Video Processing Cluster

1 Introduzione

1.1 Informazioni sul progetto

Allievi: Matteo Rüedi, Ewan Borsa, Alessandro Castelli
Formatore: Geo Petrini
Classe: I3BB, Centro Professionale Trevano, sezione informatica, modulo 306
Data inizio: 27 gennaio 2023
Data fine: 5 maggio 2023
Numero di ore a disposizione: 96 ore scolastiche(45min)

1.2 Scopo

Lo scopo di questo progetto è realizzare un sistema in cluster per l'elaborazione di filmati e l'estrazione di dati statistici.

2 Analisi

2.1 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.

2.2 Analisi e specifica dei requisiti

Requisito	Req-001	**Priorità**	1	**Versione**	1.0
Nome	Vagrant up con macchine virtuali funzionanti				
Note					
Sotto requisiti					
001					
002					
003					

Requisito	Req-002	**Priorità**	1	**Versione**	1.0
Nome	Sito funzionante su webserver				
Note					
Sotto requisiti					
001	I dati della sessione devono rimanere validi				
002					
003					

Requisito	Req-003	**Priorità**	1	**Versione**	1.0
Nome	Bilanciamento load balancer funzionante				
Note					
Sotto requisiti					
001	Gestione sessioni				
002					
003					

Requisito	Req-004	**Priorità**	1	**Versione**	1.0
Nome	Upload funzionante				
Note					
Sotto requisiti					
001					
002					
003					

Requisito	Req-005	**Priorità**	1	**Versione**	1.0
Nome	Download funzionante				
Note					
Sotto requisiti					
001					
002					
003					

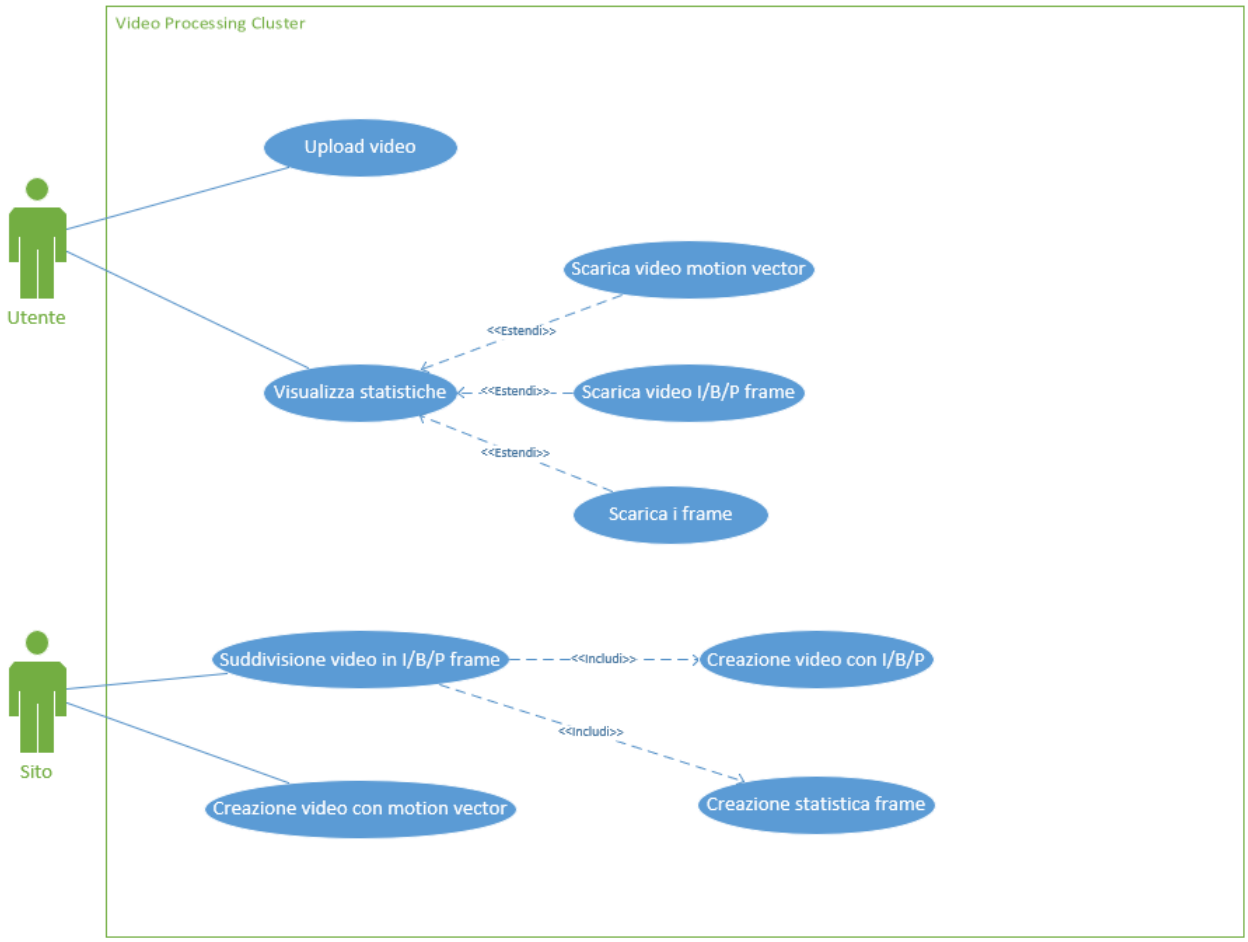
Requisito	Req-006	**Priorità**	1	**Versione**	1.0
Nome	Peso di upload massimo				
Note					
Sotto requisiti					
001					
002					
003					

Requisito	Req-007	**Priorità**	2	**Versione**	1.0
Nome	Grafico frequenza frame				
Note					
Sotto requisiti					
001					
002					
003					

Requisito	Req-008	**Priorità**	2	**Versione**	1.0
Nome	Grafico frequenza frame funzionante				
Note					
Sotto requisiti					
001					
002					
003					

Requisito	Req-009	**Priorità**	2	**Versione**	1.0
Nome	Possibilità di scaricare come immagini i frame I/B/P				
Note					
Sotto requisiti					
001					
002					
003					

2.3 Use case



2.4 Pianificazione

Le varie attività sono distribuite in modo da non sovrapporsi o dare troppo lavoro nello stesso momento. Il Gantt ha 36 righe e contiene le seguenti categorie:

1. **Analisi**: Nell'analisi sono raggruppate tutte le attività che riguardano l'analisi del progetto, necessaria per facilitare e velocizzare la progettazione, implementazione e integrazione.
2. **Progettazione**: Questo sotto capitolo contiene la struttura di rete e ricerca. Questo ci permette di avere le idee più chiare nelle fasi successive, evitando confusione o rallentamenti.
3. **Implementazione**: In questo sotto capitolo iniziano le attività di sviluppo. Grazie alle fasi precedenti è più semplice proseguire con il lavoro perché si ha già la struttura sulla quale lavorare.
4. **Integrazione**: Per ultimo ci sono i test che sono molto utili per controllare che tutto funzioni. Inoltre ci permettono di trovare degli errori non riscontrati durate l'implementazione.

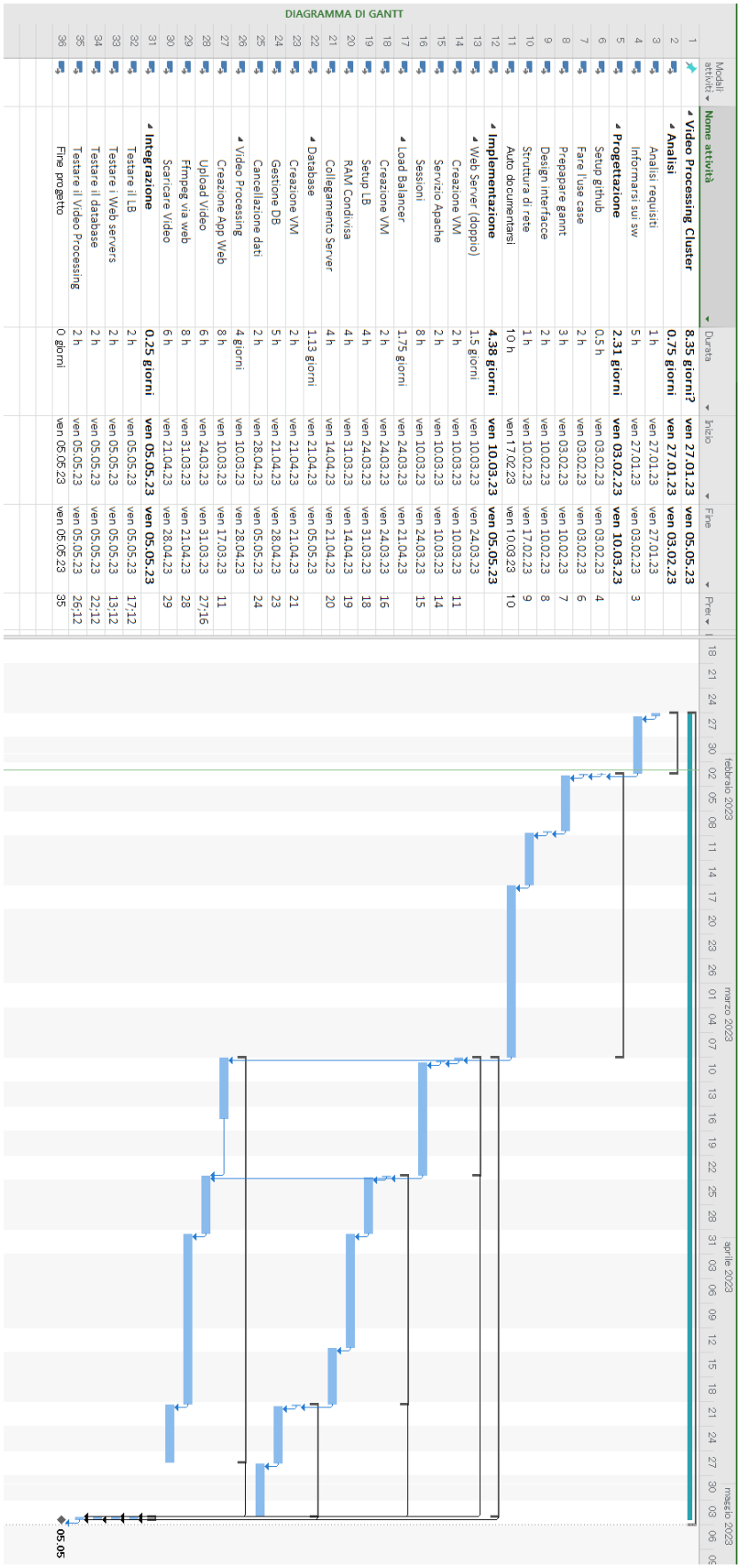


Figura 2 – Gantt Preventivo

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.

2.5.1 Software

- 4 Linux VMs
  - 2 WebServer
  - 1 Load Balancer
  - 1 Database
- PHP
- Memcached

2.5.2 Hardware

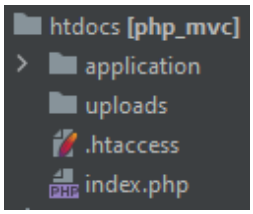
- SSD per memorizzare le Linux VMs
- PC con:
  - CPU Intel Core i7-7700
  - RAM 16 GB

3 Progettazione

3.1 Design dell'architettura del sistema

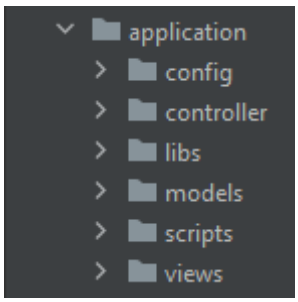
3.1.1 Struttura app web

3.1.1.1 Directory pricipale => htdocs



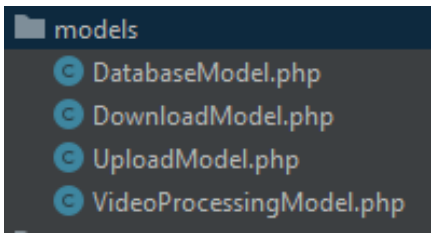
- **htdocs**: è la cartella principale dove sono contenuti i file dell'applicativo.
- **application**: è la cartella dove sono contenuti i file dell'applicazione.
- **uploads**: la cartella dove vengono messi i file video dati dall'utente.
- **.htaccess**: è un file che serve a spostare l'utente tra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

3.1.1.2 App => application



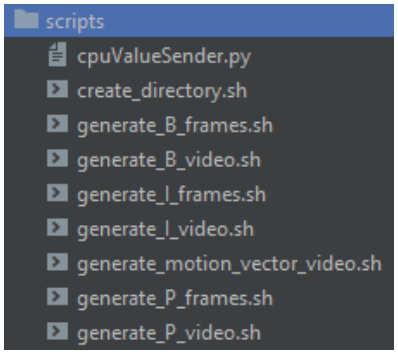
config: contiene delle configurazioni principali, i "define".
controller: contiene il file "home" che controlla il passaggio di dati.
libs: contiene le librerie, in questo caso solo application.php.
models: contiene tutte le classi di progetto.
scripts: contiene tutti gli script ffmpeg + python.
views: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

3.1.1.3 Classi => models



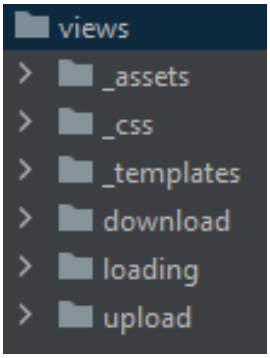
DatabaseModel: serve per collegarsi al db, serve a fare query e serve a inserire dati.
DownloadModel: serve per scaricare i files creati.
UploadModel: server per caricare il file video + controlli.
VideoProcessingModel: processa il video creando i files.

3.1.1.4 Scripting => scripts



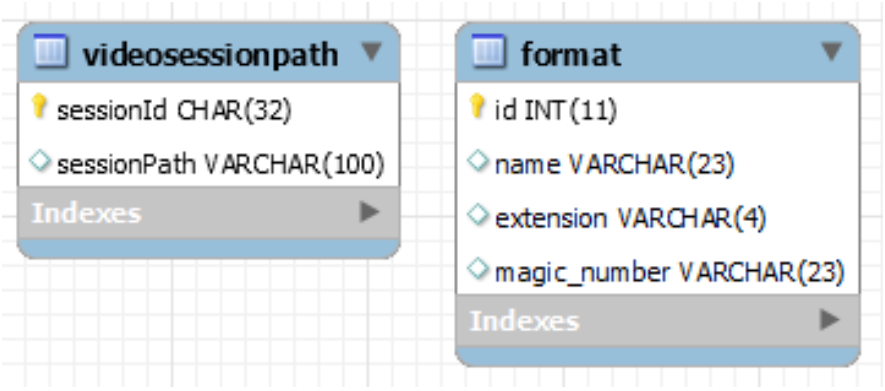
Questi scripts servono per aiutare l'applicativo.
cpuValueSender.py: manda il valore cpu al LB.
create_directory.sh: crea le varie cartelle.
generate_X_frames.sh: genera i frames X.
generate_X_video.sh: genera il video usando i frame X.
generate_motion_vector_video.sh: genera il video con i vettori visibili.

3.1.1.5 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.
loading: contiene i file per la pagina loading.

3.2 Design dei dati e database



3.3 Design delle interfacce

3.3.1 Upload

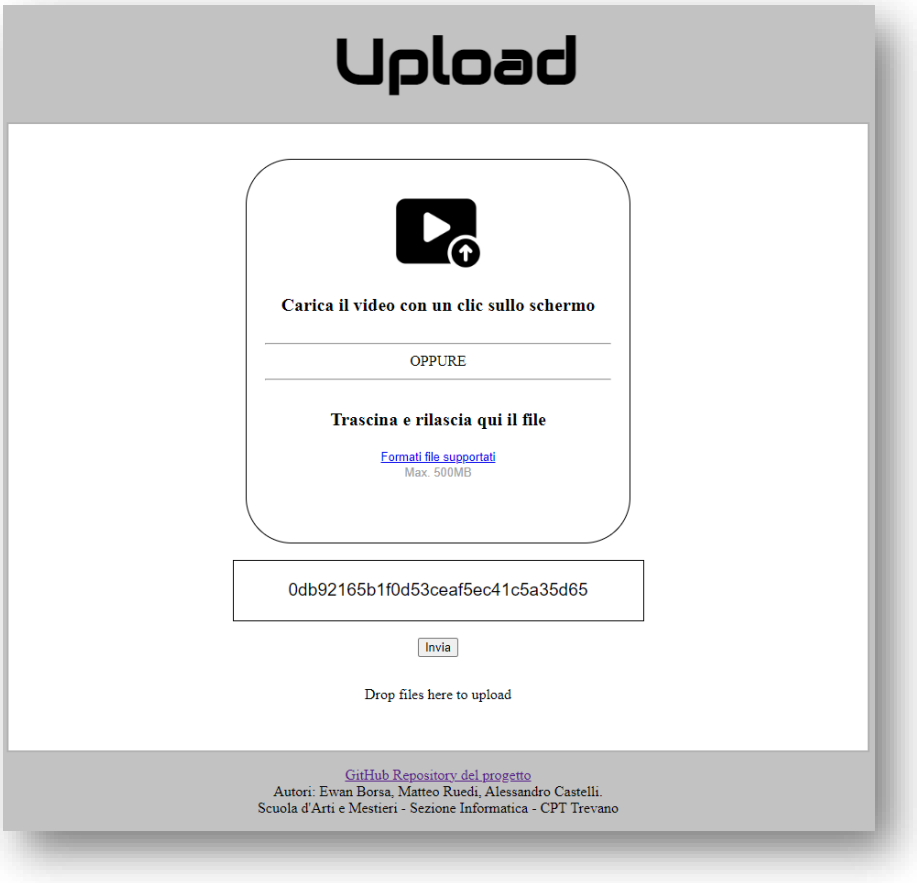


Figura 1 - Pagina di Upload

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.

3.3.2 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.

3.4 Design procedurale

Eventuale diagramma di flusso o swimlane
DA FARE

4 Implementazione

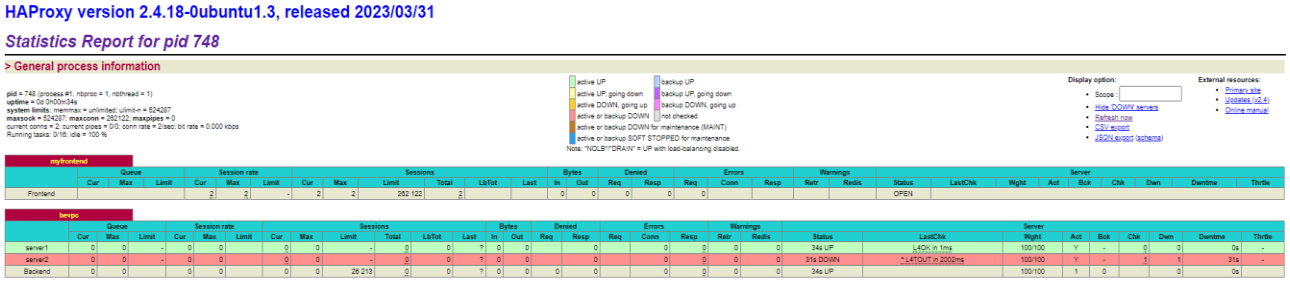
4.1 Codice

```
frontend myfrontend
        bind *:80
        default_backend bevpc
        stats uri /haproxy?stats

backend bevpc
        server server1 192.168.56.10:80 check  weight 100  agent-check  agent-addr 192.168.56.10  agent-port 9999  agent-inter 5s  agent-send ping\n
        server server2 192.168.56.20:80 check  weight 100  agent-check  agent-addr 192.168.56.20  agent-port 9998  agent-inter 5s  agent-send ping\n
        log 127.0.0.1:514 local0 debug
```

Figura 3 - File cfg haproxy

Questo codice permette al load balancer di agire con un bilanciamento del carico dato dall'utilizzo della cpu dei due backend. Grazie all'agente di haproxy e allo script presente ed eseguito in background nel backend è possibile stabilire quali dei due backend è con un carico maggiore o inferiore e in caso alzare il peso di uno e dell'altro. Inoltre è definito un percorso per accedere alle statistiche del load balancer.



Questa è la tabella di monitoraggio di haproxy. Viene data anche la possibilità di esportare i dati.

```python
import time
import psutil
import socket

# LoadBalancer IP
IP = '192.168.56.30'
# Define the port on which you want to connect
PORT = 9999

while True:
    # Create a socket object that send data
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    print('socket created')
    if s:
        try:
            s.bind(('', PORT))
            s.listen()
            print('socket listening')
            conn, addr = s.accept()
            with conn:
                print(f"Connected by {addr}")
                agent_request = conn.recv(1024).decode().strip()
                print(f'agent request: {agent_request}')

                cpu = psutil.cpu_percent(interval=1, percpu=False)
                print(f"current cpu: {cpu}%")
                response = ""
                if cpu > 10:
                    # Set server weight to half
                    response = "50%\n"
                else:
                    response = "100%\n"
                conn.sendall( response.encode(encoding="ascii",errors="ignore")  )
                print(f'weight data sent: {response.strip()}' )
                conn.close()
        except Exception as e:
            print(e)

        try:
            s.shutdown(socket.SHUT_RDWR)
            s.close()
            print('socket closed')
            pass
        except Exception as e:
            print(e)

    time.sleep(0.1)
```

Figura 4 - agent python

Questo file python viene eseguito in background sui backend ed è il responsabile dell'invio della percentuale della cpu al load balancer per il bilanciamento del carico. In questo codice viene stabilita una connessione tra backend e load balancer, una volta connessi il backend manda al server "50%\n" se la cpu è maggiore di 10 altrimenti manda "100%\n". Ciò permette al server di abbassare il peso di uno dei due backend a metà che comporta un abbassamento delle richieste che verranno indirizzate a quel backend.

4.2 Application MVC

4.3 Models(Classi)

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)

```php
class DatabaseModel
{

    private $memcache;
    private $connection;

    public function __construct()
    {
        $this->memcache = new Memcache();
        $this->memcache->addServer( host: HOST,  port: 11211);

        $this->connection = new mysqli( hostname: HOST,  username: USERNAME,  password: PASSWORD,  database: DATABASE,  port: PORT);
    }
```

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.

```php
public function insertPath($sessId, $path) : bool
{
    $this->memcache->set($sessId, $path);

    $query = "INSERT INTO videoSessionPath(sessionId, sessionPath) VALUES(".$sessId.",".$path.")";

    return (bool)$this->connection->query($query);
}
```

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

```php
public function getPath($sessId): mysqli_result|bool
{
    $result = $this->memcache->get($sessId);

    $query = "SELECT path FROM videoSessionPath WHERE sessionId = ".$sessId;

    $result = $this->connection->query($query);

    var_dump($query);

    return $result;
}
```

Il metodo "getPath" serve a chiedere il percorso attraverso l'id di sessione.

```php
function getFormatList(): array
{
    $query = "SELECT id, name, extension, signature, offset FROM format";

    $result = $this->connection->query($query);

    $formatList = $result->fetch_all( mode: MYSQLI_ASSOC);

    $result->free();

    return $formatList;
}
```

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

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)

```php
class UploadModel
{
    private $sessId;
    private $dbConn;

    public function __construct($dbConn)
    {
        $this->sessId = $this->createUUID();
        $this->dbConn = $dbConn;
    }
```

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

```php
public function uploadFile(): bool
{
    $directory = $this->sessId;
    shell_exec( command: "../scripts/create_directory.sh $directory");
    if (!empty($_FILES)) {//se ci sono files...
        $ds = DIRECTORY_SEPARATOR;
        $storeFolder = $ds . '..' . $ds . 'uploads' . $ds;
        $tempFile = $_FILES['file']['tmp_name'];
        $targetPath = dirname( path: __FILE__ ) . $storeFolder . $_GET["session_id"] . $ds;
        $targetFile =  $targetPath . $_FILES['file']['name'];
        move_uploaded_file($tempFile,$targetFile);
        return $this->dbConn->insertPath($this->sessId, $targetFile);
    }
    return false;
}
```

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.

```php
public function createUUID() {
    try {
        $data = $data ?? random_bytes( length: 16);
    } catch (Exception $e) {}
    assert( assertion: strlen($data) == 16);
    return vsprintf( format: '%s%s%s%s%s%s%s%s', str_split(bin2hex($data),  length: 4));
}
```

"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 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)

```php
class VideoProcessingModel
{
    const HEADER_SIGNATURE_LENGTH = 12;
    const MAX_FILE_SIZE = 500000000;
    private array $formats;
    private $path;
    private $dbConn;

    public function __construct($dbConn, $path)
    {
        $this->dbConn = $dbConn;
        $this->path = $path;
        $this->formats = $this->dbConn->getFormatList();
        $this->process();
    }
```

Alla creazione di questo oggetto vengono instaziati i valori e poi viene processato il tutto attraverso ffmpeg.

```php
public function process(): void
{
    $srcDest =  " " . $this->path . " " . $this->path;
    shell_exec( command: URL . "/scripts/generate_B_frames.sh" . $srcDest);
    shell_exec( command: URL . "/scripts/generate_P_frames.sh " . $srcDest);
    shell_exec( command: URL . "/scripts/generate_I_frames.sh " . $srcDest);
    shell_exec( command: URL . "/scripts/generate_B_video.sh " . $srcDest . "_VideoB");
    shell_exec( command: URL . "/scripts/generate_P_video.sh "  . $srcDest . "_VideoP");
    shell_exec( command: URL . "/scripts/generate_I_video.sh "  . $srcDest . "_VideoI");
    shell_exec( command: URL . "/scripts/generate_motion_vector_video.sh " . $srcDest . "_VideoB");
}
```

In “process()” vengono eseguiti tutti gli scripts che eseguono i comdandi ffmpeg.

```php
function checkVideo(): bool
{
    if(file_exists($this->path)){
        $file = basename($this->path);
        if(filesize($file) > $this->MAX_FILE_SIZE){
            $handle = fopen($file, mode: "rb");# Read Binary.
            $header = fread($handle, $this->HEADER_SIGNATURE_LENGTH);
            fclose($handle);
            if($this->checkFormat($header) != -1){
                return true;
            }
        }
    }
    return false;
}
```

Questo metodo serve a controllare il video

```php
function checkFormat($header){
    foreach ($this->formats as $format) {
        $signatureLength = (strlen($format["signature"])+1)/3;//Number of bytes in the signature.
        if($format["signature"] == substr($header, $format["offset"], $signatureLength)){
            return $format["id"];
        }
    }
    return -1;
}
```

Questo metodo serve a controllare specificamente il formato del video, ritorna l'id del formato, se non esiste ritorna -1.

4.4 Views(GUI)

_templates/footer.php

```
<footer>
    <p>
        <a href="https://github.com/EwanBorsa/VideoProcessingCluster">GitHub Repository del progetto</a>
        <br>Autori: Ewan Borsa, Matteo Ruedi, Alessandro Castelli.
        <br>Scuola d'Arti e Mestieri - Sezione Informatica - CPT Trevano
    </p>
</footer>
</body>
</html>
```

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

_templates/header.php

```
<!DOCTYPE html>
<html>
<head>
    <title>Video Processing Cluster</title>
    <meta charset= "UTF-8">
    <meta name= "description" content= "Questa è una page del progetto Video Proccessing Cluster">
    <meta name= "keywords" content= "video, proccessing, cluster.">
    <meta name= "author" content= "ewan.borsa; matteo.ruedi; alessandro.castelli;">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="shortcut icon" type="image/x-icon" href="application\views\_assets\processing.ico" >
    <link rel="stylesheet" href="https://unpkg.com/dropzone@5/dist/min/dropzone.min.css" type="text/css" />
    <link rel="stylesheet" href="application\views\_css\upload.css" type="text/css" >
    <link rel="stylesheet" href="application\views\_css\download.css" type="text/css" >
    <link rel="stylesheet" href="application\views\_css\default.css" type="text/css" >
    <script src="https://unpkg.com/dropzone@5/dist/min/dropzone.min.js"></script>
</head>
<body>
```

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

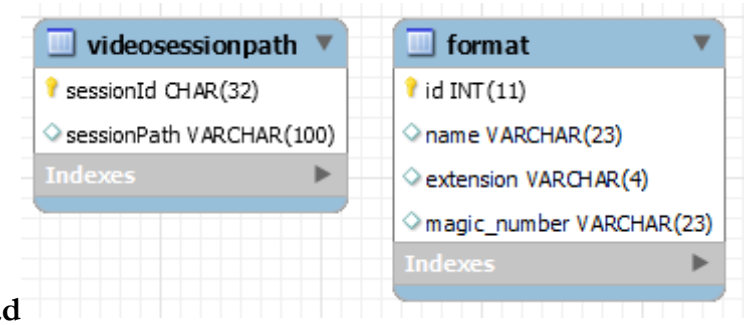
4.5 Upload

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.

```
<h1>Upload</h1>
<form action="\home\upload" class="dropzone" id="my-awesome-dropzone" method="get"><br>
    <div id="uploadCenter" class="rect-upload">
        <img src="application\views\_assets\upload.png" alt="ERR_UPLOAD_VIDEO_PNG" width="100px" style="...">
        <div class="text">Carica il video con un clic sullo schermo</div><br>
        <hr><div class="altrimenti">OPPURE</div><hr><br>
        <div class="text">Trascina e rilascia qui il file</div><br>
        <div class="formati-file">
            <a href="<?php echo URL?>\home\formatsList" target="_blank">Formati file supportati</a><br> Max. 500MB
        </div>
    </div><br>
    <div class="rect-id">
        <label class="sessId"><?php echo $sessionId ?? "ERRORE CREAZIONE ID SESSIONE" ?></php></label>
    </div><br>
    <input class="" type="submit">
</form>
```

Parte script in javascript per configurare il dropzone:

```
<script>
    Dropzone.options.myGreatDropzone = {
        maxFilesize: 500, // MB
        maxFiles: 1,
        disablePreviews: true;
    };
</script>
```



4.6 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.

```
<h1>Download</h1>
<div class="divVideo">
    <video controls class="video">
        <!--<source src="video.mp4" type="video/mp4">-->
    </video>
</div>
<div class="divButton">
    <button id="bMotionVector">Scarica video con i motion vector</button><br><br><br>

    <button id="bVideoFrames">Scarica video con i frame selezionati</button><br><br>

    <form class="videoFrames">
        <input type="checkbox" id="framesVideoI" name="framesVideoI" value="I">
        <label for="framesVideoI"> I </label>
        <input type="checkbox" id="framesVideoB" name="framesVideoB" value="B">
        <label for="framesVideoB"> B </label>
        <input type="checkbox" id="framesVideoP" name="framesVideoP" value="P">
        <label for="framesVideoP"> P </label>
    </form><br><br>

    <button id="bFrames">Scarica i frame selezionati</button> <br><br>

    <form class="frames">
        <input type="checkbox" id="framesI" name="framesI" value="I">
        <label for="framesI"> I </label>
        <input type="checkbox" id="framesB" name="framesVideoB" value="B">
        <label for="framesB"> B </label>
        <input type="checkbox" id="framesP" name="framesP" value="P">
        <label for="framesP"> P </label>
    </form>
</div>
```

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

```
<h3>Statistica dei frame del video</h3>
<div class="framePercTable">
    <table>
        <tr>
            <th>I frame</th>
            <th>B Frame</th>
            <th>P Frame</th>
        </tr>
        <tr>
            <td><?php echo $iFrameStat ?? 0 ?></td>
            <td><?php echo $bFrameStat ?? 0 ?></td>
            <td><?php echo $pFrameStat ?? 0 ?></td>
        </tr>
    </table>
</div>
<h3>Grafico dei frame del video</h3>
```

4.7 Controller ()

home.php
La classe home è il controller dell'applicazione, utilizza 5 attributi:

- UploadModel =>uploader
- VideoProcessingModel => processor
- DownloadModel => downloader
- DatabaseModel => dbConn
- String => sessId

```php
class Home
{
    private UploadModel $uploader;
    private VideoProcessingModel $processor;
    private DownloadModel $downloader;
    private DatabaseModel $dbConn;
    private string $sessId;

    public function __construct()
    {
        require "application/models/DatabaseModel.php";
        $this->dbConn = new DatabaseModel();
        require 'application/models/UploadModel.php';
        $this->uploader = new UploadModel( dbConn: $this->dbConn ?? null);
        $this->sessId = $this->uploader->getSessId();
    }
```

Nel costruttore vengono istanziati i "dbConn" per le connessioni db, "uploader" passandogli "dbConn" e infine l'id tramite un metodo che crea UUID.

```php
public function index() : void
{
    $sessionId = $this->sessId;
    require_once 'application/views/_templates/header.php';
    require_once 'application/views/upload/index.php';
    require_once 'application/views/_templates/footer.php';
}

public function formatsList(): void
{
    require_once 'application/views/_templates/header.php';
    require_once 'application/views/upload/formats.php';
    require_once 'application/views/_templates/footer.php';
}
```

I metodi "index" e "formatsList" fanno il require dei file per la GUI.

```php
public function upload() : void
{
    if($this->uploader->uploadFile()){
        $filePath = $this->dbConn->getPath($this->sessId);
        require 'application/models/VideoProcessingModel.php';
        $this->processor = new VideoProcessingModel( dbConn: $this->databaseModel ?? null, $filePath);
        $this->processor->checkVideo($filePath);
        require 'application/models/DownloadModel.php';
        $this->downloader = new DownloadModel($this->dbConn, $this->sessId);
    }else{
        require_once 'application/views/upload/error.php';
    }
}
```

Con il metodo "upload" si usa l'oggetto "uploader" per fare l'upload del file, una volta fatto si fa il controllo del video e se va benesi apre la pagina di download se no una pagina di errore.

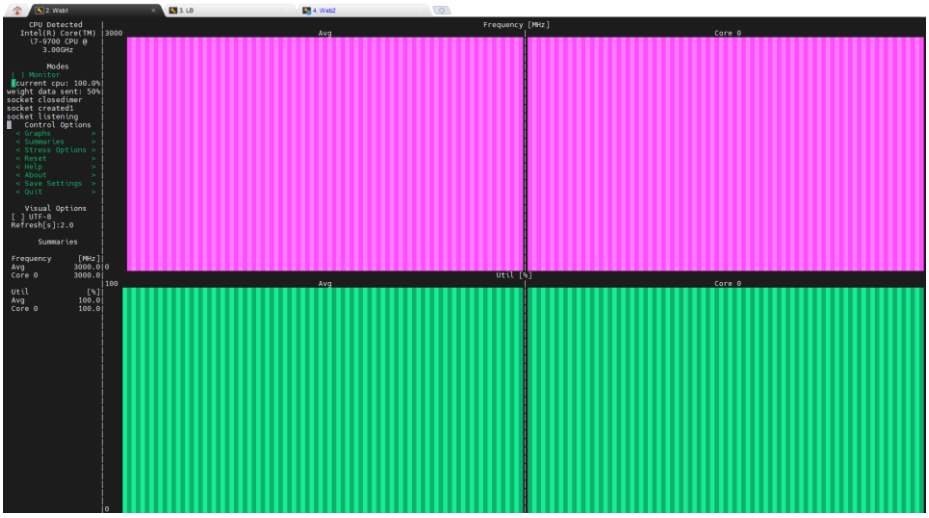
```php
public function download() : void
{
    require 'application/models/DownloadModel.php';
    $this->downloader = new DownloadModel($this->dbConn, $this->sessId);
    require 'application/models/VideoProcessingModel.php';
    $this->processor = new VideoProcessingModel($this->dbConn, $this->sessId);
    $iFrameStat = $this->processor->getFrameStat( type: "I");
    $bFrameStat = $this->processor->getFrameStat( type: "B");
    $pFrameStat = $this->processor->getFrameStat( type: "P");
    require_once 'application/views/_templates/header.php';
    require_once 'application/views/download/index.php';
    require_once 'application/views/_templates/footer.php';
}
```

Con il metodo "download" si usa l'oggetto "downloader" per fare il download dei file e vedere i dati statistici.

5 Test

5.1 Protocollo di test

All'interno dei seguenti test abbiamo utilizzato uno stess tester. Abbiamo deciso di utilizzare s-tui. Eseguendo il comando "sudo apt install stress" installiamo il software e per avviarlo basta digitare "s-tui". Tramite il bottone Stress sotto a Monitor possiamo stressare al massimo la macchina. Di seguito c'è un rappresentazione del software.



Test Case Riferimento	TC-001 REQ-001	**Nome**	Vagrant up con macchine virtuali funzionanti
Descrizione	Si controlla la creazione corretta delle macchine virtuali da parte di vagrant		
Prerequisiti	• Px attivo		
Procedura	1. Aprire il terminale con il percorso del in cui è situato il Vagrantfile. 2. Eseguire il comando: vagrant up		
Risultati attesi	Le 4 macchine virtuali vengono avviate		

Test Case Riferimento	TC-002 REQ-001	**Nome**	Sito funzionante su webserver
Descrizione	Si controlla che i due webserver abbiamo il sito funzionante		
Prerequisiti	• Vagrant up		
Procedura	1. Fare una richiesta http in un browser dell'ip del load balancer		

Risultati attesi	Viene visualizzato il sito e non apache default page

Test Case **Riferimento**	TC-003 REQ-001	**Nome**	Bilanciamento load balancer funzionante
Descrizione	Si controlla che il load balancer bilanci il carico correttamente attraverso il carico della cpu		
Prerequisiti	• Vagrant up • Stress test sui backend installati • Agent file sul valore della cpu avviato sul backend		
Procedura	2. Fare una richiesta http dell'ip del load balancer in un browser 3. Avviare lo stress(s-tui) test sul un backend 4. Visualizzare il risultato delle richieste 3. Fermare lo stress test e avviarlo sull'altra macchina		
Risultati attesi	Viene visualizzato il bilanciamento del carico in maniera correttamente		

Test Case **Riferimento**	TC-004 REQ-001	**Nome**	Database funzionante
Descrizione	Si controlla che Mysql abbia il database creato e che sia possibile inserire dati		
Prerequisiti	• Mysql installato		
Procedura	1. Accedere a Mysql 2. Eseguire il comando: show databases; 3. Fare dei Insert nelle tabelle del database;		
Risultati attesi	Il database esiste e gli Insert funzionano		

Test Case **Riferimento**	TC-005 REQ-001	**Nome**	Upload funzionante
Descrizione	Si controlla che la pagina web possa fare l'upload		
Prerequisiti			

Procedura	
Risultati attesi	

Test Case **Riferimento**	TC-006 REQ-001	**Nome**	Download funzionante
Descrizione	Si controlla che la pagina web possa fare il download		
Prerequisiti			
Procedura			
Risultati attesi			

Test Case **Riferimento**	TC-007 REQ-001	**Nome**	Peso di upload massimo
Descrizione	Si controlla che l'upload dei video è di massimo 500 MB		
Prerequisiti			
Procedura			
Risultati attesi			

Test Case **Riferimento**	TC-008 REQ-001	**Nome**	Grafico frequenza frame funzionante
Descrizione	Si controlla che funzioni il grafico di frequenza dei frame		
Prerequisiti			
Procedura			
Risultati attesi			

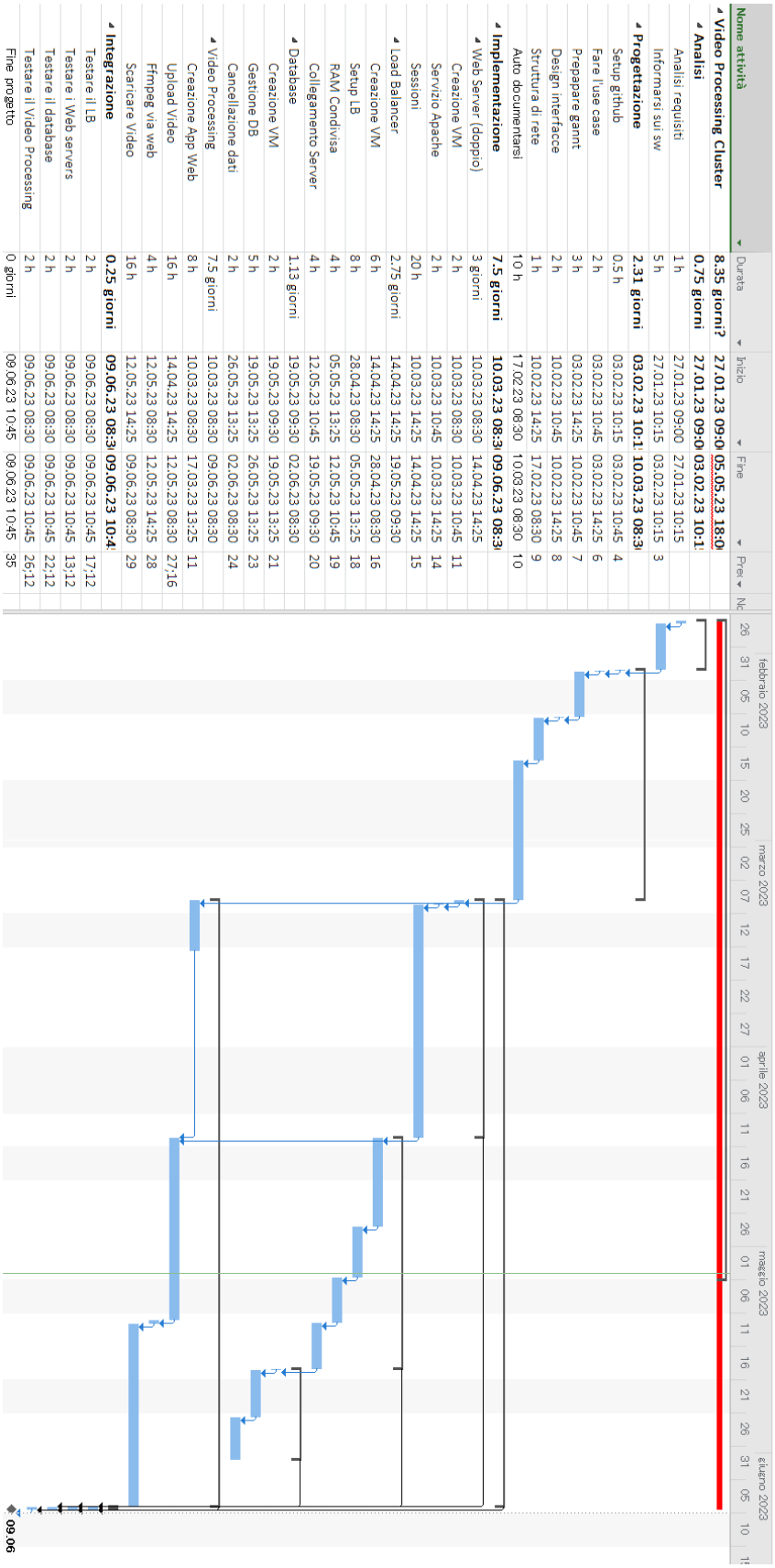
5.2 Risultati test

Test case	Esito	Risultato	Data
TC-001	**NON PASSATO**	Vagrant non crea la macchina del load balancer. La soluzione adottata è l'utilizzo della macchina virtuale creata tramite virtual box	05.05.2023
TC-002	**PASSATO**	Il sito viene visualizzato correttamente da ambe i backend e dal	05.05.2023
TC-003	**PASSATO**	Una volta avviato lo stress tester il load balancer abbassa il peso del backend stressato mandandogli una richiesta dopo averne mandate due all'altro backend. Il peso viene correttamente dimezzato.	05.05.2023
TC-004	**PASSATO**	Il database viene correttamente visualizzato con le tabelle corrette. L'inserimento di dati funziona	
TC-005	**PASSATO**	I file vengono caricati nelle apposite cartelle correttamente	
TC-006	**NON PASSATO**	A causa di imprevisti non siamo riusciti ad implementare il download	
TC-007	**PASSATO**	Aggiungendo un file con più di 500MB questo non viene caricato	
TC-008	**NON PASSATO**	A causa di imprevisti non siamo riusciti ad implementare il grafico per le statistiche	

5.3 Mancanze/limitazioni conosciute

Non siamo riusciti a causa di imprevisti a creare il grafico delle statistiche. Inoltre la cancellazione dei file, la riconnessione tramite id.

6 Consuntivo



Nome attività	Durata	Inizio	Fine	Predecessori
Video Processing Cluster	8.35 giorni?	27.01.23 09:00	05.05.23 18:00	
Analisi	0.75 giorni	27.01.23 09:00	03.02.23 10:15	
Analisi requisiti	1 h	27.01.23 09:00	27.01.23 10:15	
Informarsi sui sw	5 h	27.01.23 10:15	03.02.23 10:15	3
Progettazione	2.31 giorni	03.02.23 10:15	10.03.23 08:30	
Setup github	0.5 h	03.02.23 10:15	03.02.23 10:45	4
Fare l'use case	2 h	03.02.23 10:45	03.02.23 14:25	6
Prepapare gannt	3 h	03.02.23 14:25	10.02.23 10:45	7
Design interfacce	2 h	10.02.23 10:45	10.02.23 14:25	8
Struttura di rete	1 h	10.02.23 14:25	17.02.23 08:30	9
Auto documentarsi	10 h	17.02.23 08:30	10.03.23 08:30	10
Implementazione	7.5 giorni	10.03.23 08:30	09.06.23 08:30	
Web Server (doppio)	3 giorni	10.03.23 08:30	14.04.23 14:25	
Creazione VM	2 h	10.03.23 08:30	10.03.23 10:45	11
Servizio Apache	2 h	10.03.23 10:45	10.03.23 14:25	14
Sessioni	20 h	10.03.23 14:25	14.04.23 14:25	15
Load Balancer	2.75 giorni	14.04.23 14:25	19.05.23 09:30	
Creazione VM	6 h	14.04.23 14:25	28.04.23 08:30	16
Setup LB	8 h	28.04.23 08:30	05.05.23 13:25	18
RAM Condivisa	4 h	05.05.23 13:25	12.05.23 10:45	19
Collegamento Server	4 h	12.05.23 10:45	19.05.23 09:30	20
Database	1.13 giorni	19.05.23 09:30	02.06.23 08:30	
Creazione VM	2 h	19.05.23 09:30	19.05.23 13:25	21
Gestione DB	5 h	19.05.23 13:25	26.05.23 13:25	23
Cancellazione dati	2 h	26.05.23 13:25	02.06.23 08:30	24
Video Processing	7.5 giorni	10.03.23 08:30	09.06.23 08:30	
Creazione App Web	8 h	10.03.23 08:30	17.03.23 13:25	11
Upload Video	16 h	14.04.23 14:25	12.05.23 08:30	27;16
Ffmpeg via web	4 h	12.05.23 08:30	12.05.23 14:25	28
Scaricare Video	16 h	12.05.23 14:25	09.06.23 08:30	29
Integrazione	0.25 giorni	09.06.23 08:30	09.06.23 10:45	
Testare il LB	2 h	09.06.23 08:30	09.06.23 10:45	17;12
Testare i Web servers	2 h	09.06.23 08:30	09.06.23 10:45	13;12
Testare il database	2 h	09.06.23 08:30	09.06.23 10:45	22;12
Testare il Video Processing	2 h	09.06.23 08:30	09.06.23 10:45	26;12
Fine progetto	0 giorni	09.06.23 10:45	09.06.23 10:45	35

7 Conclusioni

7.1 Sviluppi futuri

7.2 Considerazioni personali

7.2.1 Matteo Ruedi

Questo progetto è stato molto interessante. Non siamo riusciti a finirlo tutto nonostante i nostri sforzi. Durante il percorso abbiamo riscontrato molti problemi che ci hanno ostacolati per arrivare a fine progetto. Grazie a questo progetto ho imparato a lavorare in gruppo e a gestire un gruppo. La comunicazione è stata fondamentale e la suddivisione del progetto è stata fatta bene. Sono comunque orgoglioso del lavoro svolto in squadra.

7.2.2 Ewan Borsa

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.

7.2.3 Alessandro castelli

Il nostro progetto di gruppo è stata una buona occasione per imparare a lavorare meglio di squadra. Non siamo riusciti a raggiungere gli obiettivi prefissati, a causa di molti imprevisti. Questo progetto ci ha permesso di acquisire nuove competenze e di migliorare la nostra attitudine al lavoro. Siamo orgogliosi del lavoro svolto e abbiamo imparato l'importanza dell'organizzazione e della comunicazione per raggiungere un obiettivo comune.

8 Bibliografia

8.1 Sitografia

- https://app.moqups.com/, *Moqups*, 27-01-2023.
- https://www.laratutorials.com/drag-and-drop-file-upload-using-dropzonejs-php-mysql, *Drage&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://www.haproxy.com/documentation/aloha/latest/load-balancing/health-checks/agent-checks/, *Load Balancing - Agent Check*, 17-03-2023.
- http://docs.haproxy.org/, HaProxy doc, 24-03-2023.
- https://filesamples.com/, *File Samples*, 17-03-2023.
- https://www.garykessler.net/library/file_sigs.html/, *Files Signature*, 17-03-2023.
- https://en.wikipedia.org/wiki/List_of_file_signatures, *Files Signature Wikipedia*, 17-03-2023.
- https://docs.python.org/3/library/socket.html, *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://memcached.org/about, *Memcached Informations*, 31-03-2023.
  https://www.tutorialspoint.com/memcached/memcached_environment.htm, *Memcached Environment*, 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.*
- https://www.digitalocean.com/community/tutorials/how-to-install-and-secure-memcached-on-ubuntu-20-04#step-1-installing-memcached, *Installing Memecached*, 28-04-2023.

9 Indice delle figure

10 Allegati

Elenco degli allegati, esempio:

- QdC
- Diari di lavoro
- Manuale utente
- Abstract
- Codice sorgente