Video Processing Cluster

2 Analisi 3

2.1 Analisi del dominio 3

2.5 Analisi dei mezzi 4

2.1.2 Software 4

2.1.3 Hardware 4

3 Progettazione 5

3.1 Design dell’architettura del sistema 5

3.2 7

3.3 Design dei dati e database 7

3.4 Design delle interfacce 7

4 Implementazione 9

5 Test 9

5.1 Protocollo di test 9

5.2 Risultati test 10

5.3 Mancanze/limitazioni conosciute 10

6 Consuntivo 10

7 Conclusioni 11

8 Bibliografia 12

8.1 Sitografia 12

9 Glossario 13

10 Indice delle figure 13

11 Allegati 13

# Analisi

## Analisi del dominio

Per questo progetto ci è stato chiesto di creare un sistema in cluster per l’elaborazione di filmati ed estrazione dei vari dati. Oggi per visualizzare le statistiche dei video bisogna passare attraverso molti programmi, ed è tutto meno che immediato. Con il cluster di server ci si assicura che il servizio sia sempre online e grazie alla GUI web il processo diventa user friendly ed intuitivo. Con questo progetto sarà poi possibile visualizzare le statistiche e scaricare il video con i motion vector, solo i frame I/B/P o tutti le immagini che compongono il video.

### 

### 2.5 Analisi dei mezzi

Per questo progetto ci sarà fornito 1 PC a testa, nel quale ci sarà un sistema operativo Windows 10 Pro, infine andremo ad usare delle VMs per simulare il cluster.

### Software

* 4 Linux VMs
  + 2 WebServer
  + 1 Load Balancer
  + 1 Database
* PHP
* Memcached

### Hardware

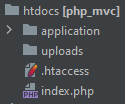
* SSD per memorizzare le Linux VMs
* PC con:
  + CPU Intel Core i7-7700
  + RAM 16 GB

# Progettazione

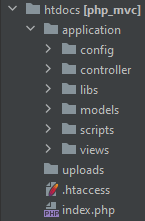
## Design dell’architettura del sistema

Struttura app web

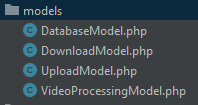
Directory pricipale => htdocs

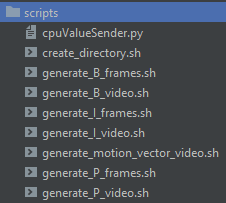
* **htdocs**: è la cartella principale dove sono contenuti i file dell’applicativo.
* **application**: è la cartella dove sono contenuti i file dell’applicazione.
* **uploads**: la cartella dove vengono messi i file video dati dall’utente.
* **.htaccess**: è un file che serve a spostare l’utente tra le pagine.
* **index.php**: è la pagina iniziale che attiva l’applicazione.

App => application

* **config**: contiene delle configurazioni principali, i “define”.
* **controller**: contiene il file “home” che controlla il passaggio di dati.
* **libs**: contiene le librerie, in questo caso solo application.php.
* **models**: contiene tutte le classi di progetto.
* **scripts**: contiene tutti gli script ffmpeg + python.
* **views**: contiene tutti i documenti html +css.

Classi => models

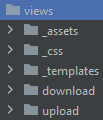
* **DatabaseModel**: serve per collegarsi al db, serve a fare query e serve a inserire dati.
* **DownloadModel**: serve per scaricare i files creati.
* **UploadModel**: server per caricare il file video + controlli.
* **VideoProcessingModel**: processa il video creando i files.

****Scripting => scripts

Questi scripts servono per aiutare l’applicativo.

* **cpuValueSender.py**: manda il valore cpu al LB.
* **create\_directory.sh**: crea le varie cartelle.
* **generate\_X\_frames.sh**: genera i frames X.
* **generate\_X\_video.sh**: genera il video usando i frame X.
* **generate\_motion\_vector\_video.sh**: genera il video con i vettori visibili.

GUI => views

In questa cartella sono contenuti tutti i file per la GUI.

* **\_assets**: contiene immagini e formati testo.
* **\_css**: contiene tutti i file css.
* **\_templates**: contiene l’header e il footer.
* **upload**: contiene i file per la pagina di upload.
* **download**: contiene i file per la pagina di download.

## Design delle interfacce

Upload

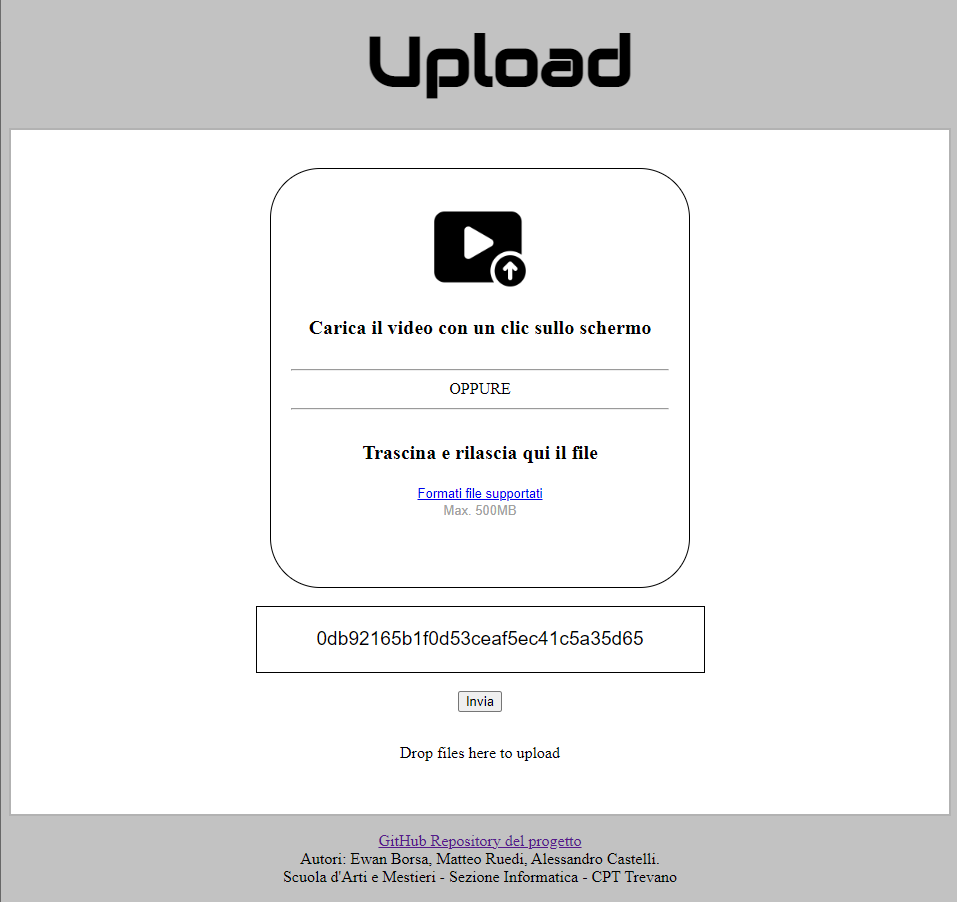


Figura 1 - Pagina di Upload

Questa è l’interfaccia della pagina di upload, dove si può o trascinare il file o schiacciare per sceglierlo con l’esplora file, al centro della pagina abbiamo messo un UUID per identificare la sessione.

Download



Figura 2 - Pagina di Download

Questa è l’interfaccia della pagina di download, dove si scaricare i file schiacciando i bottoni per sceglierli, in basso ci sono i dati statistici insieme ad un grafico che mostrano il numero di frame.

# Implementazione

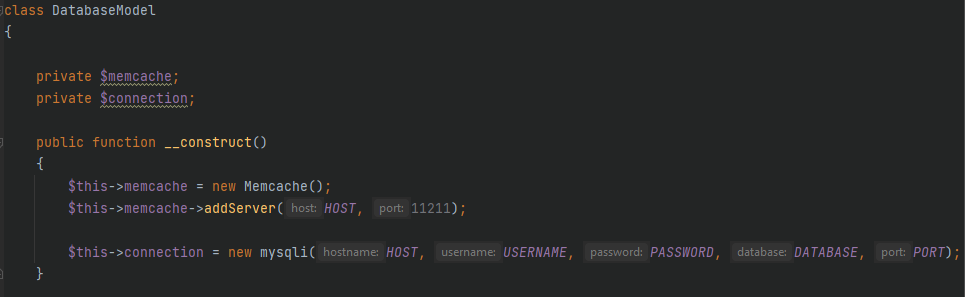
## Application MVC

### Models(Classi)

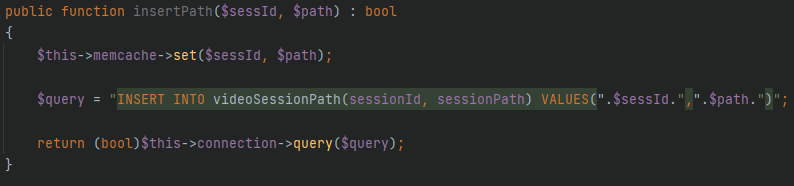
**DatabaseModel**

DatabaseModel è una classe che serve per il collegamento con il Databas.  
Ha solo due attributi privati:

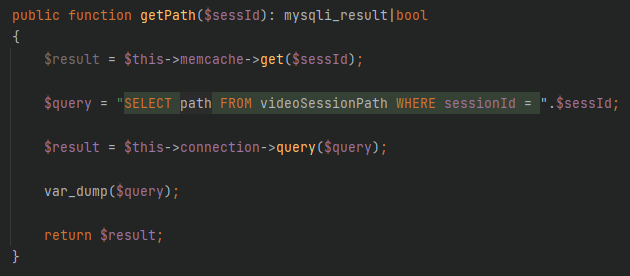
* **memcache** (per comunicare con il server Memcached)
* **connection** (oggetto mysqli per comunicare col db)



Il costruttore non chiede argomenti, crea da solo le due istanze, la prima è quella di “memcache” che servirà a connettersi al server “Memcached”, la seconda invece è “connection” che servirà a creare un oggetto “mysqli” che verrà usato per collegarsi al database.



Il metodo “insertPath” serve ad inserire un nuovo record con dentro l’id di sessione e il percorso del file.



Il metodo “getPath” serve a chiedere il percorso attraverso l’id di sessione.

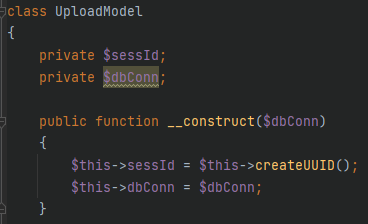


Questo metodo, “getFormatList”, serve a prendere la lista dei formati dal database.

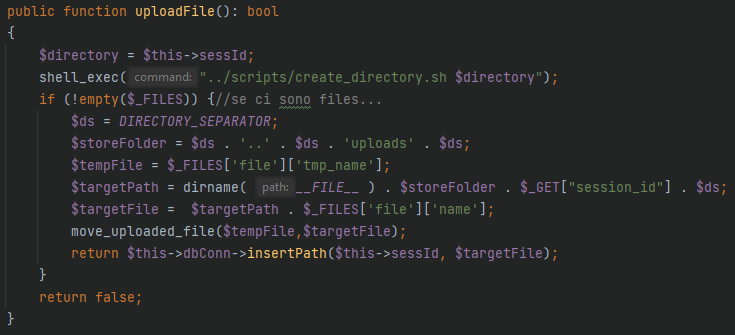
**UploadModel**

UploadModel è una classe che serve per l’upload del file video.  
Ha solo due attributi privati:

* **sessId** (Id di sessione)
* **dbConn** (DatabaseModel per le connessioni col db)

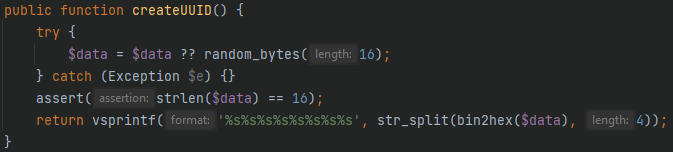


Il costruttore richiede solo il dbConn che rappresenta l’oggetto DatabaseModel che servirà a fare le query al db.



Il metodo “uploadFile()” serve a fare l’upload effettivo del file, crea la cartella dove verrà messo insieme alle versioni create da ffmpeg.

Infine inserisce un record nel db con scritto l’id e il percorso.  
Ritorna un valore booleano che indica il funzionamento del metodo.



“createUUID()” è un metodo che serve per la creazione di “Universally unique identifier” che verranno usati per gli id di sessione del nostro progetto.

La creazione del UUID avviene tramite la generazione 16 bytes di dati randomici, poi vengono stampati in esadecimale.

**VideoProcessingModel**

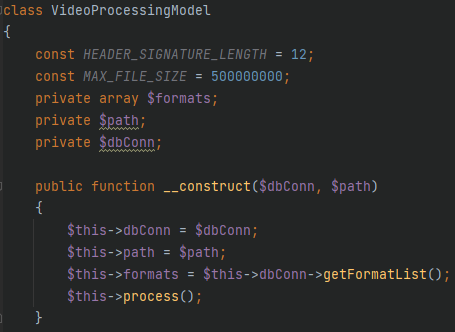
VideoProcessingModel è una classe che si occupa di controllare il file uploadato, analizzarlo e infine di creare i files tramite ffmpeg.

Ha due attributi privati:

* **formats** (array dove sarranno contenuti tutti i formati presi dal db)
* **path** (percorso della sessione attuale)
* **dbConn** (DatabaseModel per le connessioni col db)

Ha due costanti:

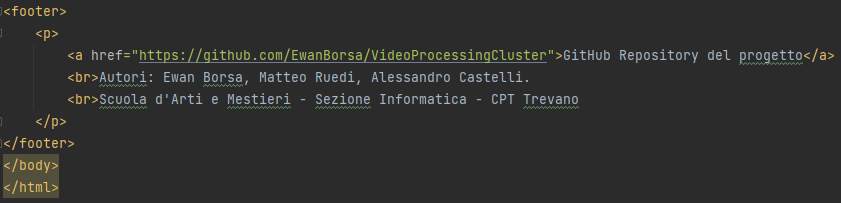
* **HEADER\_SIGNAURE\_LENGTH** (max grandezza della signature dei video)
* **MAX\_FILE\_SIZE** (Grandezza massima dei file video)



**DownloadModel**

## Views(GUI)

**\_templates/footer.php**

****

Questo è il footer generico con un link al repository e informazioni generali.

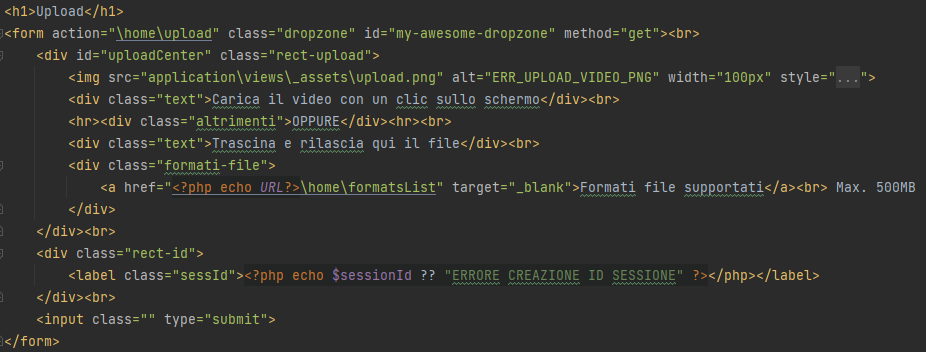
**\_templates/header.php**

****

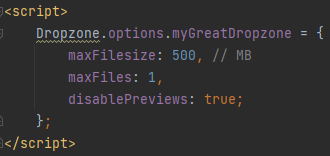
Questo è il header generico con tutte le dipendenze css/javascript.

**Upload**

In questa pagina è contenuto il form con dropzone che servirà a fare l’upload del file video. In un link verrà messo la pagina con i formati. Al centro ci sarà l’id stampato in grande. Infine ci sarà un submit per mandare il video e anadare nella pagina di download.



Parte script in javascript per configurare il dropzone:

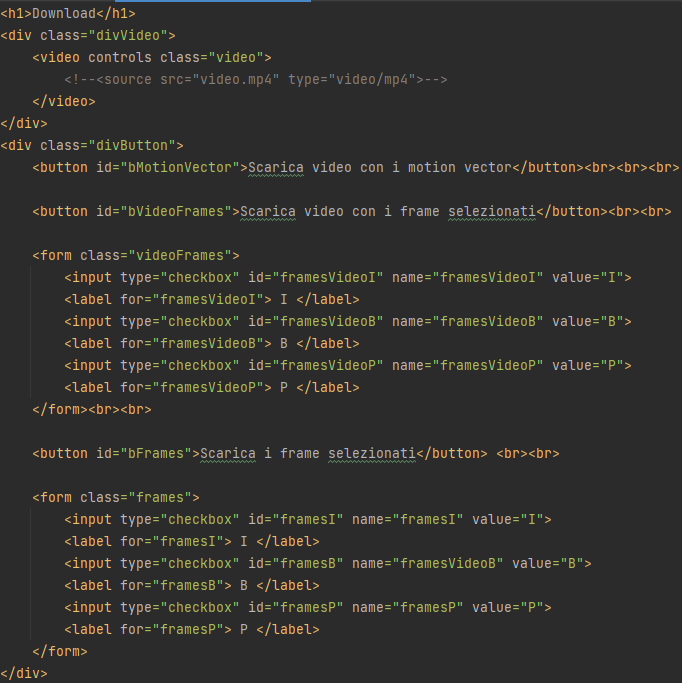


**Download**

In questa pagina è contenuto il video caricato prima, dei bottoni, una statistica e un grafico.

Nella prima parte di codice è contenuta la parte di interfaccia interagibile con l’utente:

* Il video visibile da guardare quando si vuole.
* I bottoni per scaricare i files creati da ffmeg.



Nella seconda parte di codice sono contenute la statistica e il grafico:

### Controller()

# Conclusioni

Considerazioni Ewan

Questo progetto ha avuto un gran numero di problemi, mi dispiace di non essere arrivato al risultato sperato, ma mi accontento di aver migliorato le mie conoscenze su *php* e gli altri linguaggi di programmazione che abbiamo usato per il progetto. Inoltre ho trovato molto interessante scoprire la metodologia cluster e di come funziona un *load balancer*.

# Bibliografia

## Sitografia

* <https://app.moqups.com/>, *Moqups*, 27-01-2023.
* <https://www.laratutorials.com/drag-and-drop-file-upload-using-dropzonejs-php-mysql>, *Drag&Drop Dropzone PHP*, 27-01-2023.
* <https://www.dropzone.dev/>, *Dropzone*, 27-01-2023.
* <https://www.w3schools.com/php/php_file_upload.asp>, *PHP file upload*, 27-01-2023.
* <https://docs.dropzone.dev/getting-started/setup/server-side-implementation>, *Server Side Dropzone,* 03-02-2023.
* <https://stackoverflow.com/questions/22096383/uploading-files-to-server-php>, *Uploading Files to Server in PHP*, 10-02-2023.
* <https://ffmpeg.org/>, *FFMPEG*, 10-02-2023.
* <https://www.haproxy.com/de/blog/haproxy-configuration-basics-load-balance-your-servers/>, *HaProxy Configuration*, 17-03-2023.
* <https://filesamples.com/>, *File Samples*, 17-03-2023.
* [https://www.garykessler.net/library/file\_sigs.html/](https://ffmpeg.org/), *Files Signature*, 17-03-2023.
* [https://en.wikipedia.org/wiki/List\_of\_file\_signatures](https://ffmpeg.org/), *Files Signature Wikipedia*, 17-03-2023.
* [https://docs.python.org/3/library/socket.html](https://docs.python.org/3/library/socket.html#constants), *Python Socket,* 31-03-2023.
* <https://stackoverflow.com/questions/6380057/python-binding-socket-address-already-in-use>, *Biding* *Python Socket,* 31-03-2023.
* <https://devdocs.magento.com/guides/v2.3/config-guide/memcache/memcache_ubuntu.html>, *Memcache Ubunto, 31-03-2023.*
* <https://www.tomshardware.com/how-to/stress-test-cpu-in-linux>, *CPU Stress in Linux, 21-04-2023.*
* <https://stackoverflow.com/questions/11370371/php-download-a-file-from-web-to-local-machine>, *Download File PHP, 28-04-2023.*