

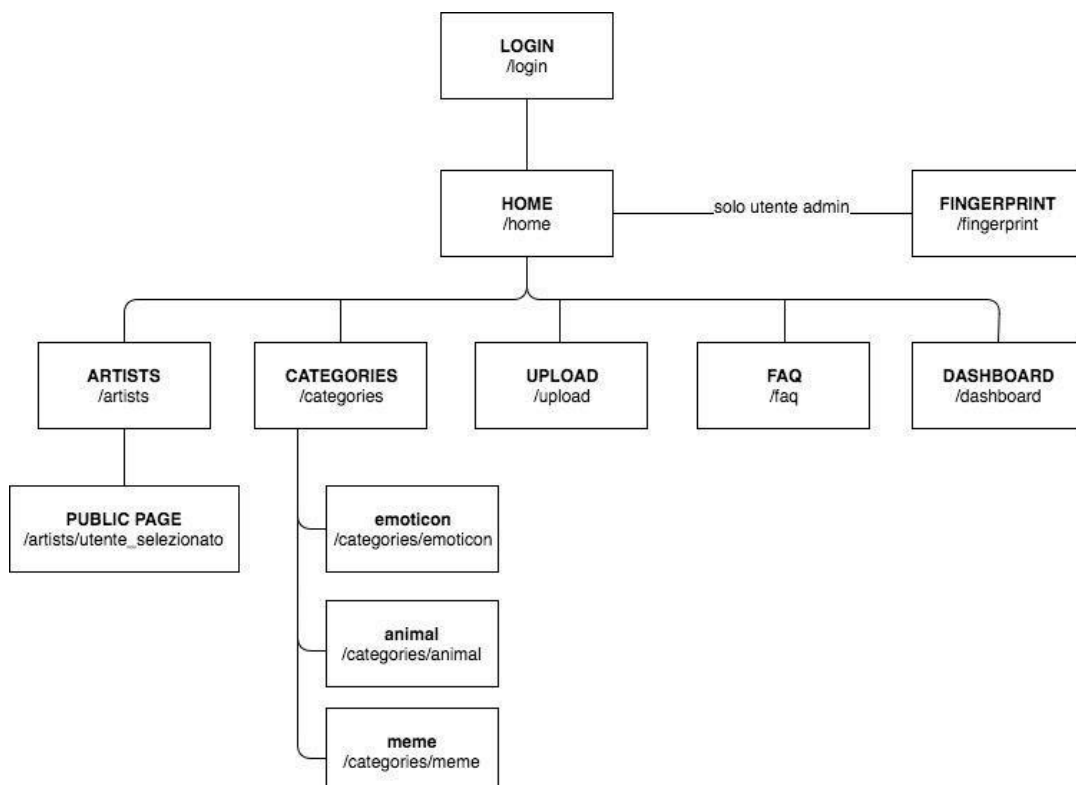
RELAZIONE PROGETTO TWEB ANDREA AZZALIN(778097)

TEMA DEL SITO:

Lo scopo principale del sito è la condivisione e la navigazione di immagini animate in formato GIF con altri utenti. L'utente può accedere alle varie pagine soltanto se registrato e possiede una pagina personale dove può gestire le proprie Gif e eliminare le preferite / scaricate. L'utente possiede anche una pagina pubblica dove tutti gli altri utenti possono vedere le sue gif caricate. Le gif sono suddivise in categorie.

Sezioni principali:

- **PAGINA LOGIN:** dove l'utente può registrarsi o accedere al sito in caso di utente già iscritto.
- **HOME:** vengono visualizzate tutte le gifs presenti sul database l'utente ha la possibilità di aggiungere una o più gif tra i preferiti o visitare la pagina del proprietario.
- **ARTISTS:** viene caricata la lista di tutti gli artisti con la possibilità di selezionarne uno per visitare la pagina pubblica dell'artista.
- **CATEGORIES:** filtra le gif in base alla categoria selezionata dal menù.
- **UPLOAD:** l'utente ha la possibilità di caricare le proprie gif e aggiungerle alla propria collezione.
- **FAQ:** pagina di informazione sulle funzionalità del sito ed eventuali domande comuni.
- **DASHBOARD:** sezione in cui l'utente può gestire le sue gif preferite e caricate.
- **BADGUY:** solo l'admin può accedere a questa pagina dove vengono visualizzate una serie di informazioni dei vari utenti registrati.



Funzionalità

il sistema è stato progettato con un architettura MVC, questa architettura ha molti vantaggi a livello di sicurezza e modularità. Quando si accede all'index del sito viene creata un'istanza della classe App dove viene mappato l'url e istanzia la classe Controller che a sua volta si occuperà di caricare la view richiesta dal client. Questo meccanismo viene chiamato **Routing**, su tale meccanismo sono basati i più moderni frameworks, grazie al routing si ha un vantaggio in termini di sicurezza perché il client non visualizza mai la reale struttura delle directory e non è più difficile eseguire attacchi di tipo XSS o html injection.

Gestioni utenze, sessioni

L'utente può accedere ai contenuti del sito solo tramite un login/registrazione, i parametri del form di registrazione e di login vengono validati lato client da un plugin jQuery chiamato validator, mentre lato server viene controllato che effettivamente siano stati settati correttamente.

Quando un utente tenta di registrarsi, viene verificato se il nickname inserito esiste già nel database e viene avvisato con una notifica. I parametri del form vengono passati al server tramite il metodo POST con AJAX in maniera trasparente all'utente. Le password degli utenti quando vengono ricevute dal server vengono subito crittate tramite l'algoritmo bcrypt (bcrypt è l'algoritmo di hashing di password di default per [BSD](#) e altri sistemi), la password viene crittata per due ragioni, la prima che gli utenti tendono a utilizzare password comuni per più siti/servizi quindi nel caso il sistema venisse bucato le password associate agli utenti non vengono poste su un piatto d'argento, e per seconda tentare un bruteforcing su tali password o tentare una decriptazione impiegherebbe troppo tempo(anni). Quando un utente riesce a loggarsi viene creata una sessione e viene mantenuta finché non scade(24 minuti min di default) o si esegue il logout. All'utente loggato viene catturato l'ip e passato come parametro a delle API fornite da ip-api.com che inviano un file JSON sul server contenente delle informazioni di profilazione dell'utente, come la zona da cui si è collegato,isp,ecc..., Queste informazioni sono visibili solo all'utente 'admin' nella sezione fingerprint, , tali informazioni possono essere utili per profilare gli utenti e creare pubblicità personalizzata o servizi simili.Quando un utente cambia vista, quindi cambia pagina viene rigenerato l'id della sessione per evitare attacchi di tipo 'hijacking session' . Per il logout, viene eseguita l'unset di tutte le variabili di sessione e successivamente viene eseguita la destroy. Per aumentare la sicurezza, ormai standard è associare al dominio un certificato SSL che codifica i pacchetti passati tra client/server.

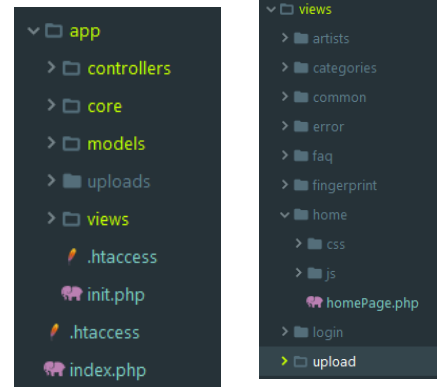
Contenuti utente

Gli utenti possono eseguire l'upload soltanto di file di tipo .gif dalla sezione upload tramite drag and drop.

Anche in questa sezione vengono eseguite validazioni degli input sia lato client che lato server e viene verificato se il titolo della gif che l'utente sta caricando è già esistente nel database o se vengono lasciati dei campi vuoti. Il file caricato viene caricato nella cartella "uploads". Gli utenti quando eliminano una gif dalla propria dashboard vengono prima eliminati dalla view tramite un metodo della libreria Masonry e successivamente dal database. Tutte queste operazioni avvengono sempre tramite una comunicazione ajax, del tutto trasparente all'utente ma comunque notificato al successo o fallimento di un'operazione.

Struttura delle directory

La struttura delle directory è stato suddiviso in base alle sezioni Model Views e Controllers del pattern, ho inserito ulteriori cartelle necessarie per un'organizzazione più ordinata. Nella cartella core ci sono i file php, appunto, "core" del sito. Nelle cartelle controllers e model sono contenuti i file relativi ai controller e models. Tutti i file che interessano il frontend sono contenuti in views dove ogni cartella è suddivisa per vista e a sua volta in sottocartelle per i fogli di stile e javascript specializzato per quella view, mentre in common ci sono i file comuni per tutte le views come gli import e il tema Bootstrap. Nella root del sito e anche nella cartella core vi è un file di configurazione .htaccess che impone alcune regole sul comportamento del server.



FrontEnd

L'intera grafica è stata implementata con Bootstrap 4 utilizzando un tema personalizzato open source chiamato Neon Glow(codice sorgente nella sezione FAQ) , essendo il responsive uno dei punti forte di questo framework non ci sono problemi di usabilità con display di varie dimensioni. Per una disposizione dinamica e "carina" delle gifs ho utilizzato una libreria js chiamata Masonry, che impostando i parametri per le dimensioni della griglia gli elementi verranno inseriti in maniera ottimale per riempire la pagina. Nella sezione in cui si interpellano le pagine pubbliche che o private degli artisti, l'immagine avatar comune per tutti gli artisti cambia colore ad ogni caricamento della pagina, questo grazie a uno script js che aggiunge un filtro con parametri random permettendo il cambio di colore continuo ad ogni refresh della pagina.

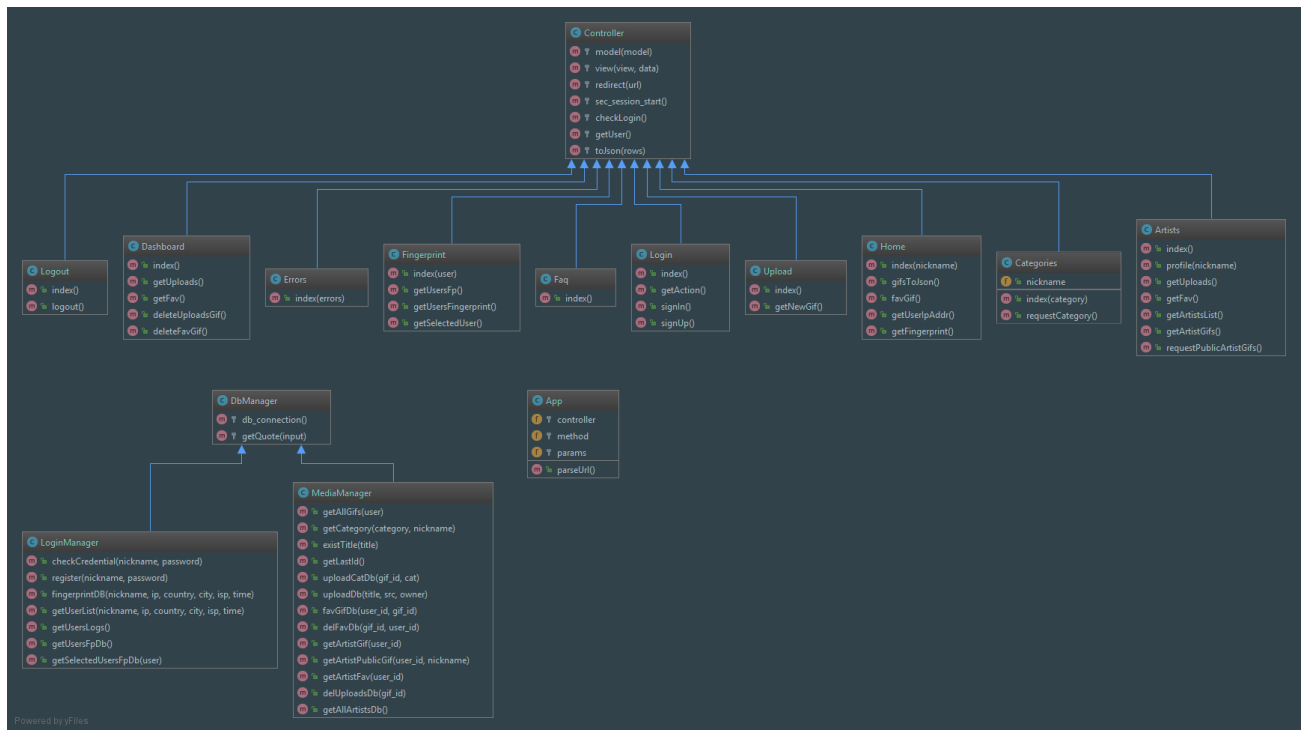
BackEnd

La classe Controller rappresenta il controller del pattern MVC e implementa i metodi comuni come l'assegnazione della view e del model o la gestione della sessione ,mentre le classi figlie estendono la classe padre con metodi specializzati per le varie sezioni del sito. La classe DbManager è la classe padre dei model e si occupa di gestire la connessione de database ed è estesa dai due classi:

- LoginManager: si occupa di gestire le query riguardanti gli utenti
- MediaManager: si occupa delle query che interagiscono con le informazioni per le Gifs.

La classe App come accennato prima, implementa il meccanismo di routing e ha come variabili di istanza il controller, il metodo, e un array per i parametri, che acquisteranno un valore diverso in base alla richiesta passata dalla url.

In questo diagramma delle classi si può notare come siano strutturate le varie classi PHP.



Esempi di metodi remoti della classe Artist:

- **getUploads()** : quando invocato ritorna tutte le gifs dell'utente con sessione attiva.

Metodo di invocazione tramite richiesta GET:

<http://localhost/tweb/artists/getUploads>

Esempio di valore di ritorno per utente admin:

```
[{"id":"37","title":"mkkj","src":"2018-01-19_18-35-00_uploadBy_admin","owner":"admin"}, {"id":"90","title":"dead inside","src":"2018-01-26_20-41-00_uploadBy_admin","owner":"admin"},...]
```

- **getArtistGifs()**: quando invocato ritorna tutte le gifs artista passto come parametro "artists"

Metodo di invocazione tramite richiesta POST con parametri artist e user settati:

<http://localhost/tweb/artists/getArtistGifs>

Esempio di valore di ritorno per utente admin e user a:

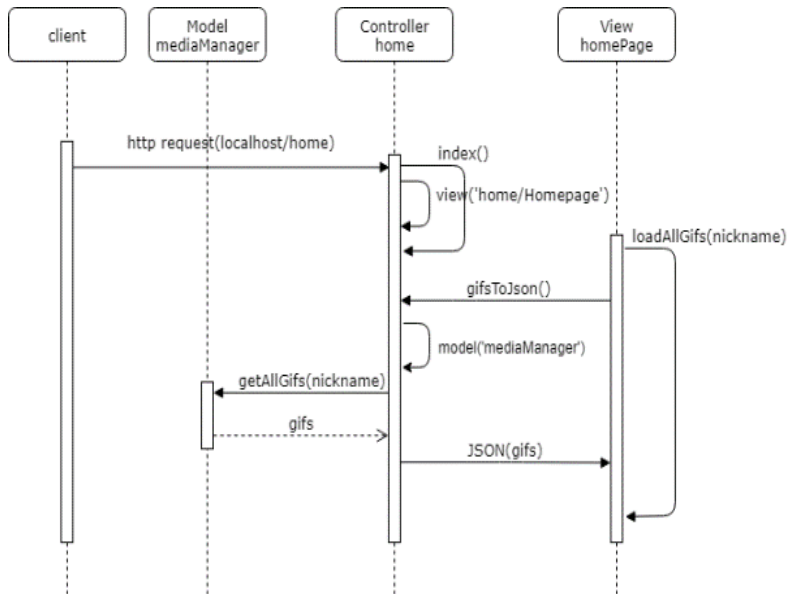
```
[{"id":"37","title":"mkkj","src":"2018-01-19_18-35-00_uploadBy_admin","owner":"admin","user":null,"gif_id":null},...]
```

Questi metodi vengono invocati dalle views ai controller, in caso di successo i file javascript ricevono l'output di questi metodi in json e reagiscono di conseguenza, modificando dinamicamente gli elementi HTML.

In questo SSD(semplificato) viene rappresentato come comunicano client e server, ho usato come esempio lo scenario in cui un utente logga e viene portato nella homepage (/home).

La classe App imposta come controller home, quindi viene eseguito il metodo index() che a sua volta imposta la view homePage.

Caricata la view viene inviata una richiesta GET tramite ajax con javascript, richiedendo tutte le gifs del database, il controller gestisce l'input e lo invia al model(MediaManager) che a sua volta interroga il database, in caso di successo ritorna le informazioni richieste dal controller. Il controller riceve le informazioni e le codifica in JSON e le rinvia alla view che modificherà la struttura HTML di conseguenza.



Il formato JSON creato in questo caso sarà:

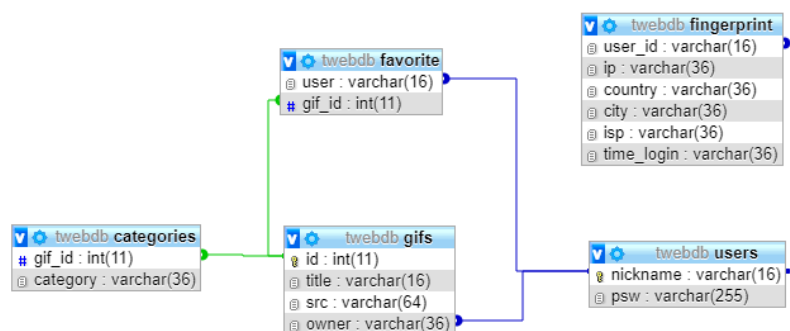
```
[{"id": "37", "title": "gif", "src": "2018-01-19_18-35-00_uploadBy_admin", "user": null, "owner": "admin"}, ...]
```

Database

La classe DbManager si occupa di creare la connessione e le classi figli implementano i metodi che interrogano il database.

In caso di errore per la connessione viene catturata l'eccezione e mostrato un messaggio di errore.

Tutte le query sono filtrate dal metodo prepare() che previene da attacchi di tipo sql injection.



MOCKUP

/login

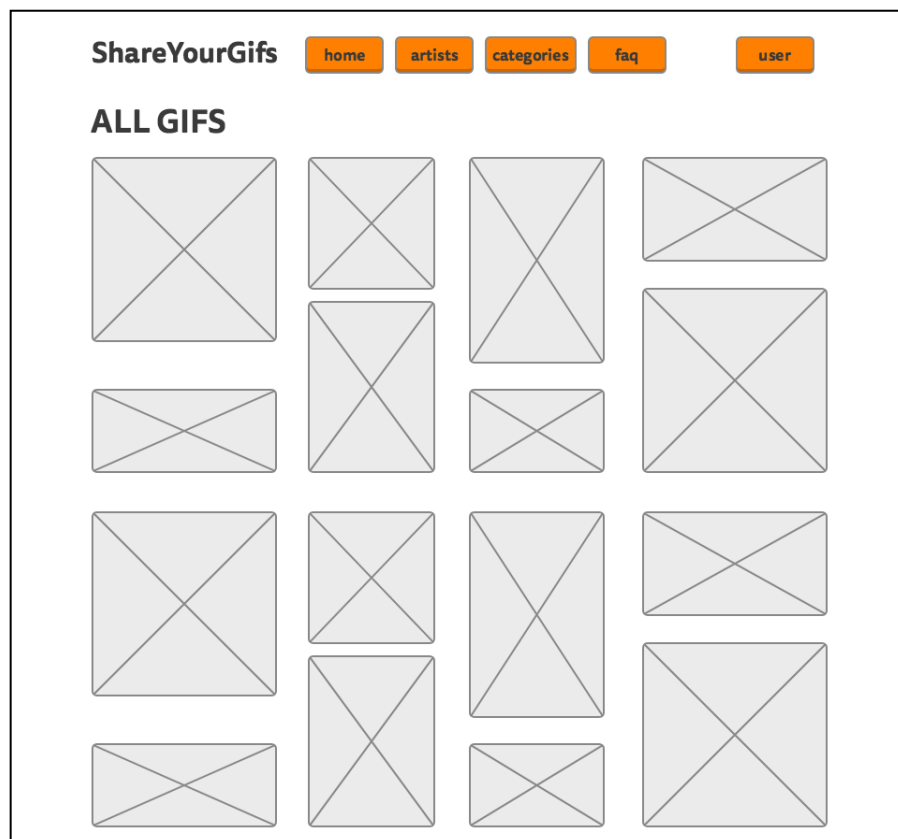
WELCOME TO ShareYourGifs

SYG is your top source for the best & newest GIFs online.
Find everything from animals GIFs, emoticons GIFs,
unique GIFs and more.Upload and share GIFs with other
people.

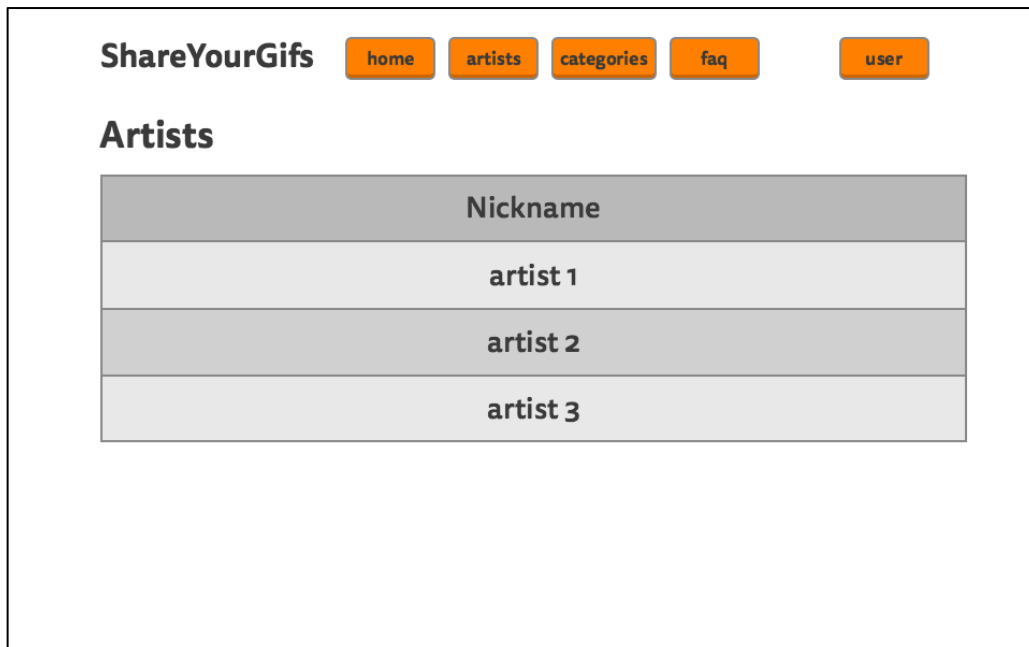
First time here? Click signUp below this banner

[signIn](#) [signUp](#)

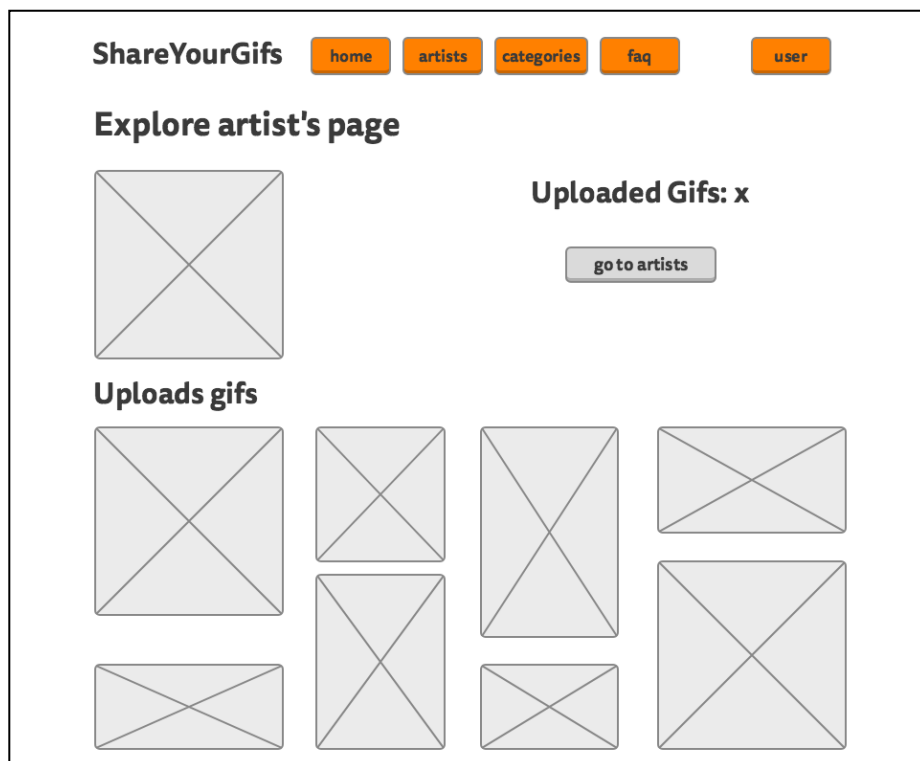
/home e /categories/categoria_selezionata



/artists



/artists/artista_selezionato

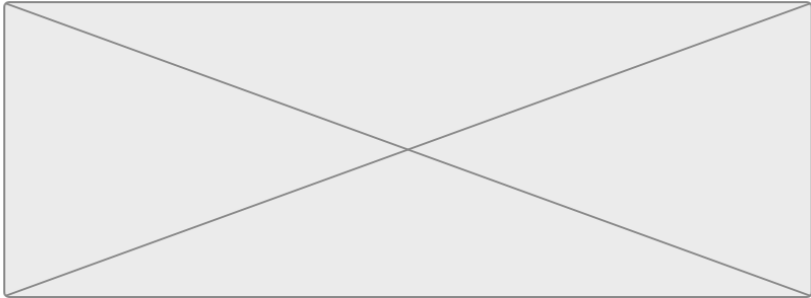


/upload

ShareYourGifs

homeartistscategoriesfaquser

UPLOAD



title

title

category

☐ Animal
☒ Selected Emoticon
☐ Meme

upload

/fingerprint

ShareYourGifs

homeartistscategoriesfaquser

BadGuy Page

select user

User	ip	country	city	isp	time
artist1	ip	country	city	isp	time
artist2	ip	country	city	isp	time
artist3	ip	country	city	isp	time

/dashboard

