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

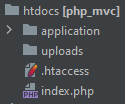
* SSD per memorizzare le Linux VMs
* PC con:
  + CPU Intel Core i7-7700
  + RAM 16 GB

# Progettazione

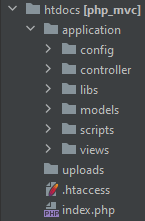
## Design dell’architettura del sistema

Struttura app web

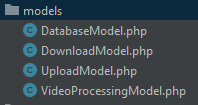
Directory pricipale => htdocs

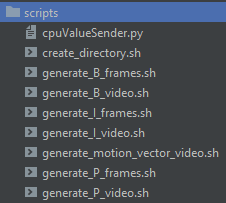
* **htdocs**: è la cartella principale dove sono contenuti i file dell’applicativo.
* **application**: è la cartella dove sono contenuti i file dell’applicazione.
* **uploads**: la cartella dove vengono messi i file video dati dall’utente.
* **.htaccess**: è un file che serve
* **index.php**: è la pagina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

* **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.

## Design delle interfacce

Upload

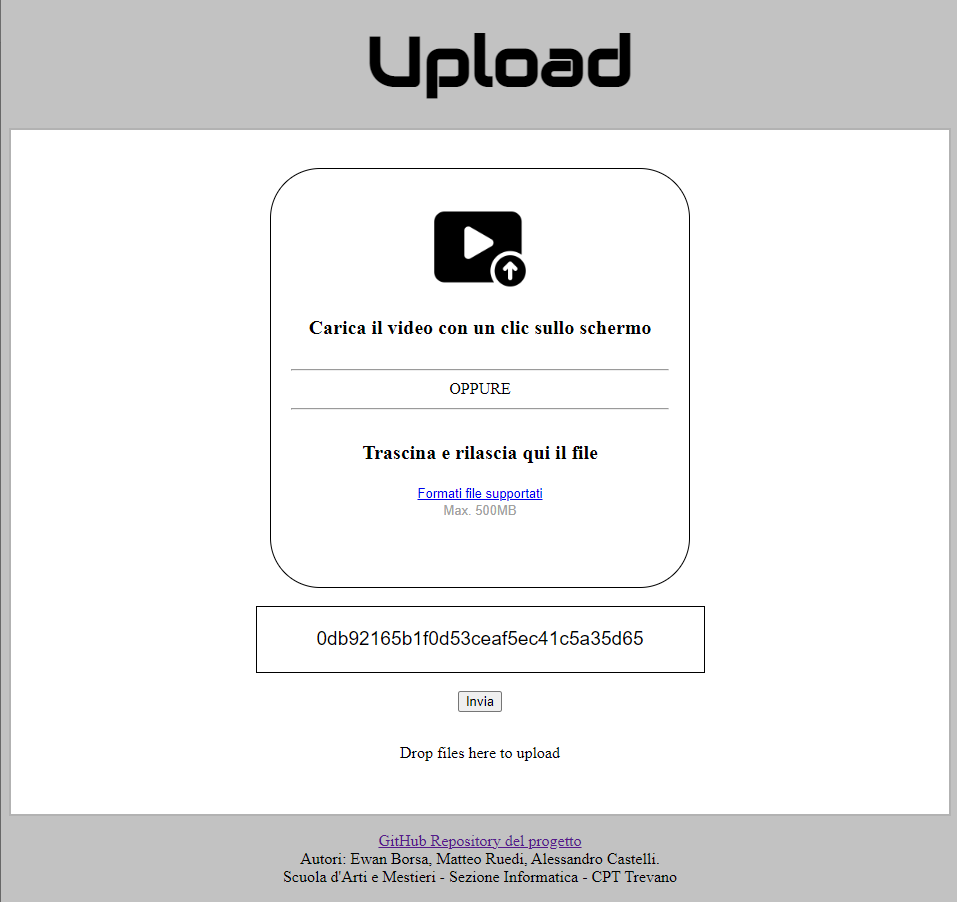


Figura 1 - Pagina di Upload

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

**DA FARE**

# Test

## Protocollo di test

## Risultati test

## Mancanze/limitazioni conosciute

**DA FARE**

# Consuntivo

**GANTT CONSUNTIVO**

# 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

1. URL del sito (se troppo lungo solo dominio, evt completo nel diario),
2. Eventuale titolo della pagina (in italico),
3. Data di consultazione (GG-MM-AAAA).

* [https://docs.python.org/3/library/socket.html](https://docs.python.org/3/library/socket.html#constants), *Python Socket,*
* https://stackoverflow.com/questions/6380057/python-binding-socket-address-already-in-use
* https://devdocs.magento.com/guides/v2.3/config-guide/memcache/memcache\_ubuntu.html
* https://www.tomshardware.com/how-to/stress-test-cpu-in-linux
* https://stackoverflow.com/questions/11370371/php-download-a-file-from-web-to-local-machine

**Esempio:**

* http://standards.ieee.org/guides/style/section7.html, *IEEE Standards Style Manual*, 07-06-2008.

# Glossario

|  |  |
| --- | --- |
| **Termine** | **Significato** |
| ASCII | American Standard Code for Information Interchange, codice per la codifica di caratteri. |
|  |  |
|  |  |
|  |  |
|  |  |
| ZFS | Zettabyte File System, filesystem opernsource a 128 bit con funzionalità avanzate. |

# Indice delle figure

[Figura 1 Esempio di diagramma di Gantt 7](#_Toc124775984)

[Figura 2 Esempio di diagramma di Gantt 8](#_Toc124775985)

[Figura 3 Diagramma ER Chen 10](#_Toc124775986)

[Figura 4 Diagramma ER Barker 11](#_Toc124775987)

[Figura 5 Esempio di diagramma di Gantt consuntivo 16](#_Toc124775988)

# Allegati

Elenco degli allegati, esempio:

* QDC
* Diari di lavoro
* Codice sorgente