This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement n: 761999

EasyTV: Easing the access of Europeans with disabilities to converging media and content.

D5.3 Mid-term report on the set up and implementation of the EasyTV Service Registry and Catalogue

EasyTV Project

H2020. ICT-19-2017 Media and content convergence. – IA Innovation action.

Grant Agreement n°: 761999

Start date of project: 1 Oct. 2017

Duration: 30 months

Document. ref.: D5.3

Disclaimer

The EasyTV Consortium consists of the following partners:

	Partner Name	Short name	Country
1	Universidad Politécnica de Madrid	UPM	ES
2	Engineering Ingegneria Informatica S.P.A.	ENG	IT
3	Centre for Research and Technology Hellas/Information Technologies Institute	CERTH	GR
4	Mediavoice SRL	MV	IT
5	Universitat Autònoma Barcelona	UAB	ES
6	Corporació Catalana de Mitjans Audiovisuals SA	CCMA	ES
7	ARX.NET SA	ARX	GR
8	Fundación Confederación Nacional Sordos España para la supresión de barreras de comunicación	FCNSE	ES
9	Unione Italiana dei ciechi e degli ipovedenti	UICI	IT

PROGRAMME NAME:	H2020. ICT-19-2017 Media and content convergence - IA Innovation action
PROJECT NUMBER:	761999
PROJECT TITLE:	EASYTV
RESPONSIBLE UNIT:	ENG
INVOLVED UNITS:	ENG, UPM, MV, CERTH
DOCUMENT NUMBER:	D5.3
DOCUMENT TITLE:	Mid-term report on the set up and implementation of the EasyTV Service Registry and Catalogue
WORK-PACKAGE:	WP5
DELIVERABLE TYPE:	Report
CONTRACTUAL DATE OF DELIVERY:	31-10-2018
LAST UPDATE:	30-10-2018
DISTRIBUTION LEVEL:	PU

Distribution level:

PU = *Public,*

RE = *Restricted to a group of the specified Consortium,*

PP = *Restricted to other program participants (including Commission Services),*

CO = *Confidential, only for members of the LASIE Consortium (including the Commission Services)*

Document History

VERSION	DATE	STATUS	AUTHORS, REVIEWER	DESCRIPTION
v.0.1	04/10/2018	Draft	ENG	Table of Contents definition and document structure
v.0.2	23/10/2018	Draft	ENG, UPM	First draft
v.0.3	30/10/2018	Draft	ENG, CERTH, MV	Filled all the empty sections; Integrated revisions and suggestions by internal reviewer (MV and CERTH); Version shared for the second review (UPM).

Definitions, Acronyms and Abbreviations

ACRONYMS / ABBREVIATIONS	DESCRIPTION
ATAG	Authoring Tool Accessibility Guidelines
AWS	Amazon Web Services
CSS	Cascading Style Sheets
ECR	Elastic Container Registry
EU	European Union
GCR	Google Container Registry
GCS	Google Cloud Service
HTML	HyperText Markup Language
IAM	Identity and Access Management
JS	JavaScript
LDAP	Lightweight Directory Access Protocol
NGO	Non-Governmental Organization
SAML	Security Assertion Markup Language
UAAG	User Agent Accessibility Guidelines
UI	User Interface
UX	User Experience
W3C	World Wide Web Consortium
WAI	Web Accessibility Initiative
WCAG	Web Content Accessibility Guidelines
WP	Work Package

Table of Contents

Summary

List of Figures

List of Tables

1. EXECUTIVE SUMMARY

The aim of the document is reporting the activities conducted during the last 6 months (M6-M12) in reference to the set up and implementation of the EasyTV Service Registry and Catalogue.

These assets are described as follows[1]:

a) the EasyTV Service Registry: a container as a Service (CaaS) environment to collect the EasyTV services "packaged" in the form of "images"; furthermore, this CaaS environment will enable external developers to build applications in a self-service manner and select from image content approved for use by the IT Operation team. External developers can then use these images to create new applications, quickly and securely.

b) the EasyTV Service Catalogue: a web-based catalogue where the final user can choose the services of his/her own interest among those running in the EasyTV multi terminal technical platform.

Consistently, the deliverable has been divided into two different sections:

1. **EasyTV Service Registry**:
    a. description and definition of all the constituent elements of the registry;
    b. presentation of a wide overview of Docker Registries, explaining how a register works and show a research about what platforms are available on the market and which are their functionalities, advantages and disadvantages.
    c. focus on a specific registry: Docker Registry

2. **EasyTV Service Catalogue**:
    a. preliminary analysis and evaluation of the EasyTV Service Catalogue's design and scope, considering the intermediate status of the whole EasyTV project;
    b. presentation of the main current references regarding Web Accessibility. This corpus of guideline and inspiration examples will lead the design and implementation of the EasyTV Service Catalogue;
    c. overview of global design trends to evaluate the proper design approach and/or methodology taking into account the main EASYTV system requirements;
    d. identification of solutions to adopt in the development phase, during which the implementation of the Service Catalogue project will be arranged.

The present "Mid-term report" is relevant to WP5 and specifically to Task 5.3: it must be considered as a starting point and a guidance for activities that will lead to the actual set up and implementation of the EasyTV Service Registry and Catalogue (D5.8 *Final report* - M25).

[1] EasyTV Proposal p.54

2. INTRODUCTION

Service Registry and Catalogue are part of the EasyTV core components together with the multiterminal technical platform and the Service Development Kit.

EasyTV technical platform, and activities regarding services implementation and integration, are aimed at improving access services, enhancing interaction and personalizing services for people with disabilities.

In fact, the multi-terminal technical platform will contain the different services derived from WP2, WP3 and WP4 which will be implemented for facilitating the access of users with disabilities to audio and video content, a multilingual crowdsourcing sign language platform and repository.

Service Registry and Catalogue will be crucial for enabling presentation of that services (and new further services) to the final users, allowing them to choose services that perfectly fit their needs.

Furthermore, external developers can develop, test and deploy brand new services using the EasyTV Service Development Kit in conjunction with the EasyTV Service Registry and Service Catalogue.

2.1. Purpose and scope

The report D5.3 is part of Work Package (WP) 5 aimed to achieve the integration of the four pillars that EasyTV component-based system is going to be built on:

- a **multi-terminal technical platform** which is a container for the deployment and the optimization of the EasyTV services needed to make broadcaster contents "accessible" to users with disabilities;
- group of **platform-based service components** which are the outputs of WP2, WP3 and WP4;
- **service registry and catalogue** which allows a user to choose the service that perfectly fits his/her needs;
- a **service development kit** which allows third-party developers/"external" software companies to include brand "new" services in the EasyTV multi-terminal technical platform[2].

Specifically, the document is part of *Task 5.3 - EasyTV Service Registry and Catalogue* that includes all activities foreseen in the project for the set up and the implementation of two main assets of the EasyTV system:

- Service Registry
- Service Catalogue

The Task 5.3 is composed by a two-step strategy for reporting the activities conducted to achieve this goal:

1. Mid-term report;
2. Final report.

The present “Mid-term” report presents an exposition of the main considerations and evidences regarding the design and future implementation of Service Registry and Catalogue emerged in the last semester of the first year of EasyTV project (M6-M12).

The subsequent “Final report”, will consolidate and continue the activities of analysis started and reported in the mid-term documentation, and describe how Task 5.3’s goals will be achieved.

2.2. Relation to other tasks

This document should be considered in directly conjunction with other deliverables:

- it considers evidences from WPs’ tasks and deliverables so far achieved, especially tasks

[2] Easy tv proposal p. 54

related to the EasyTV system's requirements and architecture as:
    - WP1 - Requirements, specification and technical architecture
        - Task 1.1 – End user requirements gathering;
        - Task 1.2 – EasyTV system requirements specification;
        - Task 1.3 - Technical architecture development;

- It is strictly related to the development of all the other tasks of the WP5
    - Task 5.1 – Multi terminal technical platform to operate the EasyTV services;
    - Task 5.2 – Creation of the multilingual crowdsourcing sign language platform and repository;
    - Task 5.4 - EasyTV Service Development Kit;
    - Task 5.5 - Integration and technical testing.

- it is considered as a guidance for other activities, especially for the D5.8 *Final report on the set up and implementation of the EasyTV Service Registry and Catalogue* (M25).

3. EASYTV SERVICE REGISTRY

3.1. Introduction

When developing any software application and run it in a local computer, everything works fine, but when this application is delivered to a client and they try to run it in a different environment, is very common that the application doesn't work in the first place and usually takes a lot of effort and time to get it run due to different operating systems, requirements and configurations. To solve this problem the containers come along.

A container is a software package which includes an application and all its libraries and dependencies, and can run independent and in any operative system. Docker is an open software platform that allows to create, deploy and run containers. Docker allows the automation virtualization of containers on the same instance of a light-weight Linux, avoiding the use of heavy-weight virtual machines.

EasyTV offers a variety of services that can run independent or together interacting each other. EasyTV will develop a Docker container for each service. At the end, the user who wants to host all this service in a server will have to deal with a lot of containers, that is why registries are needed.

A registry is a server-side application that allow the user to store, manage and distribute the Docker images of containers. So, why having a registry is important? Manage all Docker images is tedious, so a registry allows developers to pull and push images into the registry as in a repository. Once the image is in the registry it can be deployed as many times as wanted to different servers. Moreover, the registry works as a repository with different versions of the image. When the developer pushes new changes to the image, the registry automatically upgrades the container, also the client can deploy a tagged old version of the image instead of the last one.

3.2. Overview of Docker Registries

The aim of this section is to explain how a registry works and make a review about what platforms are available on the market and which are their functionalities, advantages and disadvantages.

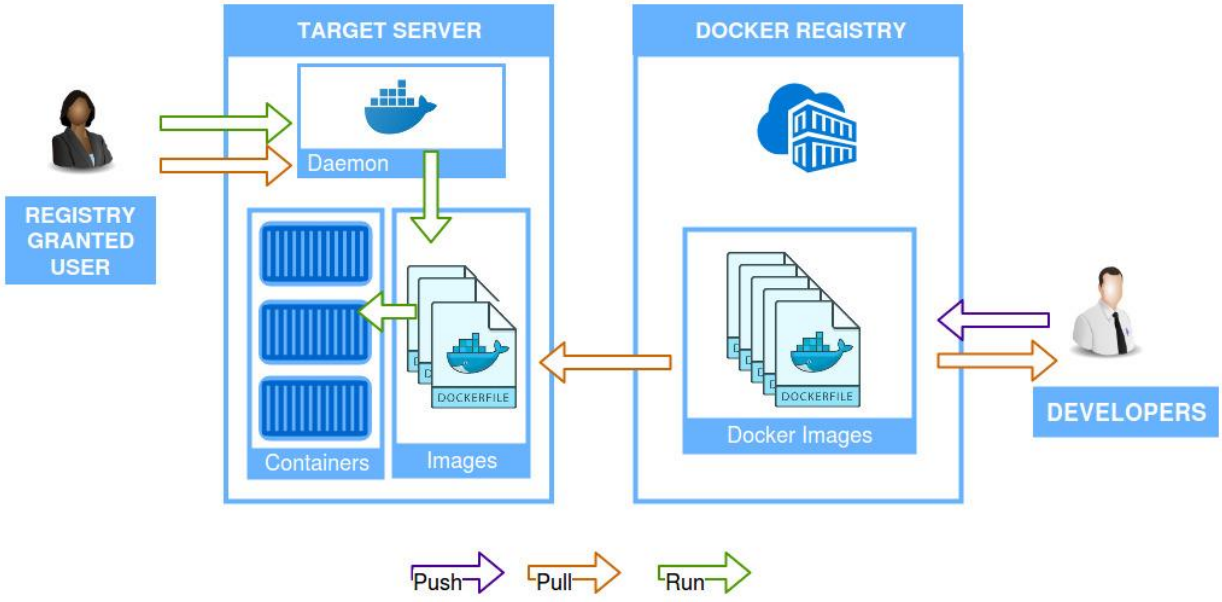

Fig. 1 Registry Architecture

In the Figure 1 we can see how is the life cycle of a docker image in a registry. The developers create an application, then they dockerize it and then build a Docker image. Once created the developers store the image on the registry by pushing it. Then the image became available for deploy as many times as wanted.

When a client wants to deploy a container of the images to a server hosting a Docker engine, first of all the the client order the Docker daemon to pull the image from the registry into the server. Once the image is in the server the client orders to the daemon to run the container so the application will become available on the server.

3.2.1. **Docker Hub**

Docker hub is the Registry from Docker, it is well documented and very easy to manage. All the users can create one free private repository and then pay depending on the private repositories they have.

Users can add individual collaborators to repositories. You can also create groups with access to repositories, but that means all the members of the group share the same level of access to a repository. Docker Hub supports Coarse Grained Authorization but it doesn't support LDAP, Active Directory or SAML.

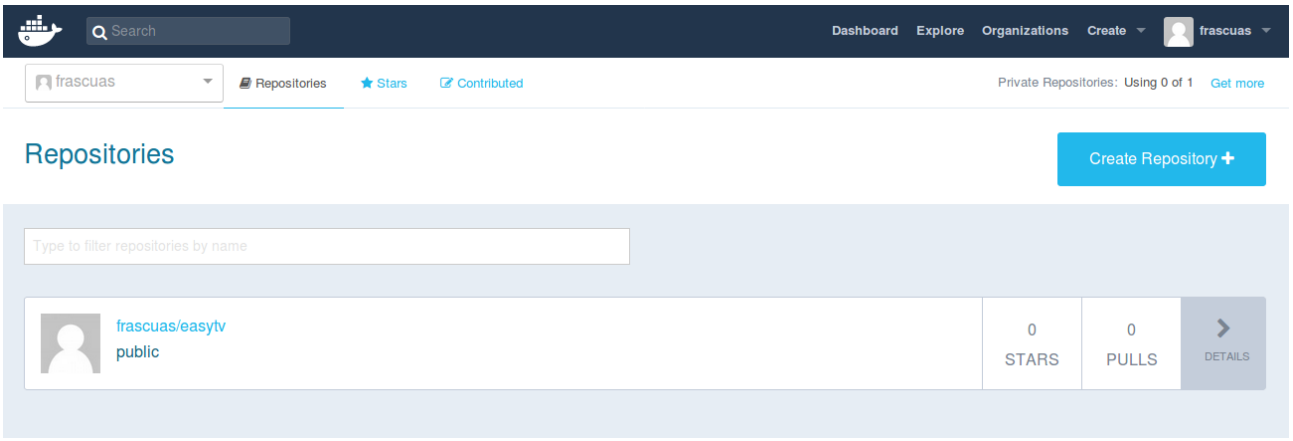

Fig. 2 Docker Hub Dashboard

When setting up an automated build, individual git branches and tags can be associated with docker tags, which is quite useful for building multiple versions of a docker image.

Docker hub can be integrated only with Github or Bitbucket to automatically build new images from a Dockerfile. Repositories branches and tags can be hooked with docker tags, so it is possible to have multiple versions of the docker images.

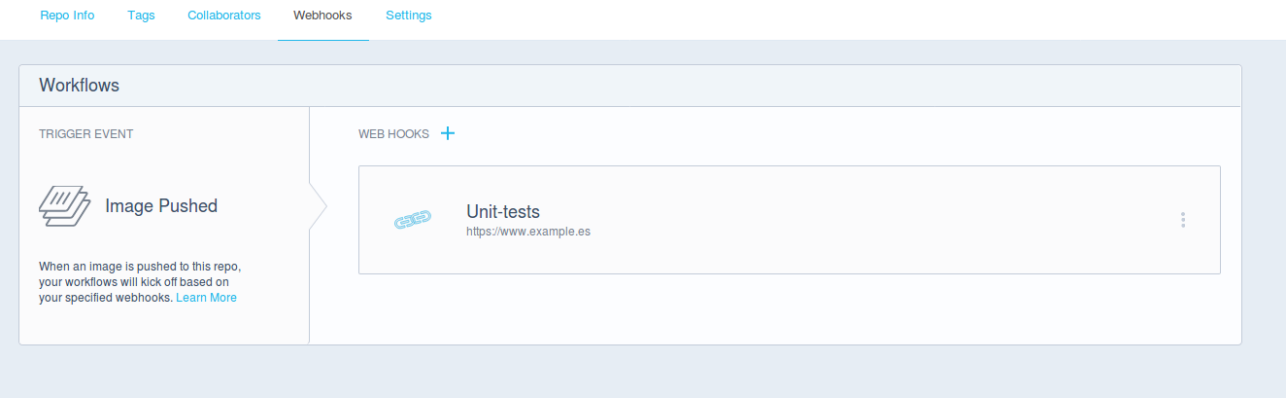

Fig. 3 Docker Hub Webhooks

Docker Hub can automate builds so an update in a repository will trigger an update in the linked image. Docker Hubs also allow to configure web-hooks to make HTTP calls when a specific event is triggered.

ADVANTAGES	DISADVANTAGES
Easy integration with Github and BitBucket	Build repositories lack support for custom git repositories
Github collaboration model	Limited insight into registry usage
Repository links and webhooks for build automation	Lacks support for external authentication providers
Well-documented and easy to use	uneven performance
ability to create organizations	lacks fine-grained access control
Inexpensive and usage-independent pricing	
Possibility of self-hosted	

Table 1 Docker Hub advantages and disadvantages

3.2.2. Quay.io

Quay.io is the Registry tool from CoreOs. Is free for public repositories and then a multiple paid plan depending on the number of private repositories to host. There is no additional charge for storage or band width.

Quay allows to create groups and teams, where each team has its own permissions. There are different levels of permissions from only pull, push and pull and admin. Users in teams inherit the permissions of the team. This allows a more controlled access to the repositories. Quay allows Oauth authentication but it doesn't support SAML, LDAP or Active Directory.

Quay has a very intuitive interface to automate builds. It supports custom gits repositories so is not depending on github or bitbucket. It supports regular expression for mapping branches and docker tags to automatically generate new docker images when an application is updated. Quay includes robot accounts to manage permissions for repositories which need to be shared across those repositories. Quay also has a very powerful notification system with email, webhooks or even HipChat integration.

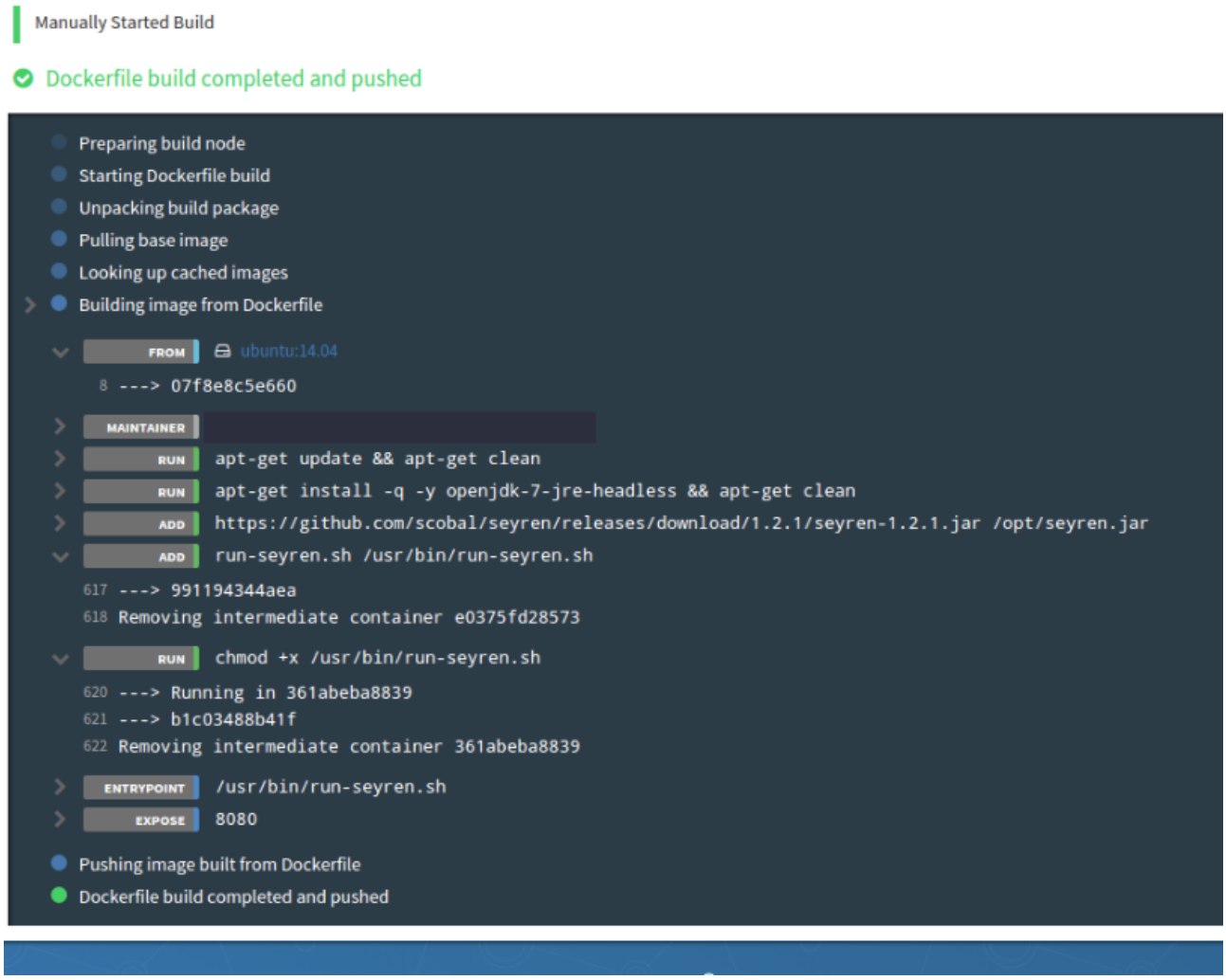

Fig. 4 Log of Quay manual Build

Quay has great tools to visualize the evolution of the image, get the build logs or see the audit trail. Quay offers visibility into all aspects of the docker registry.

ADVANTAGES	DISADVANTAGES
Clean, intuitive and streamlined interface limited	limited OAuth-only authentication support
Great automated builds	Not free private repositories
Great visibility into the registry usage, webhooks and rich event notifications	
webhooks and rich event notifications	
ability to create organizations and teams	
fine-grained access control through permissions	

Table 2 Quay advantages and disadvantages

3.2.3. **Artifactory**

Artifactory is an open source registry manager for binary artifacts. Artifactory focuses on artifact management, high availability and security. The cloud solution is very expensive, it has a monthly feed and they charge for storage and bandwidth.

It is a bit difficult to start using the tool, the docker registry is just another repository in artifactory, then they have three different repositories, local repositories hosted by artifactory, remote repositories where artifact is just a proxy and virtual repositories with a single endpoint to use multiple registries.

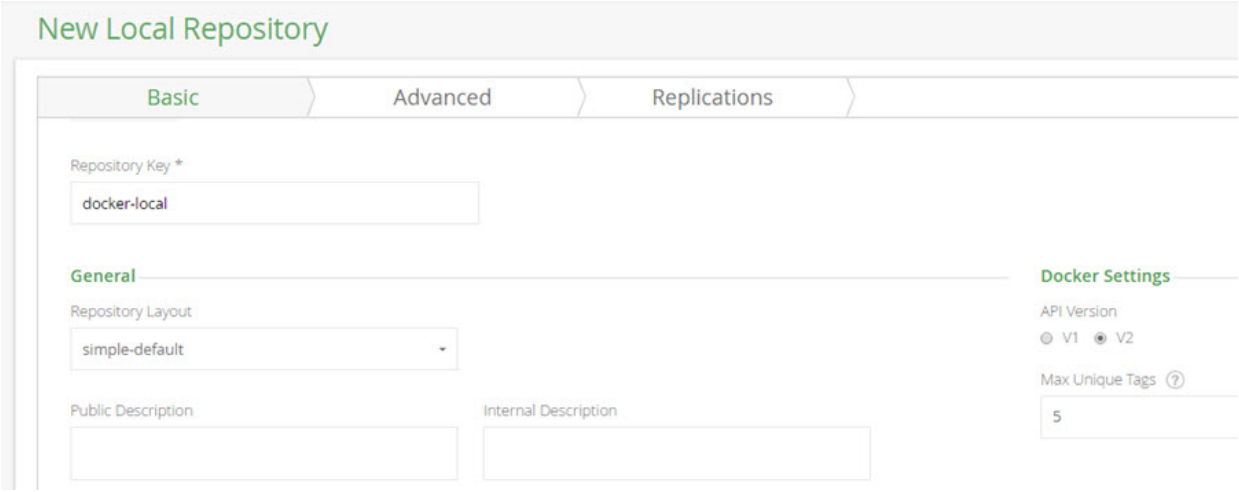

Fig. 5 Artifactory registry creation

The kind of repository similar to the others is the local repository. It allows to store tagged Docker Images. Artifactory allows fine-grained access control, allowing to give permissions to users and groups. It also supports other authentication options like LDAP or SAML.

ADVANTAGES	DISADVANTAGES
all-in-one artifact management model	Very difficult to use
comprehensive authentication capabilities	limited insight into registry usage
fine-grained access control	Very expensive
remote repositories for high availability	

Table 3 Artifactory advantages and disadvantages

3.2.4. Google container Registry (GCR)

GCR is the registry solution from google. The images are simply stored in a Google Cloud Storage (GCS), thanks to that fact GCR is not an expensive service. Also allows high availability and fast access.

The registry service doesn't support webhooks or notifications when a repository is updated. GCR doesn't provide insights of the registry usage.

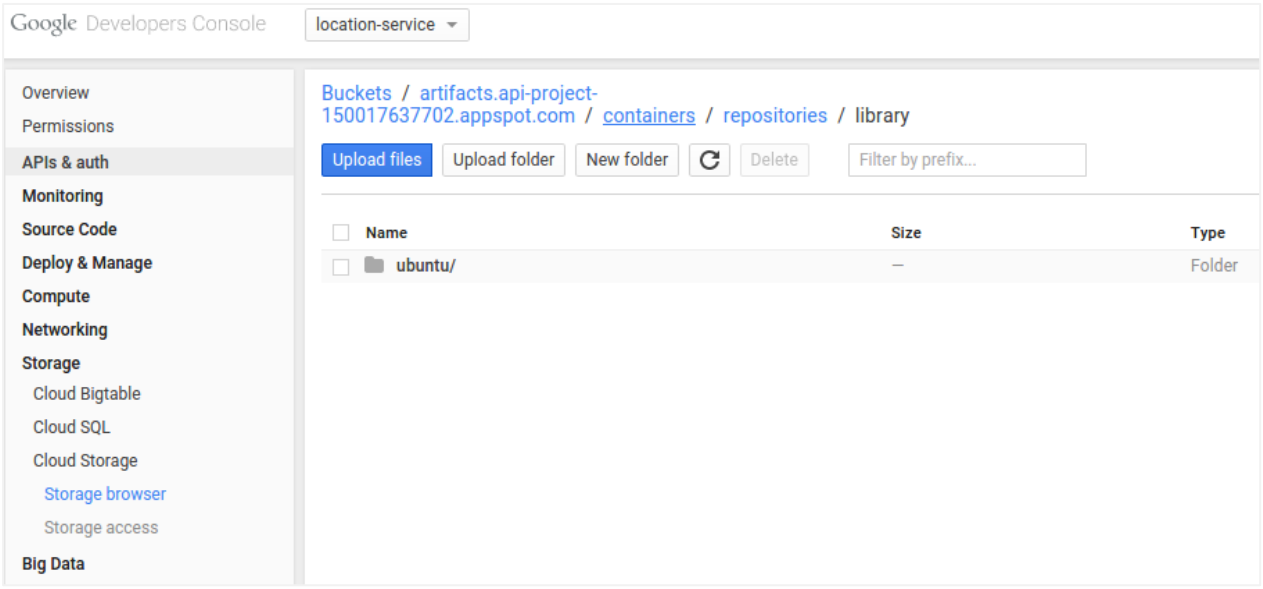

Fig. 6 GCR Console

For security GCR uses short period tokens when accessing the registry, the images are encrypted before write them on the GCS. The access to individual images and repositories are restricted through access control list.

ADVANTAGES	DISADVANTAGES
fine-grained access control with GCS ACLs	incompatibility with docker client
enhanced security through short-lived temporary auth tokens	minimal support for integration with build and deployment pipelines
Fine-grained access control	Limited visibility into registry usage

support for LDAP sync	
Uses the highly available Google cloud storage layer	
low price	

Table 4 GCR advantages and disadvantages

3.2.5. Amazon Elastic Container Registry

ECR is the registry solution from amazon, it is like any other registry but it takes advantage of the AWS ecosystem. It charges based on storage and bandwidth.

Figure 1

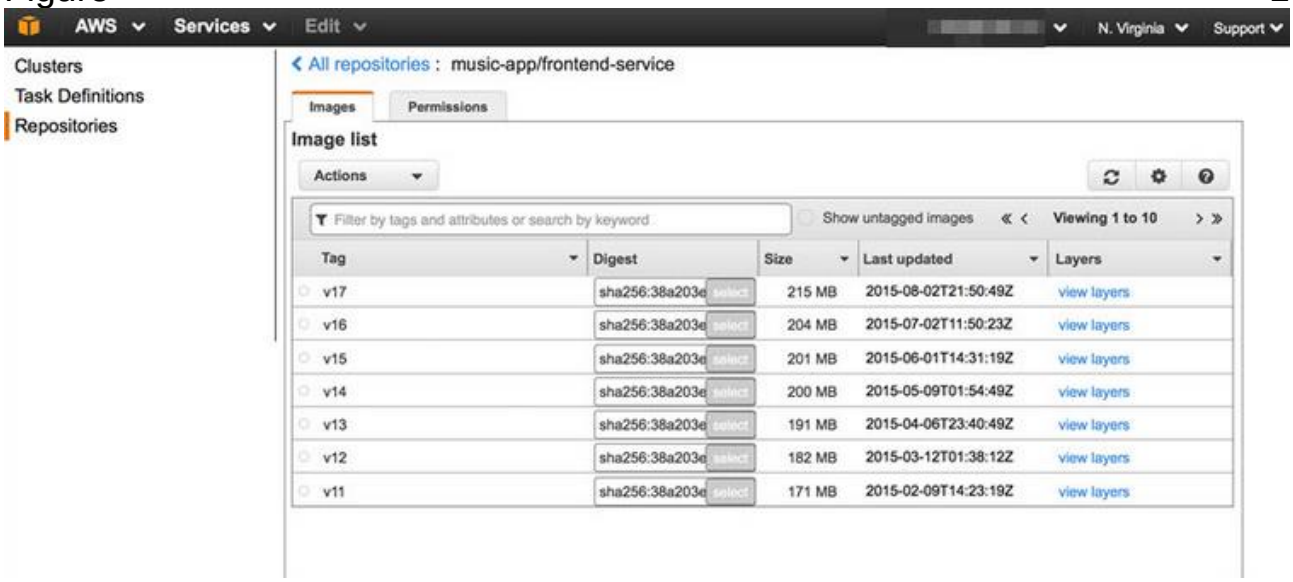

Fig. 7 Amazon ECR dashboard

ECR offers fine grained permissions and access control via AWS Identity and Access Management (IAM). ECR is integrated with Amazon EC2 Container Service (ECS), this simplify the development to production workflow.

ECR eliminates the need to operate your own container repositories or worry about scaling the underlying infrastructure. ECR hosts your images in a highly available and scalable architecture, allowing you to reliably deploy containers for your applications.

ADVANTAGES	DISADVANTAGES
Familiar to AWS users and easy to use.	Lack of insight into registry usage
Manage permissions and control access to user's images using AWS IAM users and roles	minimal support for integration with build and deployment pipelines
Fine-grained access support for LDAP sync	Requires creating a temporary token to connect with Docker

Not fees, pay just for what you use	expensive if the containers are not in AWS
Easy integration with Amazon ECS and the Docker CLI	

Table 5 Amazon ECR advantages and disadvantages

3.3. Docker Registry

It's been decided to use Docker registry, which is the on-premise version of Docker Hub. The registry of Docker was chosen because it obviously has full integration with Docker, but also is open source, has a big developer community and is provided with very complete image repositories, automated builds and web hooks.

Once the use of the docker solution was decided, there are two options, use the on-cloud version, Docker Hub, or the on-premise self-hosted option, Docker Registry. The final decision was to use the Docker Registry for these reasons:

- Get full control of where the images are being stored
- Fully ownership of the image's distribution pipeline
- Integration of the image storage and distribution tightly into self-in-house development work-flow
- Get as many private repositories as wanted without increasing the price of the system.

3.3.1. Installation

The first step is to install Docker on the host. Because a registry is an instance of the `registry` image, and runs within Docker. When Docker is installed in the host, the registry can be started with this command:

```
$ docker run -d -p 5000:5001 --restart=always --name registry registry:2
```

To configure the local port and the forwarding port the -p flag must be set. In the example the port 5000 is forwarding to the port 50001, where 5000 is the port in the host and 5001 is the port in the container.

If the registry is a part of a permanent infrastructure and the registry needs to start automatically the command must have the -restart flag set to always.

Once the registry is ready, these are the actions available:

3.3.1.1 Pull

This command pulls an image from a register

```
$ docker pull ubuntu:16.04
```

3.3.1.2 Tag

This command creates an additional tag for the existing image. When the first part of the tag is a host name and port, Docker interprets this as the location of a registry, when pushing.

```
$ docker tag ubuntu:16.04 localhost:5000/my-ubuntu
```

3.3.1.3 Push

This command pushes an image to the registry.

```
$ docker push localhost:5000/my-ubuntu
```

3.3.1.4 Remove

This command removes the local version of the image.

```
$ docker image remove localhost:5000/my-ubuntu
```

3.3.1.5 Stop

This command stops the registry.

```
$ docker container stop registry
```

3.3.1.6 Rm

This command removes the registry containers

```
$ docker container rm -v registry
```

3.3.2. **Customization**

The previous sections show a default configuration of a registry. But normally this is not he way is done in production. To customize the configuration Docker provides some options to change the values. The registry configuration is based in a Yaml file, that must be edited and review before moving the system to production.

To point out to docker the config file path, must be set on the run command.

```
$ docker run -d -p 5000:5000 --restart=always --name registry \
             -v `pwd`/config.yml:/etc/docker/registry/config.yml \
             registry:2
```

These are all the configuration options that can be included in the config file:

3.3.2.1 Version

The version option is required, it specifies the configuration's version.

```
version: 0.1
```

3.3.2.2 Log

The `log` subsection configures the behaviour of the logging system. The logging system outputs everything to stdout. You can adjust the granularity and format with this configuration section

```
log:
  accesslog:
    disabled: true
  level: debug
  formatter: text
  fields:
    service: registry
    environment: staging
```

PARAMETER	REQUIRED	DESCRIPTION
`level`	no	Sets the sensitivity of logging output. Permitted values are error, warn, info, and debug. The default is info.
`formatter`	no	This selects the format of logging output. The format primarily affects how keyed attributes for a log line are encoded. Options are text, json, and logstash. The default is text.

`fields`	no	A map of field names to values. These are added to every log line for the context. This is useful for identifying log messages source after being mixed in other systems.

Accesslog

Accesslog configures the behaviour of the access logging system. By default, the access logging system outputs to stdout. Access logging can be disabled by setting the boolean flag `disabled` to `true`.

```
accesslog:
  disabled: true
```

3.3.2.3 Hooks

This subsection configures the logging hooks' behaviour, like for example send an email.

```
hooks:
  - type: mail
    levels:
      - panic
    options:
      smtp:
        addr: smtp.sendhost.com:25
        username: sendername
        password: password
        insecure: true
      from: name@sendhost.com
      to:
        - name@receivehost.com
```

3.3.2.4 Storage

This option is required and defines which storage backend is in use. Just one storage system can be configured.

STORAGE	DESCRIPTION
`filesystem`	Uses the local disk to store registry files.
`azure`	Uses Microsoft Azure Blob Storage.
`gcs`	Uses Google Cloud Storage.

s3	Uses Amazon Simple Storage Service (S3) and compatible Storage Services.
Swift	Uses Openstack Swift object storage.
oss	Uses Aliyun OSS for object storage.

Table 6 Storage parameters

File system

```
filesystem:
  rootdirectory: /var/lib/registry
```

Azure

```
azure:
  accountname: accountname
  accountkey: base64encodedaccountkey
  container: containername
```

GCS

```
gcs:
  bucket: bucketname
  keyfile: /path/to/keyfile
  rootdirectory: /gcs/object/name/prefix
```

S3

```
s3:
  accesskey: awsaccesskey
  secretkey: awssecretkey
  region: us-west-1
  regionendpoint: http://myobjects.local
  bucket: bucketname
  encrypt: true
  keyid: mykeyid
  secure: true
  v4auth: true
```

```
    chunksize: 5242880
    multipartcopychunksize: 33554432
    multipartcopymaxconcurrency: 100
    multipartcopythresholdsize: 33554432
    rootdirectory: /s3/object/name/prefix
```

Swift

```
  swift:
    username: username
    password: password
    authurl: https://storage.myprovider.com/auth/v1.0p
    tenant: tenantname
    tenantid: tenantid
    domain: domain name for Openstack Identity v3 API
    domainid: domain id for Openstack Identity v3 API
    insecureskipverify: true
    region: fr
    container: containername
    rootdirectory: /swift/object/name/prefix
```

OSS

```
  oss:
    accesskeyid: accesskeyid
    accesskeysecret: accesskeysecret
    region: OSS region name
    endpoint: optional endpoints
    internal: optional internal endpoint
    bucket: OSS bucket
    encrypt: optional data encryption setting
    secure: optional ssl setting
    chunksize: optional size valye
    rootdirectory: optional root directory
    rootdirectory: /swift/object/name/prefix
```

3.3.2.5 Storage: Maintenance

Set up how the file system is going to be maintained. Currently, upload purging and read-only modes are the only available.

Uploadpurging

Upload purging is a background process that periodically removes orphaned files from the upload directories of the registry. Upload purging is enabled by default.

```
maintenance:
  uploadpurging:
    enabled: true
    age: 168h
    interval: 24h
    dryrun: false
```

PARAMETER	REQUIRED	DESCRIPTION
`enabled`	yes	Set to `true` to enable upload purging. Defaults to `true`.
`age`	yes	Upload directories which are older than this age will be deleted.Defaults to `168h` (1 week).
`interval`	yes	The interval between upload directory purging. Defaults to `24h`.
`dryrun`	yes	Set `dryrun` to `true` to obtain a summary of what directories will be deleted. Defaults to `false`.

Table 7 Uploadpurging parameters

Readonly

If the readonly section under maintenance has enabled set to true, clients will not be allowed to write to the registry.

```
redirect:
   disable: false
```

3.3.2.6 Storage: Delete

Use the delete structure to enable the deletion of image blobs and manifests by digest. It defaults to false.

```
delete:
  enabled: true
```

Cache

Use the cache structure to enable caching of data accessed in the storage backend. Currently, the only available cache provides fast access to layer metadata, which uses the blobdescriptor field if configured. You can set blobdescriptor field to redis or inmemory. If set to redis,a Redis pool caches layer metadata. If set to inmemory, an in-memory map caches layer metadata.

```
cache:
  blobdescriptor: inmemory
```

3.3.2.7 Auth

The auth option is optional. Possible auth providers are silly token and passwd. Just one auth configuration is permitted.

3.3.2.8 Auth: Silly

The silly authentication provider is only appropriate for development. It simply checks for the existence of the Authorization header in the HTTP request.

```
auth:
  silly:
    realm: silly-realm
    service: silly-service
```

PARAMETER	REQUIRED	DESCRIPTION
realm	yes	The realm in which the registry server authenticates.
service	yes	The service being authenticated.

Table 8 Auth parameters

3.3.2.9 Auth: Token

Token-based authentication allows you to decouple the authentication system from the registry. It is an established authentication paradigm with a high degree of security.

```
token:
  realm: token-realm
  service: token-service
  issuer: registry-token-issuer
  rootcertbundle: /root/certs/bundle
```

PARAMETER	REQUIRED	DESCRIPTION
realm	yes	The realm in which the registry server authenticates.
service	yes	The service being authenticated.
issuer	yes	The name of the token issuer. The issuer inserts this into the token so it must match the value configured for the issuer.
rootcertbundle	yes	The absolute path to the root certificate bundle. This bundle contains the public part of the certificates used to sign authentication tokens.

Table 9 Token parameters

3.3.2.10 Auth: Htpasswd

The htpasswd authentication backend allows you to configure basic authentication using an Apache htpasswd file. The only supported password format is `bcrypt`.

```
  htpasswd:
    realm: basic-realm
    path: /path/to/htpasswd
```

PARAMETER	REQUIRED	DESCRIPTION
realm	yes	The realm in which the registry server authenticates.
service	yes	The path to the `htpasswd` file to load at startup.

Table 10 Htpasswd parameters

3.3.2.11 Middleware

The middleware structure is optional. This option injects middleware at named hook points. Each middleware must implement the same interface as the object it is wrapping.

```
middleware:
  registry:
    - name: ARegistryMiddleware
      options:
        foo: bar
```

PARAMETER	REQUIRED	DESCRIPTION
name	yes	The name under which the middleware registers itself

options	yes	field is a map that details custom configuration required to initialize the middleware

Table 11 Middleware parameters

3.3.2.12 Middleware: Cloudfront

Cloudfront is a storage middleware .

```
middleware:
  storage:
    - name: cloudfront
      options:
        baseurl: https://my.cloudfronted.domain.com/
        privatekey: /path/to/pem
        keypairid: cloudfrontkeypairid
        duration: 3000s
```

PARAMETER	REQUIRED	DESCRIPTION
baseurl	yes	The SCHEME://HOST[/PATH] at which Cloudfront is served.
privatekey	yes	The private key for Cloudfront, provided by AWS.
keypairid	yes	The key pair ID provided by AWS.
duration	no	An integer and unit for the duration of the Cloudfront session

Table 12 Cloudfront parameters

3.3.2.13 Middleware: redirect

Redirect is a storage middleware.

```
middleware:
  storage:
    - name: redirect
      options:
        baseurl: https://example.com:5443
```

PARAMETER	REQUIRED	DESCRIPTION
baseurl	yes	SCHEME://HOST at which layers are served.

Table 13 Redirect parameter

3.3.2.14 Reporting

The reporting option is optional and configures error and metrics reporting tools. At the moment only two services are supported: Bugsnag and New Relic. Both can be configured at the same time.

```
reporting:
  bugsnag:
    apikey: bugsnagapikey
    releasestage: bugsnagreleasestage
    endpoint: bugsnagendpoint
  newrelic:
    licensekey: newreliclicensekey
    name: newrelicname
    verbose: true
```

3.3.2.15 Reporting: Bugsnag

PARAMETER	REQUIRED	DESCRIPTION
`apikey`	yes	The API Key provided by Bugsnag.
`releasestage`	no	Tracks where the registry is deployed, using a string like production, staging, or development.
`endpoint`	no	The enterprise Bugsnag endpoint.

Table 14 Bugsnag parameters

3.3.2.16 Reporting: New Relic

PARAMETER	REQUIRED	DESCRIPTION
`licensekey`	yes	License key provided by New Relic.
`name`	no	New Relic application name.
`verbose`	no	Set to true to enable New Relic debugging output on stdout.

Table 15 New Relic parameters

3.3.2.17 Htpp

The http option details the configuration for the HTTP server that hosts the registry.

```
http:
  addr: localhost:5000
  net: tcp
  prefix: /my/nested/registry/
  host: https://myregistryaddress.org:5000
  secret: asecretforlocaldevelopment
  relativeurls: false
  tls:
    certificate: /path/to/x509/public
    key: /path/to/x509/private
    clientcas:
      - /path/to/ca.pem
      - /path/to/another/ca.pem
    letsencrypt:
      cachefile: /path/to/cache-file
      email: emailused@letsencrypt.com
  debug:
    addr: localhost:5001
  headers:
    X-Content-Type-Options: [nosniff]
  http2:
    disabled: false
```

PARAMETER	REQUIRED	DESCRIPTION
addr	yes	The address for which the server should accept connections.
net	no	The network used to create a listening socket. Known networks are unix and tcp.
prefix	no	If the server does not run at the root path, set this to the value of the prefix. T
host	no	A fully-qualified URL for an externally-reachable address for the registry.
secret	no	A random piece of data used to sign state that may be stored with the client to protect against tampering.
relativeurls	no	If true, the registry returns relative URLs in Location headers.

3.3.2.18 Htpp: tls

The tls structure within http is optional. Used to configure TLS for the server.

PARAMETER	REQUIRED	DESCRIPTION

certificate	yes	Absolute path to the x509 certificate file.
key	yes	Absolute path to the x509 private key file.
clientcas	no	An array of absolute paths to x509 CA files.

3.3.2.19 Htpp: Letsencrypt

The letsencrypt structure within tls is optional. Used to configure TLS certificates provided by Letś Encrypt.

PARAMETER	REQUIRED	DESCRIPTION
certificate	yes	Absolute path to the x509 certificate file.
key	yes	Absolute path to the x509 private key file.
clientcas	no	An array of absolute paths to x509 CA files.

3.3.2.20 Http: Debug

The debug option is optional. Used to configure a debug server that can be helpful in diagnosing problems.

PARAMETER	REQUIRED	DESCRIPTION
addr	yes	which specifies the HOST:PORT on which the debug server should accept connections.

3.3.2.21 Http: Headers

The headers option is optional. Use it to specify headers that the HTTP server should include in responses. The headers option should contain an option for each header to include, where the parameter name is the header's name, and the parameter value a list of the header's payload values.

3.3.2.22 Http: Htpp2

The http2 structure within http is optional. Used to control http2 settings for the registry.

PARAMETER	REQUIRED	DESCRIPTION
disabled	no	If true, then http2 support is disabled.

Table 16 Http2 parameter

3.3.2.23 Notifications

The notifications option is optional and currently may contain a single option, endpoints. The endpoints structure contains a list of named services (URLs) that can accept event notifications.

```
notifications:
  endpoints:
    - name: alistener
      disabled: false
      url: https://my.listener.com/event
      headers: <http.Header>
      timeout: 500
      threshold: 5
      backoff: 1000
      ignoredmediatypes:
        - application/octet-stream
```

PARAMETER	REQUIRED	DESCRIPTION
`name`	yes	A human-readable name for the service.
`disabled`	no	If `true`, notifications are disabled for the service.
`url`	yes	The URL to which events should be published.
`headers`	yes	A list of static headers to add to each request. Each header's name is a key beneath `headers`, and each value is a list of payloads for that header name. Values must always be lists.
`timeout`	yes	A value for the HTTP timeout.
`threshold`	yes	An integer specifying how long to wait before backing off a failure.
`backoff`	yes	How long the system backs off before retrying after a failure.
`ignoredmediatypes`	no	A list of target media types to ignore.

Table 17 Notifications parameters

3.3.2.24 Redis

Declare parameters for constructing the redis connections. It caches information about immutable blobs.

```
redis:
  addr: localhost:6379
  password: asecret
  db: 0
  dialtimeout: 10ms
  readtimeout: 10ms
  writetimeout: 10ms
  pool:
    maxidle: 16
    maxactive: 64
    idletimeout: 300s
```

PARAMETER	REQUIRED	DESCRIPTION
addr	yes	The address (host and port) of the Redis instance.
password	no	A password used to authenticate to the Redis instance.
db	no	The name of the database to use for each connection.
dialtimeout	no	The timeout for connecting to the Redis instance.
readtimeout	no	The timeout for reading from the Redis instance.
writetimeout	no	The timeout for writing to the Redis instance.

Table 18 Redis parameters

3.3.2.25 Redis: pool

Use these settings to configure the behaviour of the Redis connection pool.

PARAMETER	REQUIRED	DESCRIPTION
maxidle	no	The maximum number of idle connections in the pool.
maxactive	no	The maximum number of connections which can be open before blocking a connection request.

idletimeout	no	How long to wait before closing inactive connections.

Table 19 Pool parameters

3.3.2.26 Health

The health option is optional, and contains preferences for a periodic health check on the storage driver's backend storage, as well as optional periodic checks on local files, HTTP URIs, and/or TCP servers. The results of the health checks are available at the /debug/health endpoint on the debug HTTP server if the debug HTTP server is enabled

```
health:
  storagedriver:
    enabled: true
    interval: 10s
    threshold: 3
  file:
    - file: /path/to/checked/file
      interval: 10s
  http:
    - uri: http://server.to.check/must/return/200
      headers:
        Authorization: [Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==]
      statuscode: 200
      timeout: 3s
      interval: 10s
      threshold: 3
  tcp:
    - addr: redis-server.domain.com:6379
      timeout: 3s
      interval: 10s
      threshold: 3
```

3.3.2.27 Health: Storagedriver

The storagedriver structure contains options for a health check on the configured storage driver's backend storage.

PARAMETER	REQUIRED	DESCRIPTION
enabled	yes	Set to `true` to enable storage driver health checks or `false` to disable them.
interval	no	How long to wait between repetitions of the storage driver health check.
threshold	no	A positive integer which represents the number of times the check must fail before the state is marked as unhealthy.

Table 20 Storagedriver parameters

3.3.2.28 Health: File

The file structure includes a list of paths to be periodically checked for the\ existence of a file.

PARAMETER	REQUIRED	DESCRIPTION
`file`	yes	The path to check for existence of a file.
`interval`	no	How long to wait before repeating the check.

Table 21 File parameters

3.3.2.29 Health: Head

The http structure includes a list of HTTPs URIs to periodically check with HEAD requests. If a HEAD request does not complete or returns an unexpected status code, the health check will fail.

PARAMETER	REQUIRED	DESCRIPTION
`uri`	yes	The URI to check.
`headers`	no	Static headers to add to each request. Each header's name is a key beneath `headers`, and each value is a list of payloads for that header name. Values must always be lists.
`statuscode`	no	The expected status code from the HTTP URI. Defaults to `200`.
`timeout`	no	How long to wait before timing out the HTTP request. is interpreted as a number of nanoseconds.
`interval`	no	How long to wait before repeating the check.
`threshold`	no	The number of times the check must fail before the state is marked as unhealthy. If this field is not specified, a single failure marks the state as unhealthy.

Table 22 Head parameters

3.3.2.30 Health: tcp

The tcp structure includes a list of TCP addresses to periodically check using TCP connection attempts. Addresses must include port numbers. If a connection attempt fails, the health check will fail.

PARAMETER	REQUIRED	DESCRIPTION
addr	yes	The TCP address and port to connect to.
timeout	no	How long to wait before timing out the TCP connection.
interval	no	How long to wait between repetitions of the check.
threshold	no	The number of times the check must fail before the state is marked as unhealthy. If this field is not specified, a single failure marks the state as unhealthy.
interval	no	How long to wait before repeating the check.
threshold	no	The number of times the check must fail before the state is marked as unhealthy. If this field is not specified, a single failure marks the state as unhealthy.

Table 23 Tcp parameters

3.3.2.31 Proxy

The proxy structure allows a registry to be configured as a pull-through cache to Docker Hub.

```
proxy:
  remoteurl: https://registry-1.docker.io
  username: [username]
  password: [password]
```

PARAMETER	REQUIRED	DESCRIPTION
remoteurl	yes	The URL for the repository on Docker Hub.
username	no	The username registered with Docker Hub which has access to the repository.
password	no	The password used to authenticate to Docker Hub

Table 24 Proxy parameters

3.3.2.32 Compatibility

The compatibility structure configures the handling of older and deprecated features. Each subsection defines such a feature with configurable behaviour.

```
compatibility:
  schema1:
    signingkeyfile: /etc/registry/key.json
```

PARAMETER	REQUIRED	DESCRIPTION
`signingkeyfile`	no	The signing private key used to add signatures to `schema1` manifests. If no signing key is provided, a new ECDSA key is generated when the registry starts.

Table 25 Compatibility parameter

3.3.2.33 Validation

Validation options for restrict URLs in pushed manifest.

```
validation:
  enabled: true
  manifests:
    urls:
      allow:
        - ^https?://([^/]+\.)*example\.com/
      deny:
        - ^https?://www\.example\.com/
```

3.3.2.34 Validation: enabled

The enabled flag is used to enable the other options in the validation section. They are disabled by default.

3.3.2.35 Validation: manifests

Subsection to configure the manifest validation. Organizes a list of URLs in two categories “allow” and “deny”.

`urls`	yes	A map of allow and deny options, which are a list of regular expression that restrict the URLs in pushed manifests.

Table 26 Manifest parameter

3.3.2.36 Overriding configuration

To override a configuration variable from the environment without editing the configuration file, it is possible to specify a variable by passing -e argument into a docker run command.

To override a configuration option, create an environment variable named `REGISTRY_variable`

where `variable` is the name of the configuration option and the _ (underscore) represents indention levels.

For his registry configuration option:

```
storage:
  filesystem:
    rootdirectory: /var/lib/registry
```

This is the equivalent environment variable

```
REGISTRY_STORAGE_FILESYSTEM_ROOTDIRECTORY=/somewhere
```

3.4. Example

Here is an example of a complete configuration YAML file for a docker registry with all the options available.

```
version: 0.1
log:
  accesslog:
    disabled: true
  level: debug
  formatter: text
  fields:
    service: registry
    environment: staging
  hooks:
    - type: mail
      disabled: true
      levels:
        - panic
      options:
        smtp:
          addr: mail.example.com:25
          username: mailuser
          password: password
          insecure: true
        from: sender@example.com
        to:
          - errors@example.com
loglevel: debug # deprecated: use "log"
storage:
  filesystem:
    rootdirectory: /var/lib/registry
    maxthreads: 100
  azure:
    accountname: accountname
```

```
    accountkey: base64encodedaccountkey
    container: containername
  gcs:
    bucket: bucketname
    keyfile: /path/to/keyfile
    rootdirectory: /gcs/object/name/prefix
    chunksize: 5242880
  s3:
    accesskey: awsaccesskey
    secretkey: awssecretkey
    region: us-west-1
    regionendpoint: http://myobjects.local
    bucket: bucketname
    encrypt: true
    keyid: mykeyid
    secure: true
    v4auth: true
    chunksize: 5242880
    multipartcopychunksize: 33554432
    multipartcopymaxconcurrency: 100
    multipartcopythresholdsize: 33554432
    rootdirectory: /s3/object/name/prefix
  swift:
    username: username
    password: password
    authurl:         https://storage.myprovider.com/auth/v1.0         or
https://storage.myprovider.com/v2.0                                   or
https://storage.myprovider.com/v3/auth
    tenant: tenantname
    tenantid: tenantid
    domain: domain name for Openstack Identity v3 API
    domainid: domain id for Openstack Identity v3 API
    insecureskipverify: true
    region: fr
    container: containername
    rootdirectory: /swift/object/name/prefix
  oss:
      allow:
        - ^https?://([^/]+\.)*example\.com/
      deny:
        - ^https?://www\.example\.com/

    accesskeyid: accesskeyid
    accesskeysecret: accesskeysecret
    region: OSS region name
    endpoint: optional endpoints
    internal: optional internal endpoint
    bucket: OSS bucket
    encrypt: optional data encryption setting
    secure: optional ssl setting
    chunksize: optional size valye
    rootdirectory: optional root directory
  inmemory:  # This driver takes no parameters
  delete:
    enabled: false
  redirect:
```

```
    disable: false
  cache:
    blobdescriptor: redis
  maintenance:
    uploadpurging:
      enabled: true
      age: 168h
      interval: 24h
      dryrun: false
    readonly:
      enabled: false
auth:
  silly:
    realm: silly-realm
    service: silly-service
  token:
    realm: token-realm
    service: token-service
    issuer: registry-token-issuer
    rootcertbundle: /root/certs/bundle
  htpasswd:
    realm: basic-realm
    path: /path/to/htpasswd
middleware:
  registry:
    - name: ARegistryMiddleware
      options:
        foo: bar
  repository:
    - name: ARepositoryMiddleware
      options:
        foo: bar
  storage:
    - name: cloudfront
      options:
        baseurl: https://my.cloudfronted.domain.com/
        privatekey: /path/to/pem
        keypairid: cloudfrontkeypairid
        duration: 3000s
  storage:
    - name: redirect
      options:
        baseurl: https://example.com/
reporting:
  bugsnag:
    apikey: bugsnagapikey
    releasestage: bugsnagreleasestage
    endpoint: bugsnagendpoint
  newrelic:
    licensekey: newreliclicensekey
    name: newrelicname
    verbose: true
http:
  addr: localhost:5000
  prefix: /my/nested/registry/
  host: https://myregistryaddress.org:5000
```

```
  secret: asecretforlocaldevelopment
  relativeurls: false
  tls:
    certificate: /path/to/x509/public
    key: /path/to/x509/private
    clientcas:
      - /path/to/ca.pem
      - /path/to/another/ca.pem
    letsencrypt:
      cachefile: /path/to/cache-file
      email: emailused@letsencrypt.com
  debug:
    addr: localhost:5001
  headers:
    X-Content-Type-Options: [nosniff]
  http2:
    disabled: false
notifications:
  endpoints:
    - name: alistener
      disabled: false
      url: https://my.listener.com/event
      headers: <http.Header>
      timeout: 500
      threshold: 5
      backoff: 1000
      ignoredmediatypes:
        - application/octet-stream
redis:
  addr: localhost:6379
  password: asecret
  db: 0
  dialtimeout: 10ms
  readtimeout: 10ms
  writetimeout: 10ms
  pool:
    maxidle: 16
    maxactive: 64
    idletimeout: 300s
health:
  storagedriver:
    enabled: true
    interval: 10s
    threshold: 3
  file:
    - file: /path/to/checked/file
      interval: 10s
  http:
    - uri: http://server.to.check/must/return/200
      headers:
        Authorization: [Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==]
      statuscode: 200
      timeout: 3s
      interval: 10s
      threshold: 3
  tcp:
```

```
    - addr: redis-server.domain.com:6379
      timeout: 3s
      interval: 10s
      threshold: 3
proxy:
  remoteurl: https://registry-1.docker.io
  username: [username]
  password: [password]
compatibility:
  schema1:
    signingkeyfile: /etc/registry/key.json
validation:
  enabled: true
  manifests:
    urls:
      allow:
        - ^https?://([^/]+\.)*example\.com/
      deny:
        - ^https?://www\.example\.com/
```

4. EASY TV SERVICE CATALOGUE

The EasyTV Service Catalogue is “a web-based catalogue where the final user can choose the services of his/her own interest among those running in the EasyTV multi terminal technical platform"[3].

It is part of the four pillars that constitute the EasyTV system:

- a multi-terminal technical platform;
- a group of platform-based service components;
- a service registry and catalogue;
- a service development kit;

Implementation and integration of these pillars are the main results that WP5 aims to achieve. It is immediately evident that design and development of the Service Catalogue are strictly interdependent from:

- the various technological choices made during the project;
- components and services provided by the platform;
- the final users’ needs related to services.

This preliminary consideration anticipates that the definition of the Service Catalogue is an ongoing process.

During the last 6 months (M6-M12) different activities have been conducted to set up a preliminary design phase of the EasyTV Service Catalogue.

The aim of the present section is to report considerations and evidences emerged so far in the activities related the Service Catalogue design and implementation process. Specifically, regarding the following aspects:

- **Monitoring Easy TV project deployment:** considering the ongoing process of the project, a *mid-term perspective* has been adopted to evaluate the state-of-art of EasyTV and the impact of current intermediate status on the Service Catalogue envisioning process.
- **Accessibility standard examination:** accessibility standard has been considered as the primary high-level requirement of the EasyTV catalogue.
- **Design methodology overview:** through an analysis of the main design trends, the “Systemic Design” has been identified as a widespread and robust approach. Subsequently, has been proposed for the catalogue an “Accessible design system” and has been defined the Atomic Design System as a suitable design methodology.
- **Development approach definition:** a consequent development approach has been proposed, considering the implementation objectives planned for the D5.8 Final report on the set up and implementation of the EasyTV Service Registry and Catalogue (M25).

The following paragraphs propose a design method for approaching the implementation and the set up of the EasyTV Service Catalogue, identifying and considering all the main relevant aspects, critical issues and constraints so far emerged, mainly related to EasyTV purposes and deployment. Secondly, we will define the structure of a **Design System** and delimit its perimeter in terms of components.

Moreover, further uses and finalities of Service Catalogue “design system” have been suggested.

Regarding the next phases, the scheduled activities for achieving the EasyTV web-based catalogue of services are:

1. **Catalogue Wireframing:** wireframes are used in the early phase of the design and

[3] EasyTV proposal p. 54

development process. A wireframe represents the skeletal framework of a website and defines general user flows. The wireframing phase of the Service Catalogue aims to finalize a layout for its key pages. This work supports the consistency and robustness of choices regarding the user interface elements that will be implemented, that represent a crucial phase of the interaction design process.

2. **Catalogue Graphic design:** graphic design phase aims to create the high-fidelity prototypes starting from wireframes. This stage enables to explore and define the UI elements, pinpointing which elements work best, and predicting usability problems.
3. **Catalogue implementation:** development and implementation steps represent the processes by which achieving the fulfillment of prototype system designed for the Service Catalogue. At the end of the implementation phase, the result is evaluated according to the list of requirements that was created in the definition phase.

4.1. Mid-term perspective

A "mid-term" perspective allow to present a preliminary analysis and evaluation of the EasyTV Service Catalogue's design and scope, considering the intermediate status of the whole EasyTV project.

All the deliverables produced till now have been examined (M12-Year 1) to gather the fundamental aspects of the Catalogue considering the state-of-the-art of the project. They represent the starting point of the Service Catalogue envisioning process.

In general, from the Catalogue, EasyTV services are available to be chosen by the final users[4]. Starting from evidences of contributions (Table 27), it has been firstly defined *final users* involved, *services* available and *platforms* and *technologies* adopted.

WP	Deliverable	Topic
WP1 **Requirements, specification and technical architecture**	D1.1 User scenario and requirements definition	Final User
WP1 **Requirements, specification and technical architecture**	D1.2 EasyTV system requirements specification	• Final User • General components Overview on interaction devices
WP1 **Requirements, specification and technical architecture**	D1.4 Final release of the EasyTV system architecture	• Final User • Generic System Services
WP6 **Testing with users and feedback**	D6.1 Report on Demographics for the Tests	• Final User • Group of Technologies and services
WP7 **Dissemination and**	D7.1 Early-stage market analysis and initial business model	• Final User • Product and Services

[4] EasyTV proposal p. 55

exploitation		

Table 27 Main EasyTV deliverables (at M12) impacting on Service Catalogue

The following elements represent the starting point to consider for the Service Catalogue design approach:

- **End users**: people with hearing loss or hard of hearing/people with sight loss or visually impaired.

  Since the main aspect of the EasyTV platform is to develop multiple functionalities with the aim to allow users with different disabilities accessing broadcaster content in an easier way, the EasyTV Service Catalogue is intended as the access channel to the innovative services introduced by the EasyTV project.

  Services expected: Sign Language Translation, Personalization, Recommendation of available access and interaction services, Universal remote control, Image magnification and adaptation, Object based audio, Optical character recognition (OCR), Speech to text, Subtitles, Audio description, Audio subtitling.

- **Expert users:** People with high level of expertise in Sign Language and proficiency in translation activities (i.e. professional subtitlers).

  This kind of final user will use the platform mainly to collaborate for social purposes and, in any case, without profit finality.

  Services expected: Crowdsourcing platform (it will be used by expert and end-user associations to openly contribute with subtitles and sign subtitles that, furthermore, will simultaneously be translated to multiple languages).

- **Technical users:** third-party developers or external software companies

  These actors might be interested (with/without profit finality) to develop, test and deliver their own «new» services in the EasyTV multi-terminal technical platform.

  Services expected: Service Development Kit (Kit of tools that facilitating the integration of new services into the platform).

Although the final users and platform services have been sufficiently determined so far, at the current phase of the EasyTV project issues referred to platform/devices to be used for the access to Service Catalogue, and relative service delivery/fruition modalities by users, remain hypothesized but not well-defined.

According to the roadmap of the project, the EasyTV component-based system is developed during the WP5 and better defined through the evidences that will emerge from the next deliverables.

At this point of the development (M12) the potential user groups to reach, and the services to offer, through the Service Catalog have been identified.

During the next steps of the project the effective user interaction will be better defined. At the same time, User Experience has been integrated into the present section as a background of all the following aspects addressed for the preliminary set up of the EasyTV Service catalogue: *Accessibility standard*, *Design methodology* and *Development approach*.

4.2. Accessibility standard

The EasyTV project aims at developing media improved access services and making distribution of novel accessibility features. Therefore, services in EasyTV are designed as solutions to sensory impairments, so users with visual and hearing impairments are the intended population.

Starting from this assumption, as service catalogue is intended to be Web-based, it is fundamental to outline the accessibility requirements of the catalogue in the preliminary phase.

Following, the main current references regarding Web Accessibility representing a corpus of guideline and inspiration examples for design and implementation of the EasyTV Service Catalogue are presented.

4.2.1. W3C: Web Content Accessibility Guidelines (WCAG)

The World Wide Web Consortium (W3C) is an international community that develops open international standards to ensure the long-term growth of the Web.

The W3C Web Accessibility Initiative (WAI) aims to develop standards and support materials to help design and developers understanding and implementing accessibility.

The W3C provides web accessibility standards and guidelines for different components:

- *Authoring Tool Accessibility Guidelines (ATAG)* addresses authoring tools;
- *Web Content Accessibility Guidelines (WCAG)* addresses web content and used by developers, authoring tools and accessibility evaluation tools;
- *User Agent Accessibility Guidelines (UAAG)* addresses web browsers and media players, including some aspects of assistive technologies.

Focusing on the content accessibility, **Web Content Accessibility Guidelines (WCAG)** cover a wide range of recommendations for making Web content more accessible and they aim at representing stable, referenceable, technical standards.

WCAG 1.0 was the first version of the guidelines finalized in 1999. WCAG 2.0[5] was published on 11 December 2008 and have been considered the standard reference so far. On 5 June 2018, WCAG 2.1[6], built on WCAG 2.0 was published. It is backwards compatible with WCAG 2.0, meaning web pages that conform to WCAG 2.1 also conform to WCAG 2.0. All requirements ("success criteria") from 2.0 are included in 2.1.

The 13 guidelines of WCAG 2.1, as for WCAG 2.0, are organized under 4 principles: perceivable, operable, understandable and robust, providing the basic goals that authors should work towards in order to make content more accessible to users with different disabilities.

The four principles provide the foundation for Web accessibility:

1. *Perceivable*: users must be able to perceive the information being presented.
2. *Operable:* users must be able to operate the interface.
3. *Understandable:* users must be able to understand the information as well as the operation of the user interface.
4. *Robust:* users must be able to access the content as technologies advance.

Under each principle of WCAG 2.1 there is a list of guidelines that meet the principle:

1. Perceivable:
    a. Provide text alternatives for any non-text content so that it can be changed into other

[5] https://www.w3.org/TR/WCAG20/

[6] https://www.w3.org/TR/WCAG21/

forms people need, such as large print, braille, speech, symbols or simpler language.

b. Provide alternatives for time-based media.

c. Create content that can be presented in different ways (for example simpler layout) without losing information or structure.

d. Make it easier for users to see and hear content including separating foreground from background.

2. Operable:

a. Make all functionality available from a keyboard.

b. Provide users enough time to read and use content.

c. Do not design content in a way that is known to cause seizures or physical reactions.

d. Provide ways to help users navigate, find content, and determine where they are.

e. Make it easier for users to operate functionality through various inputs beyond keyboard

3. Understandable:

a. Make text content readable and understandable.

b. Make Web pages appear and operate in predictable ways.

c. Help users avoid and correct mistakes.

4. Robust:

a. Maximise compatibility with current and future user agents, including assistive technologies.

The guidelines provide, also, the framework and overall objectives to help authors understand the success criteria and better implement the techniques. For each guideline, there are testable success criteria, which are at three levels of conformance are defined: A (lowest), AA, and AAA (highest).

4.2.2. **Beyond the WCAG recommendations**

For the envisioning process related to accessibility requirement of the EasyTV catalogue, in addition to WCAG recommendations it is important to consider examples of online (currently) website that are addressed directly to the same groups of final users of EasyTV project.

Following some inspiration examples are proposed for different group of final users:

- **People with disabilities**
  - *European disability forum* http://www.edf-feph.org/

  The European Disability Forum is an independent NGO that defends the interests of 80 million Europeans with disabilities. It is an unique platform which aims to brings together representative organization of persons with disabilities from across Europe. It is directly managed by persons with disabilities and their families and give an example of a web platform having as target group of final users the whole people with disabilities.

- **Blind and partially sighted people**
  - *European Blind Union* http://www.euroblind.org/
  - *World Blind Union* http://www.worldblindunion.org

  These websites refer, respectively, to the European and the international organizations

that represent people who are blind or partially sighted. They can be considered "inspirational" for finding web design solution addressing these kinds of final users.

- **Deaf (SL)**
  - *European Union of the Deaf* https://www.eud.eu/
  - *Word Federtion of the deaf* http://wfdeaf.org/
  - *National Association of the Deaf* https://www.nad.org/

  Examples of website of organizations working to ensure equal rights and opportunities for deaf and hard of hearing people. Also in this case, these websites can be considered to envision web design solution addressing these kinds of final users.

4.2.3. **EU policy on Web Accessibility**

For many years European Commission has worked to achieve objectives of improving accessibility of ICT and digital services.

Currently, EU Digital Single Market strategy involves policies regarding the Web Accessibility[7] in order to provide equal access and equal opportunity to persons with disabilities.

Web accessibility aims to enable all users to have equal access to information and functionalities on the web. More specifically, Web accessibility means that people with all abilities and disabilities can perceive, understand, browse and interact with the Web.

In December 2012 the European Commission presented the *Proposal for a Directive of the European Parliament and of the Council on the accessibility of public sector bodies' websites* providing common rules across all Member States.

In February 2014 the *European standard* on accessibility requirements was adopted for public procurement of ICT products and services. The current version has been released in August 2018, the EN 301 549 V2.1.2 *Accessibility requirements for ICT products and services*[8]; it is a set of functional accessibility requirements broken down into chapters. The adoption of the W3C *Web Content Accessibility Guidelines (WCAG) 2.1* is the most significant change in this version for web contents, electronic documents and non-web software, such as native mobile applications. The goal is to meet the needs of the EU Web Accessibility Directive.

The *Directive (EU) 2016/2102 of the European Parliament and of the Council of 26 October 2016 on the accessibility of the websites and mobile applications of public sector bodies* was published on 2 December 2016.

The main purpose of the Directive is allowing people with disabilities (especially persons with vision or hearing impairments) to have better access to the websites and mobile applications of public services.

Consistent with Web Content Accessibility Guidelines (WCAG) 2.0 mentioned above, the four principles of accessibility cited in the EU Directive are:

- *Perceivability*: information and user interface components must be presentable to users in ways they can perceive;
- *Operability*: meaning that user interface components and navigation must be operable;
- *understandability*: information and the operation of the user interface must be understandable;
- *robustness*: meaning that contents must be robust enough to be interpreted reliably by a

[7] https://ec.europa.eu/digital-single-market/en/web-accessibility

[8] https://www.etsi.org/deliver/etsi_en/301500_301599/301549/02.01.02_60/en_301549v020102p.pdf

wide variety of user agents, including assistive technologies.

The *Directive (EU) 2016/2102* came along with the European Accessibility Act[9] (December 2015) that aims to improve the accessibility of products and services in the single market.

4.3. Design methodology

In this document, to find a design methodology means firstly finding the best solution for the currently "open" design status of the Service Catalogue that reflects the still opened (ongoing) process of the EasyTV project. Secondly, in this phase, the most critical concern is to fit the needs and wants of the final users considering all evidence, indications and constraints so far identified.

An overview of global design trends have been conducted in order to evaluate the proper design approach and/or methodology taking into account the main EASYTV system requirements.

4.3.1. Design Trend

With the increasing complexity of society, caused by multiple factors (including globalization, migration, sustainability), traditional design methods have become insufficient.

Influence of systems theory, united to the need to consider heterogeneous groups of persons/users (grouped according to different criteria) and not the individual, have conducted to a "Systemic" approach.

The strategic role of Systems Thinking has been underlined and explained in the 2012 book by Harold G. Nelson and Erik Stolterman "The Design Way: Intentional Change in an Unpredictable World" which explores design as inquiry for action.

The **Systemic design** is intended for a recent initiative in design that aim to face challenges characterised by complexity, uniqueness, value conflict, and ambiguity over objectives. It proposes to integrate Systems Thinking and Human-Centered Design with the intention of helping designers cope with complex design projects.

Systemic design is an approach that supporting to address questions related to the interdipendece between products, services, processes and policies that have social repercussions.

Following this perspective, the design process becomes both systemic (integrative and interconnected) and systematic (methodical)[10]. It allows different teams to develop an elevated perspective of the challenge and translate novel insights into rapid action.

All systemic design is provisional and open to redesign and projects carried out with this approach are entangled with social systems.

Systems Thinking leads to modelling and intervening in the world as if it is composed of open, purposeful and complex wholes. As introduced, key aspects of systemic approach are its inquiring, open, integrative, collaborative and interdependence core. Reciprocal influence between parts of a greater whole and their environment are taken into account: interdependencies between system components and their environment give rise to self-organisation, learning, adaptation, evolution, complexity that is managed.

Systemic Design approach is covering a key role in Digital Transformation process, developing methodologies and approach that help to integrate systems thinking with design towards sustainability at environmental, social and economic level.

In the last years, specifically in the web design community, one of the most operative application of the systemic approach in design practice and activities is the adoption of a **design system**.

[9] http://ec.europa.eu/social/main.jsp?catId=1202

[10] Harold G. Nelson and Erik Stolterman, "The Design Way: Intentional Change in an Unpredictable World, Second Edition"

4.3.2. **Design System**

As defined by Nathan Curtis “almost always, a design system offers a library of visual style and components documented and released as reusable code for developers and/or tool(s) for designers. A system may also offer guidance on accessibility, page layout and editorial and less often branding, data viz, UX patterns and other tools”[11].

Accordingly to the User Centred Design principles, a design system is a set of standards for design and code along with components that unify both practices, becoming a documentation and a modular toolkit for designers and developers. Engineering teams are able to refer to a single source at the implementation phase and this impacts inevitably in a positive way system consistency (i.e. all call-to-actions look the same across screens).

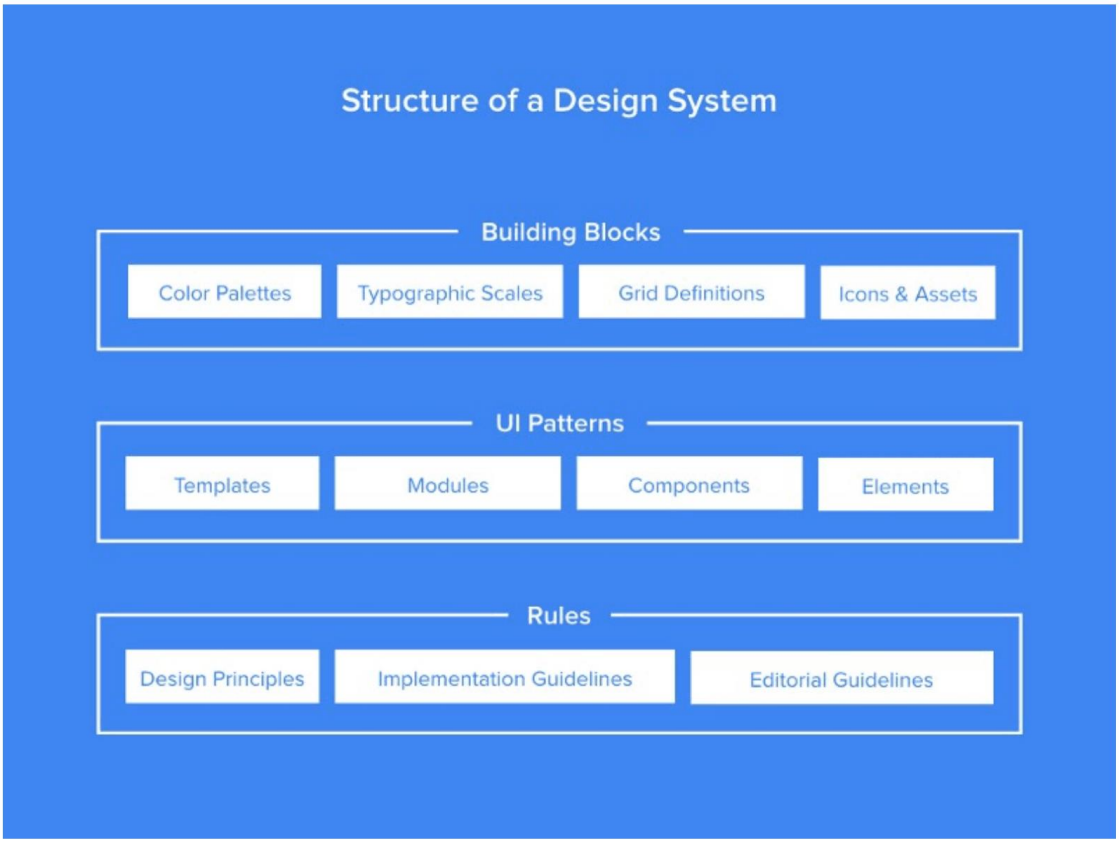

Fig. 8. Example of Design System Structure by UX Pin

A design system is not conceived as a deliverable, but as a set of deliverables and they evolve constantly with the products, tools and new technologies. Design systems are flexible, adapting naturally to changes in the product and synchronizing design and code, for an easier way to create consistent experiences.

One of the strengths of this design solution is its scalability: a design system is a collection of reusable components that can be assembled together to build any number of interfaces, following a set of standards guiding the use of those components.

Futhermore, components of a design system are adaptable in terms of “interfaces-crossing”: they allow to be adopted regardless of interface sizes, device and operative system environments used.

[11] Nathan Curtis “Defining Design Systems.Getting to the Root of What Your System Really Is” (2017) https://medium.com/eightshapes-llc/defining-design-systems-6dd4b03e0ff6

The following design systems represent an example of the widely diffusion of this approach to design:

- *Commercial Design Systems*: Material[12] from Google, Fluent[13] from Microsoft
- *Government Design Systems*: US Government Design Standards[14]

Considering the potential opportunities offered by a system design, that allow designing systems of components in a very flixible, scalable, adaptive and consistent way, an important aspect to evaluate, in order to usefully achieve the purposes of the present deliverable, is the identification of a robust methodology to adopt.

In 2013 Brad Frost introduced the **Atomic Design**[15] as a design system methodology.

The Atomic Design Systems methodology considers all the details that contributing to create and maintaining robust design systems using chemistry metaphor.

The starting point for Frost is the consideration that interfaces are made up of smaller components. His basic conceptual operation consisted in breaking entire interfaces down into fundamental building blocks and work up from there.

Atomic design is composed of five distinct stages working together to create interface design systems in a more deliberate and hierarchical manner.

Fig. 9. Structural components of Atomic Design Methodology

There are five distinct levels in atomic design:

- *Atoms*: parts of an interfaces serve as the foundational building blocks that comprise all user interfaces. These atoms include basic HTML elements like form labels, inputs, buttons, and others that can't be broken down any further without ceasing to be functional;
- *Molecules*: are relatively simple groups of UI elements functioning together as a unit (i.e. a form label, search input, and button can join together to create a search form molecule);
- *Organism*: are relatively complex UI components composed of groups of molecules and/or atoms and/or other organisms. These organisms form distinct sections of an interface;
- *Templates*: Templates are page-level objects that place components into a layout and articulate the design's underlying content structure. They focus on the page's underlying content structure rather than the page's final contents. Design systems must account for the

[12] https://material.io/guidelines/#

[13] https://fluent.microsoft.com/

[14] https://standards.usa.gov/

[15] http://atomicdesign.bradfrost.com/

dynamic nature of content, so it's very helpful to articulate important properties of components like image sizes and character lengths for headings and text passages;

- *Pages*: are specific instances of templates that show what a UI looks like with real representative contents in place. Pages are essential for testing the effectiveness of the underlying design system.

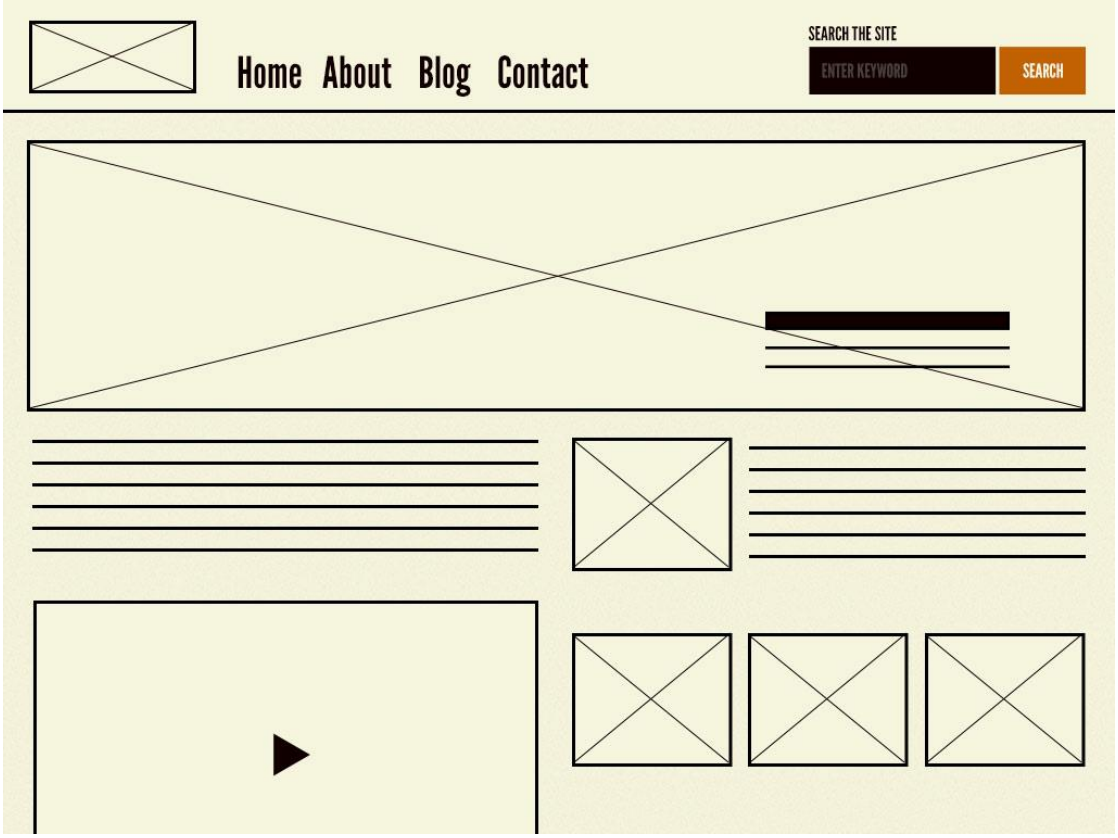

Fig. 10. Example of web page wireframe composed using Atomic Design Methodology

As emerged, rather than a linear process, Atomic Design is a mental model considered helpful for constructing a user interface design system as both a cohesive whole and a collection of parts at the same time.

4.3.3. An "Accessible Design System" for the EASYTV Catalogue

Considerations resulting from the overview of design trends, combined with consequent focus on design system as a widespread adopted tool to create interfaces, have led the design approach to the set up and implementation of the EasyTV Catalogue in the present mid-term report.

A design system represents a suitable design solution for the Catalogue for different reasons:

- *Scalability*: Design systems allow to manage design at scale. Its components are modular and can be reused in different combinations, that allow to build different interfaces solutions using the same basic "design blocks" (definable atoms, molecules, organisms, in Atomic Design methodology);
- *Openness*: it allows to be "open" in absence of functional requirements that cannot be defined at this stage. It evolves with the evolution of the project;
- *Consistency*: it contributes to create a consistent branding, look and feel, and User Experience;

- *Efficiency*: it improves efficiency connecting different teamworks of the project and optimizing timing and resources involved in the stages of design and front-end development;

For the EasyTV Catalogue, to define a design system means considering particularly the dimension of accessibility. Consequently, the design of EasyTV Catalogue will be characterized by this general fundamental requirement.

In conclusion, this section on “design methodology” has led to focusing on design system as a design tool that allows to provide a consistent approach for the EasyTV Catalogue set up. Specifically, starting from its “openness”, it will be possible, in the next design phases, to overcome any effective technical implementation constraints of the service catalogue.

Following, it is proposed to plan a series of activities in order to achieve an **Accessible design system** for the EasyTV catalogue.

Considering the presence of many different teams involved in the services development process of this project (back end/front end), the creation of an Accessible design system is expected to have much wider impact: it will offer the opportunity both to consortium partners and third-party developers (or external software companies) to provide a unique and upgradeable set of guidelines useful for creating coherence between the various EasyTV services and based on a User-Centred approach.

4.4. Development approach

Following preliminary assumptions about the development approach for the EasyTV Service Catalogue are displayed.

This step will support (in the next months) the identification of solutions to adopt in the development phase, during which the implementation of the Service Catalogue project will be arranged.

Starting from the previous considerations regarding the design approach, the main characteristics and parameters foreseen for the Catalogue, and in particular for its **interface development**, are the following:

- Accessibility
- Usability
- Responsiveness
- Modularity
- Openness

As stated, final users will directly interact with the Service Catalogue’s interface: it will allow them to explore services available.

Interfaces strongly impact in the User Experience of the final users. For this reason, User Interfaces (UI) design and Front-End development of the Catalogue are both strategic (and interconnected) phases. They allow to achieve the main goals of the whole EasyTV project.

As claimed in the previous paragraph, referring to the design approach of the Service Catalogue, designing an “Accessible Design System” represents a proper solution considering the requirements emerged.

Similarly, research in the last months focused also on development approach to adopt for finding the best solution.

4.4.1. **Front-end Framework**

Website or web application user interface usually consists of:

- HTML code that is responsible for a structure (information order);
- CSS (Cascading Style Sheets) which main task is to visually format website;
- JavaScript (JS) code, used to incorporate dynamic elements, e.g. slideshows, calculators or expanding menu.

To support the creation of the web-based Service Catalogue that includes the properties emerged, the adoption of a **Front-End Framework** can be considered.

Front-End Frameworks are Web Design Kits consisting of a set of ready components, which allow developers to start a project quickly and efficiently considering fundamental aspects as: accessibility, responsiveness, compatibility for different browsers and customization of the user interface.

Moreover, a front-end framework helps to take into account other elements involved in the development of a website such as code optimization for better performance, scalability and compliance to standards.

Typically, Front-End Frameworks contain the following components:

- a grid which makes it simple to organize the design elements;
- defined font styles and sizing that varies based on its function (different typography for headings versus paragraphs, etc.);
- Pre-built website components like side panels, buttons, and navigation bars.

Since there are many front-end frameworks available nowadays, an overview on the most popular and widespread Front-End Frameworks has been conducted (based on their GitHub popularity[16]):

- Bootstrap
- Semantic UI
- Foundation
- Materialize
- Material UI
- Pure
- UIkit

Each framework has its own strengths and weaknesses, and specific areas of application. As emerged from the analysis, **Bootstrap** is currently the most popular front-end component library.

Its main characteristics are:

1. It is an open source toolkit and it's hosted, developed, and maintained on GitHub.
2. Bootstrap includes HTML and CSS based design templates for typography, forms, buttons, tables, navigation, modals, image carousels, as well as optional JavaScript plugins.
3. The last version of Bootstrap, Bootstrap 4, presents improved responsive features that help to easily and efficiently scale websites and applications with a single code base, from phones to tablets to desktops with CSS media queries.
4. Regarding browser compatibility, Bootstrap 4 is compatible with all modern browsers (Chrome, Firefox, Internet Explorer 10+, Edge, Safari, and Opera).
5. Among the Front-End Frameworks, Bootstrap is the larger in terms of community support and makes available an excellent documentation.

For all these features, Bootstrap 4 (last version of Bootstrap) represents a suitable option for the front-end development of the EasyTV Service Catalogue.

[16] https://github.com/search?o=desc&q=stars%3A%3E1&s=stars&type=Repositories

5. CONCLUSION

In this document has been illustrated the preliminary considerations and indications regarding the set up and the implementation of EasyTV Service Registry and Catalogue emerged during the researching activities conducted in the last 6 months (M6-M12).

In reference to **EasyTV Service Registry** have been:

- described and defined all the constituent elements of the registry;
- presented a wide overview of Docker Registries, explaining how a register works and show a research about what platforms are available on the market and which are their functionalities, advantages and disadvantages;
- identified a specific registry to use: Docker Registry.

Regarding the **EasyTV Service Catalogue** have been:

- presented the main current references regarding Web Accessibility that will lead the design and implementation of the EasyTV Service Catalogue;
- defined a design approach considering the intermediate status of the whole EasyTV project, the global design trends and the main EasyTV system requirements;
- identified solutions to adopt in the development phase, during which the implementation of the Service Catalogue project will be arranged.

The present “mid-term report” will be considered as a starting point and a guidance for activities that will lead to the actual set up and implementation of the EasyTV Service Registry and Catalogue foreseen for M25.

Furthermore, it may be evaluated relevant in the next phases of EasyTV project development.

6. REFERENCES

[1] Configuring a Registry [Online]. Available: https://docs.docker.com/registry/configuration/

[2] Overview of Docker Hub [Online]. Available: https://docs.docker.com/docker-hub/

[3] Artifactory User Guide [Online]. Available: https://www.jfrog.com/confluence/pages/viewpage.action?pageId=46107472

[4] Query Enterprise Documentation [Online]. Available: https://coreos.com/quay-enterprise/docs/latest/

[5] Amazon ECR Developers guide [Online]. Available: https://docs.aws.amazon.com/AmazonECR/latest/userguide/what-is-ecr.html

[6] Container registry Documentation [Online]. Available: https://cloud.google.com/container-registry/docs/

[7] An Overview of Docker Registries [Online]. Available: https://blog.codeship.com/overview-of-docker-registries/

[8] An Overview of Various Docker Registries [Online]. Available: https://www.infoq.com/news/2016/11/overview-docker-registries

[9] Comparing Four Hosted Docker Registries [Online]. Available: https://rancher.com/comparing-four-hosted-docker-registries/

[10] Nelson, H. G., Stolterman, E. (2003). *The design way: Intentional change in an unpredictable world: Foundations and fundamentals of design competence.* Educational Technology**.**

[11] Creating a Design System in UXPin [Online]. Available: https://www.uxpin.com/studio/blog/creating-design-system-uxpin-3-minute-guide/

[12] Design System Handbook [Online]. Available: https://www.designbetter.co/design-systems-handbook/introducing-design-systems

[13] A comprehensive guide to design systems [Online]. Available: https://www.invisionapp.com/inside-design/guide-to-design-systems/

[14] Atomic Design by Brad Frost [Online]. Available: http://atomicdesign.bradfrost.com/