PapyrOS: Architecture specification

Leia Media: PapyrOS -

Architecture Specification

Created by: Ulf Byskov

Creation date: 26.03.2013

Document details

Document Owner:	Ulf Byskov
Version number:	1.0
Date created:	26.03.2013
Last modified and by whom:	
Status of document:	Draft
Confidentiality:	Company: Leia Media, Codemate

Table of Contents

Change History

-

Scope

This document specifies the architecture of the PapyrOS. It will explain the purpose of the high level components of the system and how external systems interact with them. External systems are LeiaME (Leia Media Extender) and possible others. Although these systems may be mentioned they will otherwise not be described in this document. Each external system developed by Codemate will have its own specification document.

Summary

PapyrOS is a system used to prepare media content to be pushed to *Leia Media Viewer* devices.

The system is used to maintain

- media content
- users and subscriptions of media content
- provisioning (enable services to end users) of *Leia Media Viewer* devices

The system will provide a REST API to be used by instances of the LeiaME (Leia Media Extender) application.

LeiaME is an end user application that will act as a bridge between PapyrOS and *Leia Media Viewer* devices in order to publish content on the devices. The application will be run on devices (e.g. Android phones) owned by the end user.

The LeiaME application is not a part the PapyrOS implementation.

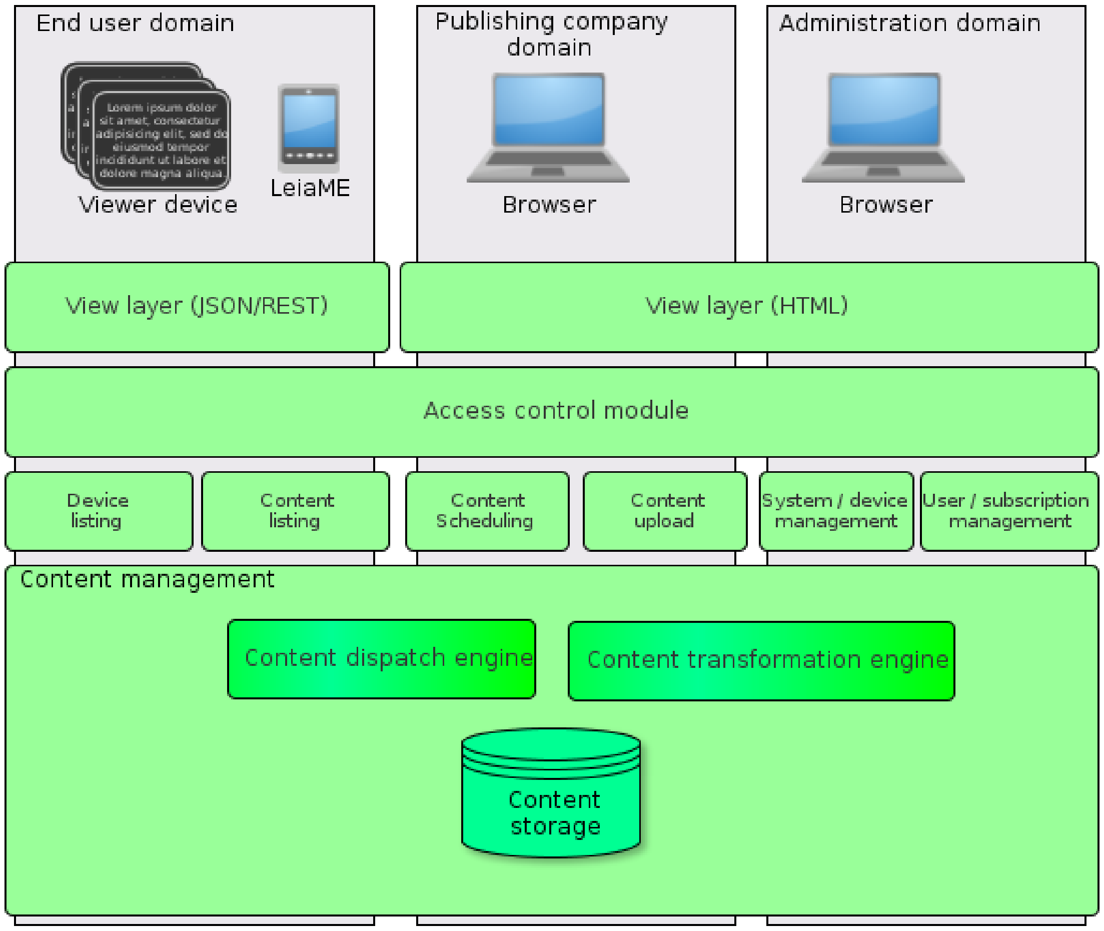
Requirements

- Requirement specification PapyrOS

System overview

General

The following diagram shows the high level components and what external systems interact with them. The external systems are grouped by the domains in which they are used.



Source (yED): system_overview.graphml

Domains

Domain	Description
End user	The end user is the primary target of the solution as he/she is the consumer of the media content being published. He/she will be in possession of one or more *Leia Media Viewer* devices and will use the LeiaME application to subscribe to media content and update his/her devices according to the subscription(s).
Publishing company	A publishing company is a Leia Media client who produces content to be published. Through a web UI a publishing company can upload media content to be distributed to end users according to their subscriptions.
Administration	Administrators are managing the users, their subscriptions and devices. Depending on client agreements administrators may come from both Leia Media and the client organisations.

Components

Component	Description

View layer (JSON/REST)	The JSON view layer is the interface used by external systems to interact with PapyrOS. The layer exposes an API based on the REST principle. Requests and responses sent over the interface shall be encoded in the JSON format and transported using the HTTP protocol.
View layer (HTML)	This view layer provides a standard HTML over HTTP interface to be used by web browsers. This interface is used for personal interactions (by publishing companies and administrators) with PapyrOS.
Access control module	To control access to the various functionalities of PapyrOS all interaction with the system will need to be authenticated. Any person or devices accessing the system will need to provide a username and a password. Predetermined user roles (assigned when users accounts are created) decide what parts of the system can be accessed.
User management	This component enables users with Administrator role to create other users and assign them appropriate roles.
Subscription Management	*LeiaME/Leia Media Viewer* users can subscribe to various media content. This component manages subscription types and per user subscriptions. **Note: This functionality is unclear and must be discussed with Leia media.**
Device management	This component enables a PapyrOS administrator to see existing *Leia Media Viewer* devices and what subscriptions are mapped to them. New device/subscription mappings can also be added. **Note: This functionality is unclear and must be discussed with Leia media.**
System management	Log files, caches, database(s), etc are maintained through this component
Content upload	Publishers are able to upload content and make it available for subscriptions.
Content Scheduling	This component enables publishers to make media content subscribable at specific dates.
Device listing	End users can see the devices he/she has acquired
Content listing	End users can see and manage the media content he/she has subscribed to
Content dispatch engine	To access uploaded content media through the PapyrOS view layers this component will locate requested media and construct URLs for downloading it.
Content transformation engine	When content media has been uploaded (as PDF) it will be converted to images suitable for the *Leia Media VIewer* and stored to the content storage.
Content storage	The content storage contains all uploaded and derived files (from the uploaded PDFs).

Actors

This section describes who is using and what areas of PapyrOS they can access. The PapyrOS *access control module* and *user management* component is derived from this description. In this architecture there is a direct mapping between actors and system roles. Using roles means that every PapyrOS user can "act" in one or more of listed roles (see below). One exception to this is that users having the *LeiaME user* role cannot have additional roles or change to another role.

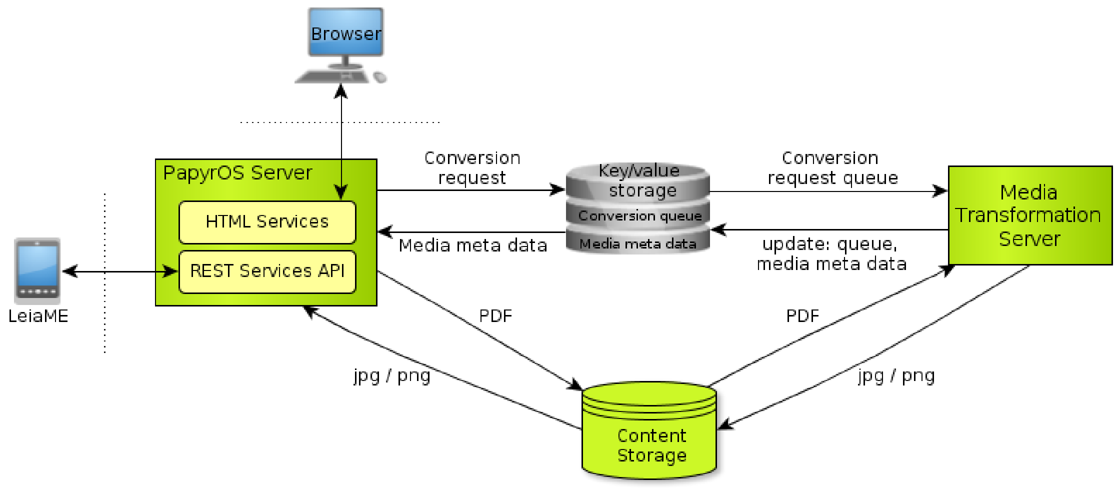
Domain	Actor / role	Description
End user	Leia Media Viewer user	A physical user accessing the Leia Media Viewer device. This does not involve a user role in PapyrOS.
	LeiaME device	A device having an id that is also found as an entry in the PapyrOS device list. When viewer devices are paired with a LeiaME device, the device will act as a proxy and update the viewer devices according to the subscriptions involving the viewer devices. This action will be performed regardless of who owns the LeiaME device and application. This does not involve a user role in PapyrOS.

	LeiaME user	A user authenticated through the *PapyrOS* back-end (JSON/REST view layer). **Note:** Having a viewer device does not require you to have a LeiaME user account. In such a scenario the LeiaME device can only act as a proxy (See *LeiaME device* actor) This user can: • Upload content to the LeiaME device (which will be synced to *PapyrOS*) • Download content synched with *PapyrOS* to the LeiaME device and deliver it to viewer devices • Receive list of allowed devices • Update selected viewer devices with selected content • Receive information of new content
Publishing company	Power user	A user authenticated through the PapyrOS Web UI (HTML view layer). This user can: • Create and manage *LeiaME* users • Create and manage *Regular* users • Create and manage subscription and *Leia Media Viewer* device mappings
	Regular user	A user authenticated through the PapyrOS Web UI (HTML view layer). This user can: • Upload content to PapyrOS (pdf format) • Create and manage media content delivery schedules • Add/Remove media content *Leia Media Viewer* devices (addition/removal will occur when devices are updated).
Administration	Site administrator user	A user authenticated through the PapyrOS Web UI (HTML view layer). This user can: • Upgrade and manage the PapyrOS installation
	General administrator	A user authenticated through the PapyrOS Web UI (HTML view layer). This user can: • Introduce *Leia Media Viewer* devices to the system • Map *Leia Media Viewer* devices to the publishing companies
	Internal administrator	A user authenticated through the PapyrOS Web UI (HTML view layer). This user can: • Create, manage and remove Publishing companies allowed to access PapyrOS • Introduce Power users, Regular users and General administrators to the system

Design

Deployment environment

The following diagram shows the core system of the solution and other systems that interact with it.



Scalability and high availability

The core system was designed with scalability and high availability in mind.

With minor modifications a deployment can utilize multiple instances of the PapyrOS Server, Key/value storage, Media transformation server and Content storage. Thus scalability is a matter of distributing a sufficient number of instances on a sufficient number of hosts. The distribution also lowers the load on each instance/host in use making sure they will be available (with minimal delays) to internal and external systems.

Note: The goal of the first implementation of PapyrOS is to deploy it to a single host environment or to an environment having only one instance of each server/storage component. The implementation will most likely reflect this goal, meaning that a deployment with multiple instances may require minor architecture and implementation changes.

Software components

All the software will run in a Linux/Unix environment.

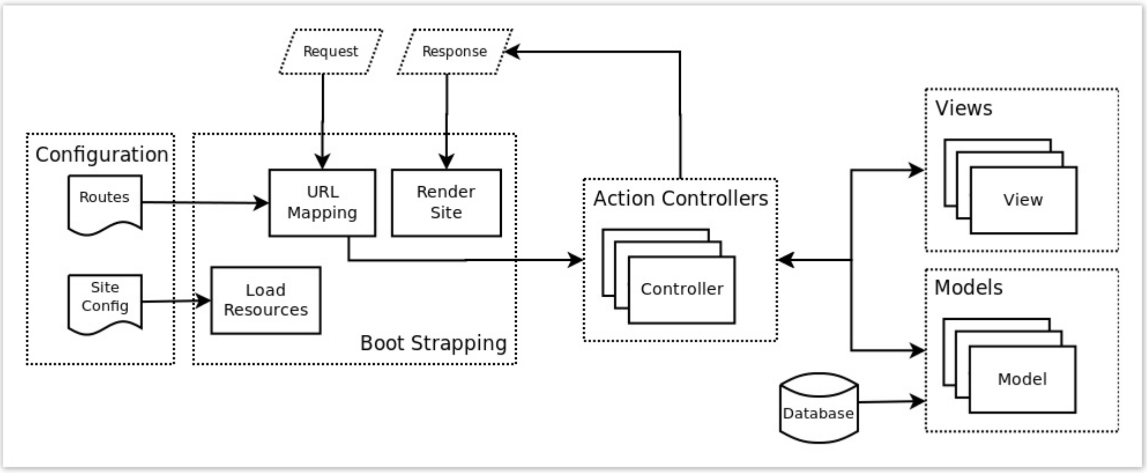
The following table lists the entities from the deployment diagram (external entities excluded), what their main purposes are and what technology will be used to implement the service provided by each entity.

System entity	Description	Selected technology
PapyrOS Server	Provides the interfaces for the view layers using HTML/JSON over HTTP. Media conversion and storage is initiated by user/LeiaME interaction with the server.	Apache, PHP 5.4, Zend Framework 2, PostgreSQL 9.2
Key/value storage	A storage used to maintain a queue of media to be converted by the *Media Transformation Server*. It also has a register of every file present in the *Content Storage*.	Zookeeper 3.x
Media Transformation Server	This entity is a server/daemon that monitors the Key/value storage conversion queue. It converts the media listed in the queue and maintains meta data related to the converted media.	Java 7, GraphicsMagick 1.3
Content Storage	The content storage contains the files (PDFs) uploaded by browser users and the images (jpg/png) derived from them through conversion	File system (directories and files)

PapyrOS Server - Software structure

The *PapyrOS Server* is implemented using the MVC architectural pattern. As mentioned in the *Software components* section the *Zend Framework 2* is used. This framework is well suited for implementing according to the MVC pattern.

The following diagram shows the MVC and other code modules used in the application.



Boot Strapping

Applications written and running on the Apache + PHP combination are stateless meaning that every time a request is received all objects and variables need to be loaded to memory. State is introduced to *PapyrOS* by keeping *long lasting* data in configuration files and in the database. *Short lasting* data is saved in the web servers session (memory).

When a request is received the boot strapping code loads site specific configuration data from files. This data includes:

- Database connection information
- Routing rules (a mapping of URLs to specific functionality)

The most important boot strapping task is to identify and run action controllers. When an action controller has been identified it takes over execution and the boot strapping code "waits" for any content it may return.
The resulting content send back as a response to the the end user (browser).

MVC: Action Controllers

Common for all action controllers is that they control what should be displayed in the browser. Data to be displayed is fetched from the models and pushed to the view (presentation layer). The boot strapping maps the current URL to the correct action controller. Before any action controller does it's "job" it is checked if the current user has permissions to perform the action.

MVC: Views

A view defines how data will be displayed to the end user. It is a dynamic HTML file that contains data related to specific contexts (users, devices, media). It is dynamic because it may show different data (different values from the same context) depending on what it's corresponding action controller may send to it. The data to be shown mostly originates from the model.

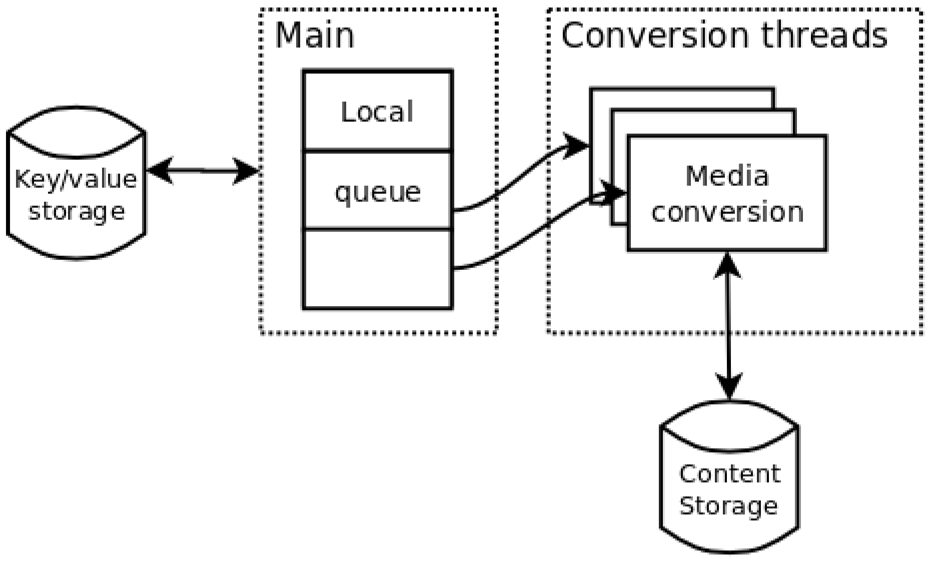
MVC: Models and Persistence

- **Models**
  - The models manages the behavior and data of the system. Most of the business logic is implemented in the models.

- **Persistence**
  - To access the data sources (tables) a database adapter is used. The adapter makes it easy to fetch and save (by query) data from/to the database and change database vendor (if needed).
  - When applicable an ORM (object relational mapping) module will be used. If complicated database table manipulations are needed, an ORM may not suitable and SQL statements will have to be written in the model code.
  - A file system cache will prevent queries from exhausting the database.

Media Transformation Server - Software structure

This server uses a simple thread model to handle media conversions and media meta data.



Thread: Main

This thread is monitoring the K*ey/value storage* for new conversion jobs to perform. The *Key/value storage* will provide necessary meta data such as the names and locations of the files to be converted. When the server has capacity (defined by a configurable number of maximum threads) to perform new conversions it will add jobs to a locally maintained queue and spawn conversion threads to do the actual conversions. After a conversion has been performed the server job queue as well as the job queue and meta data in the *key/value storage* will be updated.

Thread: Media conversion

A conversion thread will attempt to convert an incoming file and move it to the Content Storage. After report the conversion result to the Main thread the conversion thread will be terminated.

Entity model

This section describes the main entities used in the system.

Note: These do not directly reflect the data schemas used in the *PapyrOS Server* database, *Key/value storage* or *Content Storage*.

To be described further after discussing with Leia Media.

- Key/value storage
  - queue
  - media meta data
- Content storage
  - path name
  - file name
- PapyrOS Server database
  - User
  - Device
  - LeiaME
    - ?
  - Media
    - edition
  - Subscription