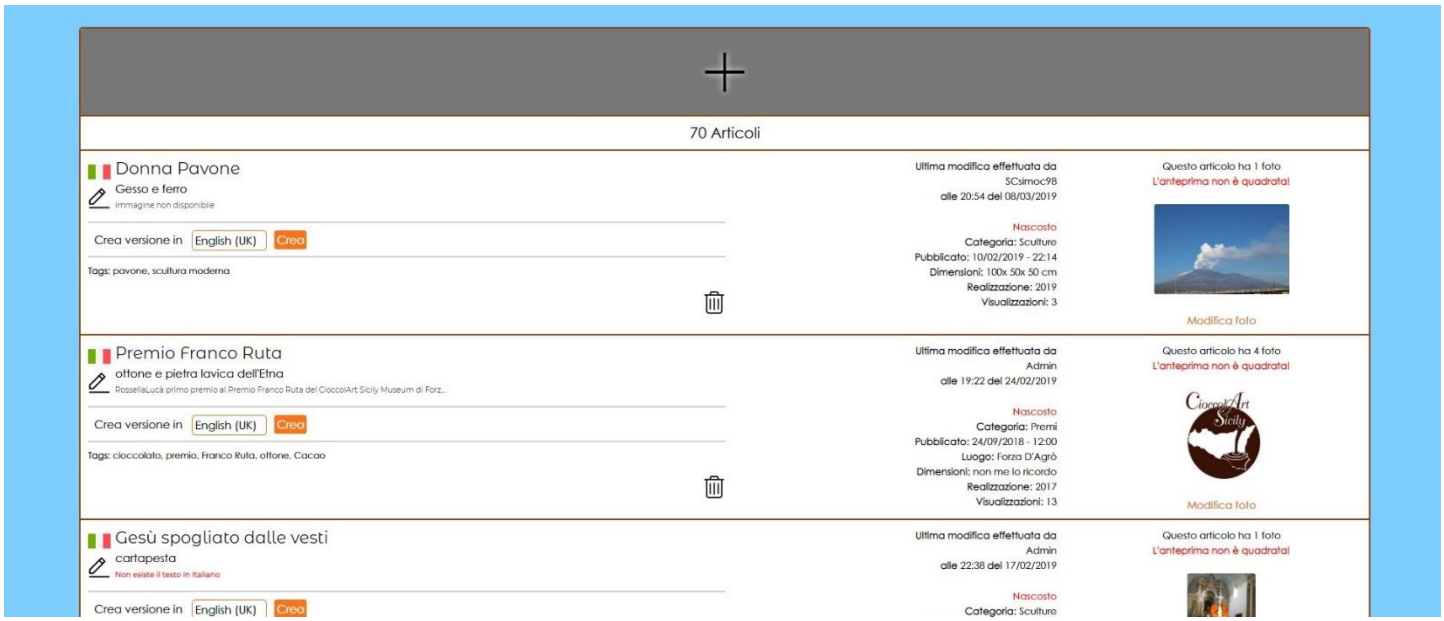
Sono molteplici le funzioni per gestire gli articoli, a cominciare dalle categorie (sempre in "HTML") poiché è possibile aggiungerle e rimuoverle. Queste verranno automaticamente aggiornate anche nella galleria. Sempre nella stessa pagina è possibile consultare i file log delle operazioni effettuate sul database in caso di problemi di vario genere.

La sezione "Articoli" offre diversi servizi. È possibile sfruttare la barra di ricerca per trovare facilmente gli articoli che rispettano i criteri per modificarli o aggiornarli. Selezionando le apposite spunte è possibile scegliere se si vuole o no ricercare le parole chiavi anche in quei campi selezionati. È possibile filtrare ulteriormente i risultati scegliendo eventualmente una categoria specifica da cercare, un utente che ha effettuato l'ultima modifica oppure tutti gli articoli

che sono oppure non sono in una determinata lingua. Infine, questi risultati possono essere ordinati secondo un campo, ma non bisogna dimenticare che in qualsiasi caso **gli articoli nascosti si trovano sempre in alto**. Ecco come appare la barra di ricerca e i suoi filtri.

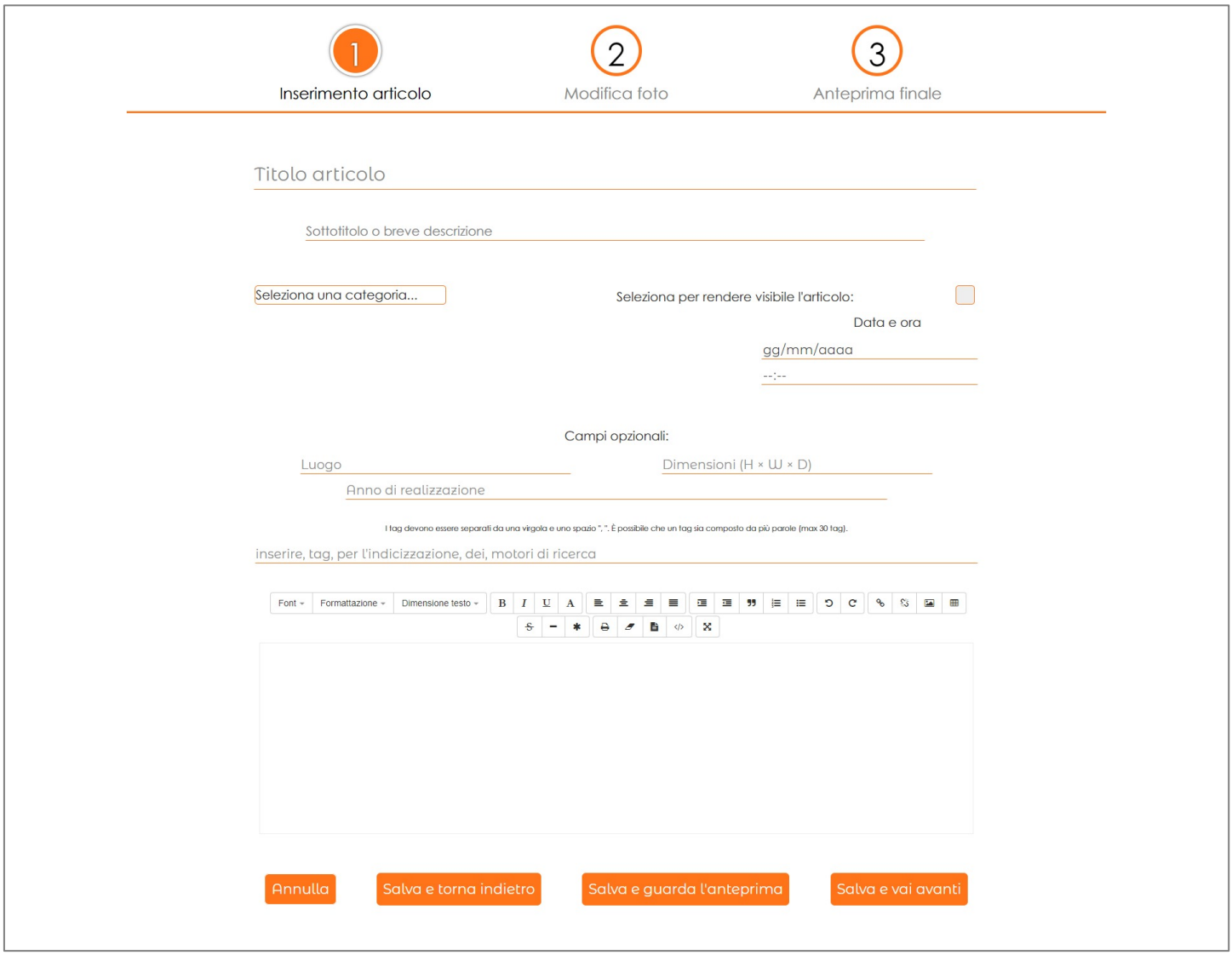
Subito in basso abbiamo il pulsante per aggiungere gli articoli e tutti gli articoli esistenti. La lingua predefinita è sempre l'italiano.

Ci sono gli appositi pulsanti per effettuare tutte le operazioni sugli articoli e sulle relative sezioni. **Non sarà possibile eliminare la sezione di un articolo se questa è unica**.



Aggiungere un articolo è molto semplice: i campi minimi sono il titolo e la categoria. Sarebbe buona norma aggiungere anche un sottotitolo, un anno di realizzazione (dal 1950 al 2200), dei tag per aiutare l'indicizzazione sui motori di ricerca (ogni tag va separato con ", " dal successivo) ed eventualmente, quando ha senso, luogo dell'evento o della posizione dell'opera e dimensioni dell'opera. Si noti che attraverso la spunta si può decidere se pubblicare immediatamente l'articolo o renderlo provvisorio per eventuali modifiche successive. Opzionalmente si può decidere manualmente data e ora dell'articolo: questo è stato fatto

soprattutto per far sì che gli eventi abbiano una fascia oraria a discrezione dell'utente, ma anche in caso di previsione di una pubblicazione. Durante la modifica sarà presente una spunta per decidere se aggiornare l'orario di pubblicazione o lasciarlo invariato.



Salvando l'articolo e andando avanti si giunge alla sezione di aggiunta, modifica e rimozione delle foto.

Passando con il mouse su una foto appariranno quattro funzioni: elimina, sposta a destra o a sinistra e modifica. È facilmente intuibile lo scopo delle prime tre funzioni, mentre per quanto riguarda l'ultima apparirà una schermata dove sarà possibile rinominare la foto, ruotarla o scaricarla, come mostrato a destra.

Queste erano le funzionalità generali della web app.

Statistiche

Per migliorare il servizio esiste una pagina dedicata alla visualizzazione delle statistiche, anch'essa molto intuitiva.