# VES Project

Design document

Filippo Solimando

# Table of Contents

# General design

## 1.1 System architecture

The project is to be realized as a 3-tier application. The division in tiers reflects this schema:

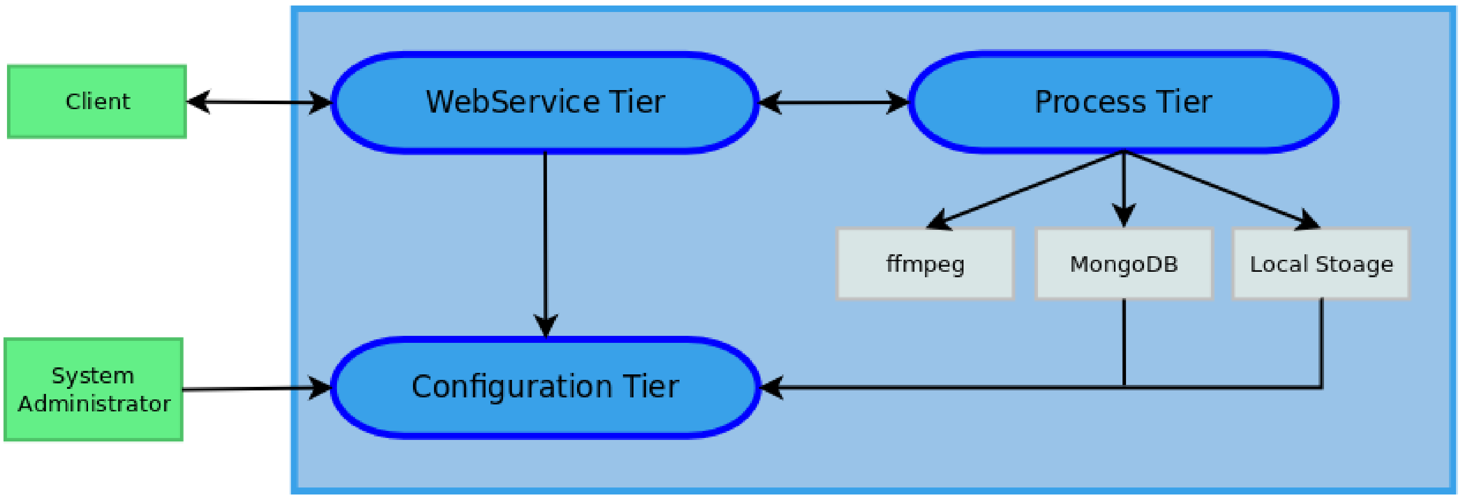
- **Process tier**: every video editing request is identified by a session. The session contains all the reference of the client's resources and the video process status. Client's resources are video and subtitles files and resize settings.
- **WebService tier**: handle http requests and routes them to the appropriate action. This layer mediate the interaction between the client and the core application.
- **Configuration tier**: responsable to guarantee the access to all the managed resources for the business login.

The application will be distributed into a compact Dockers Container Image, Configuration Tier will be the tool with an administrator can configure the single VES service into a given service farm.

## 1.2 Technology specification

The identified tiers defines the main design choice of the project. The following list of subsystems can provide more details about the implementation architecture:

- **Client-VES interaction**: the project is designed as a RESTful service, Java JAX-RS technology will route to the specified functionality.
- **Entities**: the session is only entity defined, it contains the process request identity, process status and all the resource dependency to complete the video editing operation.
- **Database**: a MongoDB service will provide a full management of the session entity.
- **Storage**: Video and subtitle files will be stored into local file system. This subsystem will check file system availability and manage the access and the operations to the file resources.
- **Video editing**: it will be guarantee by the ffmpeg library (http://www.ffmpeg.org)



# Process tier

## 2.1 Session
Session is the entity that define an entire video editing process, from the resource gathering to the final video file generation. Session contains:

- Univoque ID
- Last modify timestamp, used to detect expired and unresolved requests to delete
- Video file, stored in local storage into a path returned by Storage subsystem
- Subtitle files, stored in local storage into a path returned by Storage subsystem
- Resize parameters: width and height
- Process status:
  - Waiting for resource
  - Ready to burn video
  - Video burn in process
  - Burn completed, video result available
  - Burn interrupted
- Video editing asynchonous thread

Session is also responsable of current internal functionalities:

- Save client's uploaded file into local storage, interacting with Storage subsystem to define file path.
- Save every parameter changes interacting with Database subsystem
- Start video editing process into a indipendent thread, by executing ffmpeg tool process
- Register video editing process' progress into session
- Detect when video editing ended and make it available to download eventually
- Delete database record and local file resource as soon as the client recevied the operation result, wether the video editing failed or not

## 2.2 Storage
Storage subsystem will be initialized by the Configuration Tier the local file system location to store client's file. On a session request, Storage subsystem will generate and univoque folder to store session's file resources.
Also it will return a list of all session's folder for the Session Restore process

## 2.3 Database
This subsystem, after initialized by Configuration Tier, will store session entity.

## 2.4 Video editing process
This process will be executed by the session after a client's burn request and when all mandatory resource are declared.
Mandatory resource are:

- A video file to process
- A subtitle file or a resize request or both

To start a video process, the system will start a new thread where a ffmpeg process will be executed. ffmpeg process' output will be redirect and stored into session parameter to track the current editing progress. The client can access to this information by the specific request.

## 2.5 Session restore
On the application start, all the incompleted sessions will be restored into the system, available to complete the video editing process. Incomplete sessions could happen if the service crashed or as been turned off before complete the burn process or the resulting video file has not been downloaded yet.
A session, to be restored, must has the following features:

- Exists on the database

- All file resource exists
- The status is "Ready to burn video", "Burn in progress", "Burn completed"

Every other session and its resources will be deleted.

### 2.6 Garbage collection

In progress

# WebService Tier

### 3.1 RESTful API reference

In progress

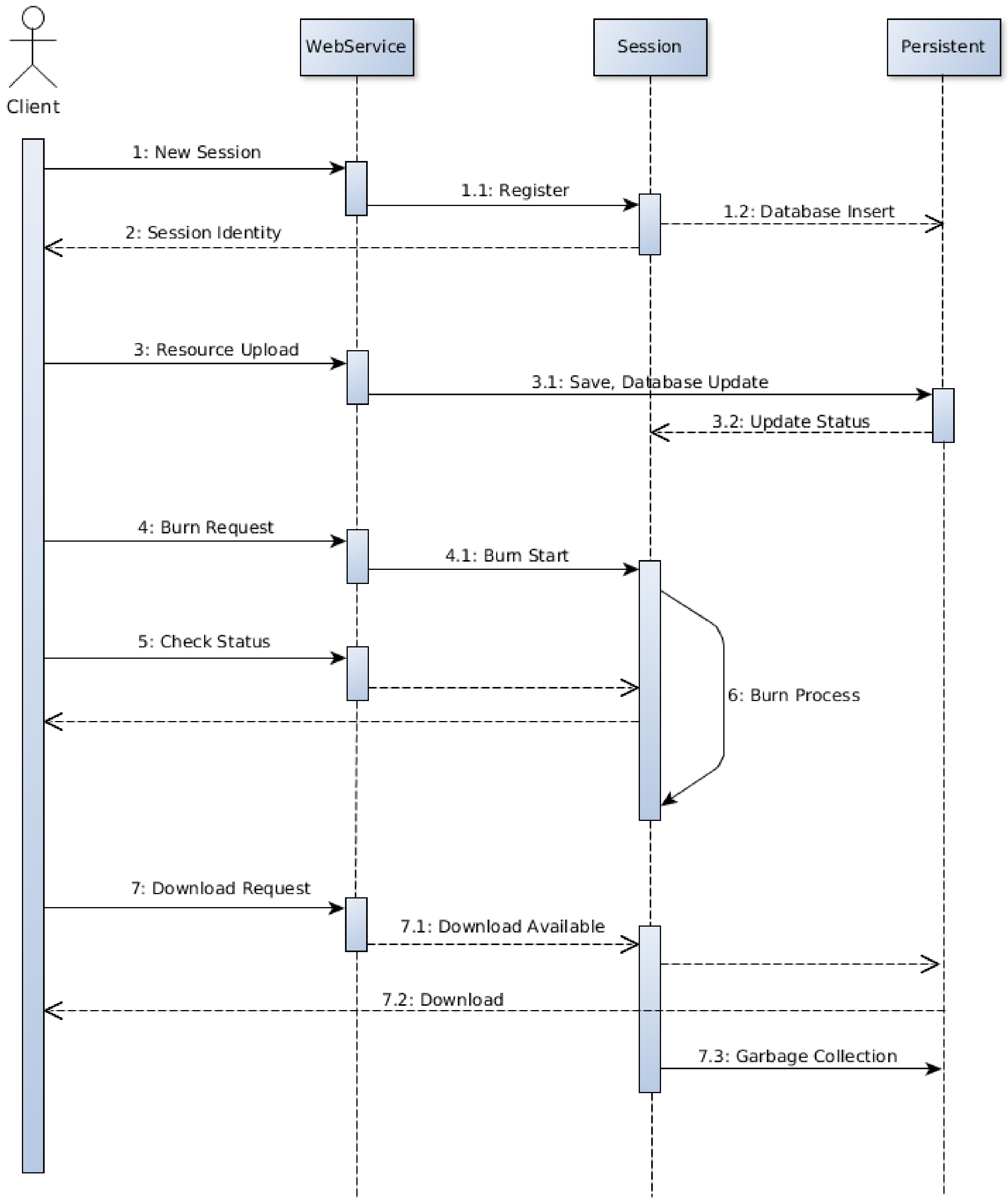
# Configuration Tier

In progress

# High Level Diagram

## 5.1 System Sequence Diagram

In the following diagram, we refer as Persistent the group formed by Database subsystem and Storage subsystem.



## 5.2 Package Diagram

In progress