Bloomen

Blockchains in the new era of participatory media experience

HORIZON 2020

762091 – BLOOMEN - H2020-ICT-2016-2
ICT-19-2017 Media and content convergence

D4.8 Bloomen overall integrated system - 2nd cycle

Version:	1.0
Date:	29/02/2020
Authors:	ATC
Type:	Demonstrator
Dissemination level:	Public

Co-funded by the Horizon 2020 programme of the European Union

Table of Contents

1 Introduction

The overall integrated system of Bloomen refers to the fully functional set of Bloomen modules together with the Bloomen platform which fulfils the requirements of Bloomen project. The various functional components of Bloomen are brought together and interact with each other in order to cover the needs of a variety of business workflows. In this deliverable we are covering the methodology and the definition of the approach we have used to integrate and validate the various modules of Bloomen into one single system, updating the information provided in D4.7.

The three use cases of Bloomen project (Music, Photo and WebTV) always play an important role in technical development and integration. Core elements, such as smart contracts are available via an extensive API, which simplifies calling features and functionalities where needed.

Our main focus is still learning about how to put blockchain technology to good use and this is evaluated through the different use cases, that are trying to tackle the limitations of this promising technology.

1.1 About this deliverable

This deliverable is the second cycle of a software prototype. In the course of the Bloomen project this series of deliverables is the specific outcome of Task 4.4 “Overall integration and validation”. The second version of the software prototype is accompanied by this short document, which provides a description of the approach we are using for the integration and validation of the system, providing, also, short, updated, descriptions of the different components.

D4.8 contains all the updates of the components that have been already presented in D4.7 (the first cycle of the software prototype), along with more details for their testing plan.

The next and final iteration of this deliverable, the third cycle, will be delivered in M36.

1.2 Document Structure

D4.8 follows the same structure with D4.7.

- A general description of the Integration Plan and Methodology is provided in Section 2;
- Section 3 presents the different components that are integrated.
- Section 4 contains the conclusions and the plans for the next iteration.

2 Integration Plan and Methodology (Overall)

As already described in D4.7, for the integration purposes of the project we are following the Agile Software Development Practices with frequent integration cycles, rapid prototyping and close collaboration between self-organizing, cross-functional teams. Based on agile principles, we are applying Continuous Integration techniques to perform automated building, testing and deployment of the provided modules, while we set-up a development environment containing a set of continuous integration tools for adopting Continuous Integration practices.

Continuous integration (CI) comprises practices such as daily builds and additional checks so as to prevent bugs. In order to enable automatic daily builds, Continuous Integration software gathers the whole source code in one place (with different revisions), automates the build process and testing, and provides the latest working executable to anyone involved in the project.

The CI model comprises a set of activities for the process implementation: building the system, running tests, deployment activities, and finally reporting test and deployment results. The practice of Continuous Integration assumes a high degree of tests, which are automated into the software: a facility that can be seen as "self-testing code", often using a testing framework.

In the scope of the Bloomen project, we adopted an Agile development process based on the Kanban methodology. The methodology is based on the assumption that all tasks are described as tickets which are assigned to Bloomen team members and are distributed and collected in various columns of a Kanban board. The initial requirements are described as user stories.

The main integration of each pilot with the Bloomen platform is being achieved through the Bloomen platform API. The API provides all the necessary functions for each use case, which are constantly being updated through a shared spreadsheet, and provides an abstraction layer to the complexity of the lower level blockchain functionalities, wherever these are needed.

The User Management and Authentication modules, and all other global functionalities, are provided for all use cases through the shared Admin Panel, which is an extension of the Bloomen platform. Also the pilots will all use the same Ethereum based token called BLO, creating an economy between them. The BLO tokens can be accessed through the wallet functionality of either the mobile app (video use case) or photo use case profile page.

Concerning the testing approach and the release management policy, the components used the information presented in Section 4 of D4.7.

3 System Components

The components that are integrated, are presented in this section. The description of every component is updated and accompanied by the methodology of the integration, the fulfilled requirements and the testing plan.

3.1 Bloomen Wallet

3.1.1 Description and role in Bloomen system

The Bloomen wallet consists of an application for mobile devices that allows interaction with smart contracts hosted in the Alastria Blockchain. The distributed nature of this technology allows us to access functionalities/services offered by smart contracts from any node of the blockchain without having a single point of interaction as in traditional systems.

This component has a specific deliverable within the project (**[M18] D4.3 - Bloomen mobile clients - 1st cycle && [M32] D4.4 - Bloomen mobile clients - 2nd cycle**) where you can find documentation about its functionalities.

3.1.2 Integration with Bloomen platform

This component is integrated with the other modules through Smart Contracts deployed in the blockchain.

3.1.3 Integration Methodology

Ethereum JSON RPC API.

3.1.4 Requirements fulfilled

The requirements fulfilled can be found in Table 1.

ID (from D2.2)	**Name**	**Fulfilled**
RNF-GT-4	Privacy by design Data Storage	X
RNF-GT-8	User Digital Wallet	X
RNF-GT-9	Anonymous Personalization	X
RF-WTV-1	User Registration	X

RF-WTV-2	Create Account	X
RF-WTV-3	Create Wallet	X
RF-WTV-4	View Wallet	X
RF-WTV-7	Purchase Product	X

Table 1: Requirements fulfilled for Bloomen Wallet

3.1.5 Testing Plan

Due to the technical complexity of automating the tests, it has been decided to perform manual test tasks following the specifications of the documents (**D4.3** & **D4.4**) where the software is described.

3.2 Kendraio App

3.2.1 Description and role in Bloomen system

The Kendraio App provides a framework for prototyping copyright management workflows within the Bloomen project. As such the application intends to extend features and functionalities, based on the Bloomen core architecture. The framework will also be developed to be used as a management portal for accessing Bloomen administrative features.

3.2.2 Integration with Bloomen platform

The Kendraio App is a standalone app that integrates via the Bloomen API. The Kendraio App provides bulk import services to the API, the ability to manage users and their permissions, and manipulating metadata for content items.

3.2.3 Integration Methodology

The Kendraio App uses Adapters to provide integration with external systems and services. The current iteration of the Kendraio App uses the Bloomen REST API via HTTP connection. Integration with blockchain components via web3 may be required for the functionality to be developed as part of the management portal.

3.2.4 Requirements fulfilled

The requirements fulfilled can be found in Table 2.

ID	Name	Fulfilled
1	Tag created content	X
2	Define usage terms	X
3	Bloomen API integration	X
4	Copyright management workflows	X
5	Asset management	X
6	Integration with third party APIs	X

Table 2: Requirements fulfilled for Kendraio App

3.2.5 Testing Plan

The testing plan for the Kendraio App aims to ensure correct operation of the application by using manual testing scripts. Sample data was provided by partners and used to confirm the valid operation of workflows within the App.

3.3 REST API

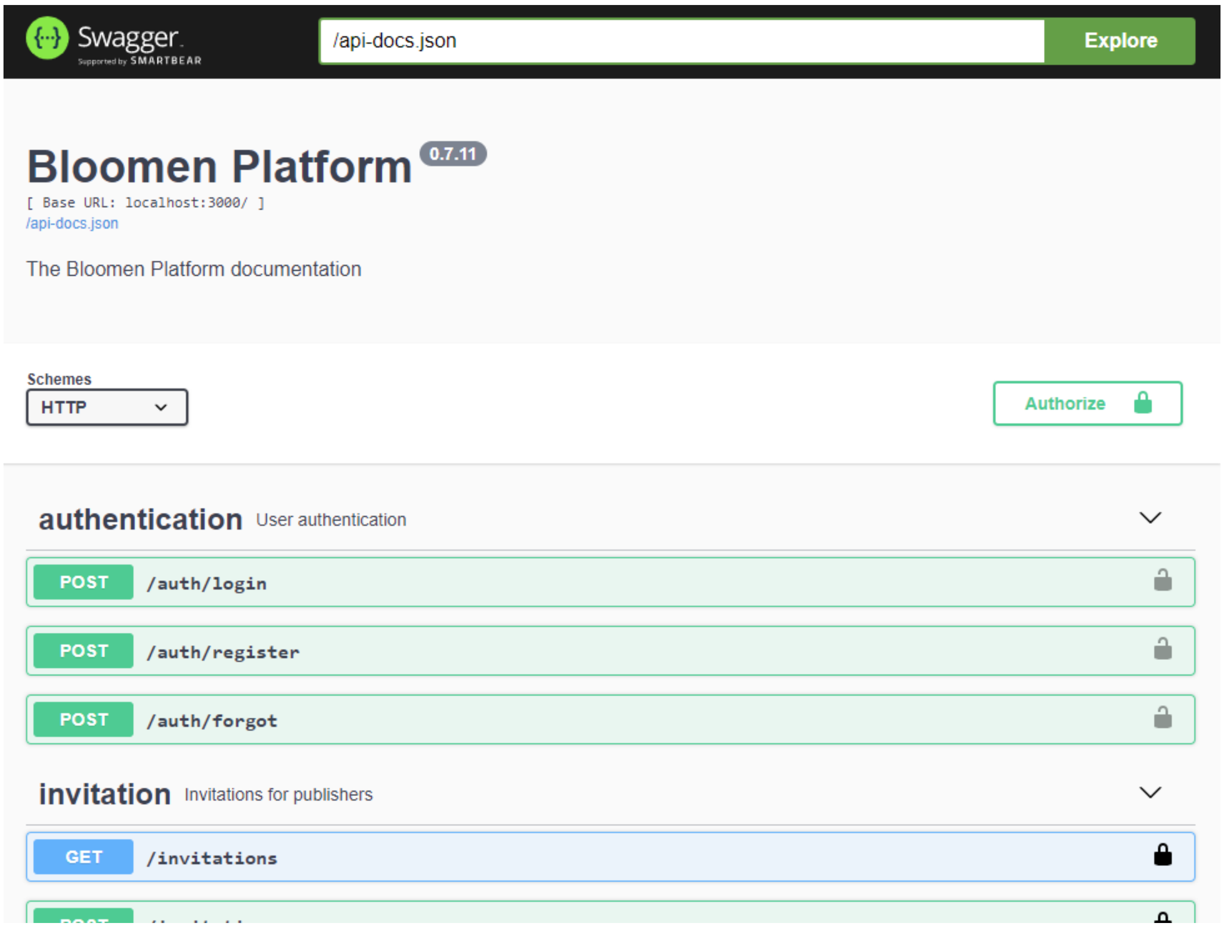
3.3.1 Description and role in Bloomen system

The Bloomen platform API is a RESTful API that allows the different pilots to use basic functionality for each of their cases, such as authentication management, file uploads, etc., and gain access to blockchain operations. The API achieves this without having to interact with the blockchain infrastructure and general complexity, thus making blockchain operations as simple as an HTTP API call. The role of the Bloomen platform API is to glue together the different functionalities for the already created pilots, as well as potential future pilots created by the blockchain or any other communities interested in secure transparent transactions. Further the API removes all the sophisticated knowledge otherwise required by developers to use blockchain technologies. The API replaces these needs through simple and easily customized components. This approach opens the door to fill as many use cases as possible, thus creating a Bloomen ecosystem of applications.

3.3.2 Integration with Bloomen platform

The Bloomen platform API is at the center of the integration effort, as it provides an interface for all the basic functionalities that the pilots require for them to work. Each pilot uses the parts of the Bloomen platform API that it needs for basic functionalities, but is also free to use whatever else backend, or blockchain, it chooses to, to complete its purpose. If pilots need direct access to the smart contract and blockchain functionalities, they can use different tools from the API for direct integration with the blockchain, such as the web3 library.

To use the API, pilots can use the swagger documentation for understanding what functionality each endpoint provides as seen below (Figure 1).



Bloomen Platform 0.7.11

[Base URL: localhost:3000/]
/api-docs.json

The Bloomen Platform documentation

authentication User authentication

POST /auth/login

POST /auth/register

POST /auth/forgot

invitation Invitations for publishers

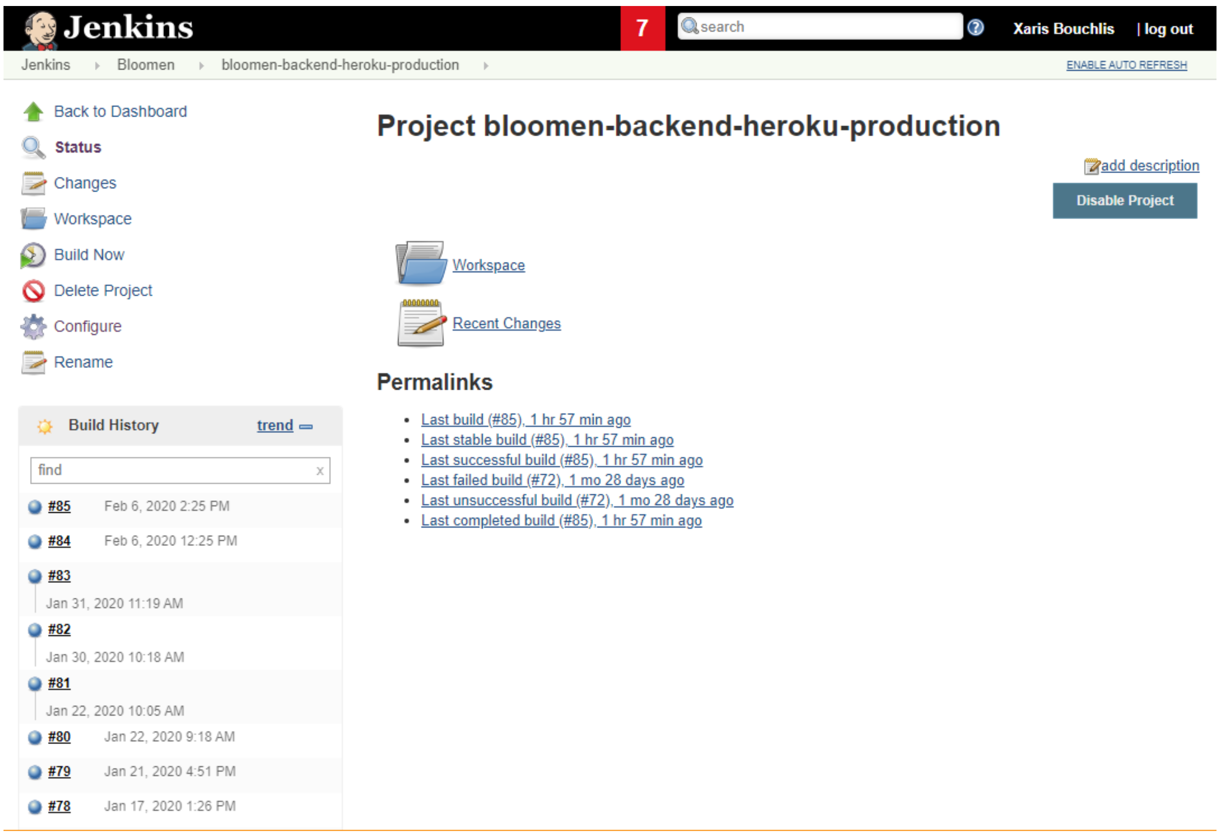
GET /invitations

Figure 1: swagger documentation

3.3.3 Integration Methodology

For the integration purposes of the Bloomen platform we follow the Agile Software Development Practices with frequent integration cycles, rapid prototyping and close collaboration between self-organizing, cross-functional teams. Based on agile principles, we are also applying Continuous Integration techniques for performing automated building, testing and deployment of the provided modules. For adopting Continuous Integration practices we are going to setup a development environment containing a set of continuous integration tools such as the nyc testing framework, Jenkins, etc.

The Jenkins job for building the Bloomen platform backend is shown in the following screenshot (Figure 2).


Jenkins 7 search Xaris Bouchlis | log out

Jenkins › Bloomen › bloomen-backend-heroku-production › ENABLE AUTO REFRESH

Back to Dashboard
Status
Changes
Workspace
Build Now
Delete Project
Configure
Rename

Project bloomen-backend-heroku-production

add description
Disable Project

Workspace
Recent Changes

Permalinks

- Last build (#85), 1 hr 57 min ago
- Last stable build (#85), 1 hr 57 min ago
- Last successful build (#85), 1 hr 57 min ago
- Last failed build (#72), 1 mo 28 days ago
- Last unsuccessful build (#72), 1 mo 28 days ago
- Last completed build (#85), 1 hr 57 min ago

Build History trend

find

#85	Feb 6, 2020 2:25 PM
#84	Feb 6, 2020 12:25 PM
#83	Jan 31, 2020 11:19 AM
#82	Jan 30, 2020 10:18 AM
#81	Jan 22, 2020 10:05 AM
#80	Jan 22, 2020 9:18 AM
#79	Jan 21, 2020 4:51 PM
#78	Jan 17, 2020 1:26 PM

Figure 2: Jenkins for Bloomen

3.3.4 Requirements fulfilled

The requirements fulfilled can be found in Table 3.

ID (from D2.2)	Name	Fulfilled
RF-PH-1	Upload image	X
RF-PH-2	Set image price	X
RF-PH-3	Browse images	X
RF-PH-4	Search for images	X
RF-PH-5	Filter images	X
RF-PH-6	Buy image rights	X

RF-PH-7	Give image rights	X
RF-PH-8	Vote contributor	X
RF-PH-9	View analytics	X
RNF-PH-1	Set privacy settings	X
RNF-PH-2	Workflow efficiency	X
RNF-PH-3	Reliability of payments	X

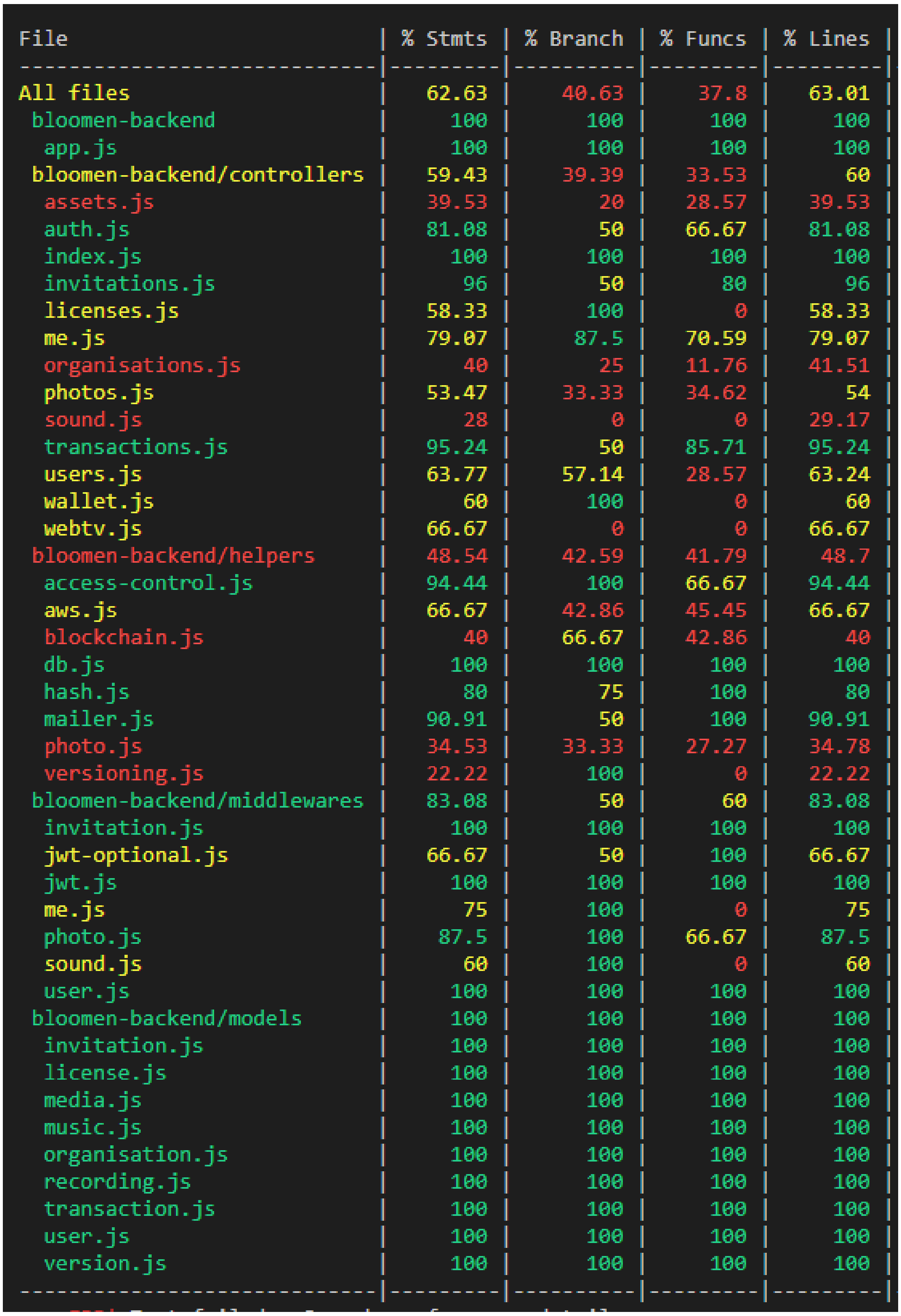
Table 3: Requirements fulfilled for REST API

3.3.5 Testing Plan

The basis of the Bloomen platform API testing is unit testing the different controllers for each route, as well as every functionality and service that can be tested, including the contracts and interactions with the contracts. Since the smart contract functionality is also tested through the individual route and service unit tests, this kind of testing could also be considered integration testing between the Express server and the Smart Contract and blockchain infrastructure.

For unit testing the application, we have used the mocha and chai libraries, as well as the tools provided by the truffle suite, to test the contracts individually. In addition, with these frameworks, we use nyc CLI tool to keep track of test coverage and keep testing relevant, after making changes to the code.

Code test coverage at this point is at about 60% and tests are added continuously.



```
File                          | % Stmts | % Branch | % Funcs | % Lines |
------------------------------|---------|----------|---------|---------|
All files                     |   62.63 |    40.63 |    37.8 |   63.01 |
 bloomen-backend              |     100 |      100 |     100 |     100 |
  app.js                      |     100 |      100 |     100 |     100 |
 bloomen-backend/controllers  |   59.43 |    39.39 |   33.53 |      60 |
  assets.js                   |   39.53 |       20 |   28.57 |   39.53 |
  auth.js                     |   81.08 |       50 |   66.67 |   81.08 |
  index.js                    |     100 |      100 |     100 |     100 |
  invitations.js              |      96 |       50 |      80 |      96 |
  licenses.js                 |   58.33 |      100 |       0 |   58.33 |
  me.js                       |   79.07 |     87.5 |   70.59 |   79.07 |
  organisations.js            |      40 |       25 |   11.76 |   41.51 |
  photos.js                   |   53.47 |    33.33 |   34.62 |      54 |
  sound.js                    |      28 |        0 |       0 |   29.17 |
  transactions.js             |   95.24 |       50 |   85.71 |   95.24 |
  users.js                    |   63.77 |    57.14 |   28.57 |   63.24 |
  wallet.js                   |      60 |      100 |       0 |      60 |
  webtv.js                    |   66.67 |        0 |       0 |   66.67 |
 bloomen-backend/helpers      |   48.54 |    42.59 |   41.79 |    48.7 |
  access-control.js           |   94.44 |      100 |   66.67 |   94.44 |
  aws.js                      |   66.67 |    42.86 |   45.45 |   66.67 |
  blockchain.js               |      40 |    66.67 |   42.86 |      40 |
  db.js                       |     100 |      100 |     100 |     100 |
  hash.js                     |      80 |       75 |     100 |      80 |
  mailer.js                   |   90.91 |       50 |     100 |   90.91 |
  photo.js                    |   34.53 |    33.33 |   27.27 |   34.78 |
  versioning.js               |   22.22 |      100 |       0 |   22.22 |
 bloomen-backend/middlewares  |   83.08 |       50 |      60 |   83.08 |
  invitation.js               |     100 |      100 |     100 |     100 |
  jwt-optional.js             |   66.67 |       50 |     100 |   66.67 |
  jwt.js                      |     100 |      100 |     100 |     100 |
  me.js                       |      75 |      100 |       0 |      75 |
  photo.js                    |    87.5 |      100 |   66.67 |    87.5 |
  sound.js                    |      60 |      100 |       0 |      60 |
  user.js                     |     100 |      100 |     100 |     100 |
 bloomen-backend/models       |     100 |      100 |     100 |     100 |
  invitation.js               |     100 |      100 |     100 |     100 |
  license.js                  |     100 |      100 |     100 |     100 |
  media.js                    |     100 |      100 |     100 |     100 |
  music.js                    |     100 |      100 |     100 |     100 |
  organisation.js             |     100 |      100 |     100 |     100 |
  recording.js                |     100 |      100 |     100 |     100 |
  transaction.js              |     100 |      100 |     100 |     100 |
  user.js                     |     100 |      100 |     100 |     100 |
  version.js                  |     100 |      100 |     100 |     100 |
------------------------------|---------|----------|---------|---------|
```

Figure 3: Bloomen platform API testing

3.4 Smart Contracts

3.4.1 Description and role in Bloomen system

“Smart contracts” are a big innovation of blockchain technologies. The promise here is to reduce the manual work of setting up and the even more complex work of checking the actual transactions versus the fine print agreed on in an analog contract. This is why, in the course of Bloomen smart contracts are considered an important technology element.

Several smart contracts have been built and deployed in the context of the Bloomen System. The PRM Smart Contract is considered responsible for the implementation of all the methods required in order an external web service to interact with the blockchain as well as the data stored within. As a matter of fact, it means that this contract manages the sale and purchase of photos along with their ownership rights. On the other hand, PRM Token is an example ERC223 that interacts with the PRM Smart Contract to fulfill all the payment procedure of the blockchain transactions.

Furthermore, there are smart contracts, developed in order to facilitate the functionalities for the various components of the Bloomen. More specifically these smart contracts are divided into two categories, the smart contracts for music asset conflict resolution and the smart contracts, for the KYC implementation. As far as the smart contracts for the Music assets resolution is concerned, these smart contracts are responsible for storing, retrieving updating music asset claims for ownership and more importantly they contain functionalities in order to detect any conflicts in the music asset claims. As for the KYC smart contracts, they are responsible for the CRUD operations of the KYC information of the user. These smart contracts are able to store and retrieve important information of the users that wish to be or are KYC approved, maintaining in the process the anonymity of the users.

3.4.2 Integration with Bloomen platform

All contracts are written in Solidity and are deployed on Quorum instance running inside the Alastria ecosystem. All the communication between blockchain and other components is implemented in JavaScript through web3.js library. Although it is highly recommended to use the Bloomen API by all external applications, there is an ability to interact directly with the blockchain by using web3.js or a similar library. Developers are not encouraged to do so, but, nevertheless, it is supported by the platform.
More details about the PRM smart contracts can be found in the open-source GitHub repository: https://github.com/atcilab/Smart-contract.

As for the music asset conflict resolution the smart contracts can be found: https://github.com/bloomenio/bloomen-decentralized-rights-management-v2/tree/master/truffle/contracts/registry

The KYC smart contracts can be found : https://github.com/bloomenio/kyc-contracts

3.4.3 Integration Methodology

For the integration purposes of the contracts in this project, the Agile Software Development Practices were followed. This methodology requires frequent integration cycles, rapid prototyping and close collaboration between self-organizing, cross-functional teams. In particular, Kanban methodology was adopted in this component. The methodology is based on the assumption that all tasks are described as tickets which are assigned to Bloomen team members using trello platform and are distributed in the various columns of a Kanban board. The initial requirements are described as user stories that typically follow the following template: As a < type of user >, I want < some goal > so that < some reason > e.g. "As a consumer, I want to search uploaded photos so that I can find the right one.". The lifecycle of development was split in 2-weeks iterations, named Sprints. Each Sprint contains a board where tasks were organized between "TODO", "In Progress", "To Test" and "Done" columns.

3.4.4 Requirements fulfilled

The requirements fulfilled can be found in Table 4.

ID (from D2.2)	Name	Fulfilled
RF-PH-1	Upload image	X
RF-PH-2	Set image price	X
RF-PH-3	Browse images	X
RF-PH-4	Search for images	X
RF-PH-5	Filter images	X
RF-PH-6	Buy image rights	X
RF-PH-7	Give image rights	X
RF-PH-8	Vote contributor	X

RF-PH-9	View analytics	X
RNF-PH-1	Set privacy settings	X
RNF-PH-2	Workflow efficiency	X
RNF-PH-3	Reliability of payments	X

Table 4: Requirements fulfilled for Contracts

3.4.5 Testing Plan

The testing plan for the contracts aims to ensure that the produced output will meet all defined requirements regarding stability, performance, safety and business needs. The areas of software testing covered by Bloomen contract are the Unit testing and Beta software testing. In particular, Unit testing was conducted by using the Truffle environment that offers a dedicated testing framework. The Truffle testing framework offers an automated process while ensuring that a fresh set of contracts is used during each process. An additional benefit is the ability to review contracts while coding in Solidity, throughout the development stage. In order to track bugs affecting the business functionality, we used the Atlassian Jira platform. As mentioned earlier, the software development process followed the agile methodology based on sprints. The development and testing teams worked closely together with the domain experts collaborate through Jira to monitor the progress of development. Any issues or malfunctions were reported and addressed, in order to maintain a high product quality.

3.5 Anonymisation Module

3.5.1 Description and role in Bloomen system

The Anonymization Module offers privacy preserving techniques, personalization and identity management over blockchain enabled media delivery platforms. This module mainly focuses on the distribution and decentralization of the identity management functionality including anonymity and personalization functionalities as well. Also it provides functionalities in order to enable the KYC approval of the users that wish to use the Bloomen platform. This module also implements specific techniques and methodologies in order to preserve the anonymity of the user despite by implementing both private and public smart contracts in the Quorum blockchain.

More specifically users are able to commit all the files needed in order to be KYC approved. These files are stored in a decentralized IPFS storage which is accessed only by the KYC admins. After the full evaluation of the files, the KYC approval is triggered by the admins. In order to ensure anonymity, only a strict amount of information is accessible through the public smart contracts deployed in the Quorum Blockchain.

The main prototype of this module is described in deliverables D3.4 - Anonymous personalization services - 1st cycle and D3.5 - Anonymous personalization services - 2nd cycles and finally the work is concluded in D3.6 - Anonymous personalization services - 3rd cycles.

3.5.2 Integration with Bloomen platform

The integration with the Bloomen platform is simplified by using the Bloomen API, enabling connections between different modules and across current and future use cases.

3.5.3 Integration Methodology

The Anonymization Module is communicating with the different components of Bloomen Platform through the Bloomen API. For example, the communication with smart contracts deployed on Quorum blockchain platform is based on Bloomen API.

3.5.4 Requirements fulfilled

The requirements fulfilled can be found in Table 5.

ID (from D2.2)	Name	Fulfilled
RNF-MUS-1	Privacy Policies	X

RNF-GT-1	Privacy by design Logic	X
RNF-GT-2	Privacy by design Data Anonymization	X

Table 5: Requirements fulfilled for Anonymization module

3.5.5 Testing Plan

A first step in the testing process of the integration-connection of Anonymization service with Bloomen Platform, is the use of the Anonymization Service in the WebTV pilot. Additionally, our aim is to ensure that the produced output will meet all defined requirements, which are previously presented.

3.6 Management portal

3.6.1 Description and role in Bloomen system

The Management Portal provides access to an admin user interface for common functionality across the whole Bloomen platform. Requirements that are related to administration of the system or management of the pilots are to be implemented here.

Features have been developed as part of the Kendraio App to provide user interfaces for the requirements. These interfaces are created as workflows within the App, which integrate with the rest of the Bloomen system via the API.

3.6.2 Integration with Bloomen platform

The implementation of the management portal relies on integration with the Bloomen platform by connecting into the Bloomen API. It builds on top of already implemented functionality to provide a user interface for management of the Bloomen platform.

For the user interfaces that make up the management portal functionality, workflows are created as part of an Adapter for the Kendraio App. Many workflows have been created, with the main flows relevant to the portal functionality being the following:

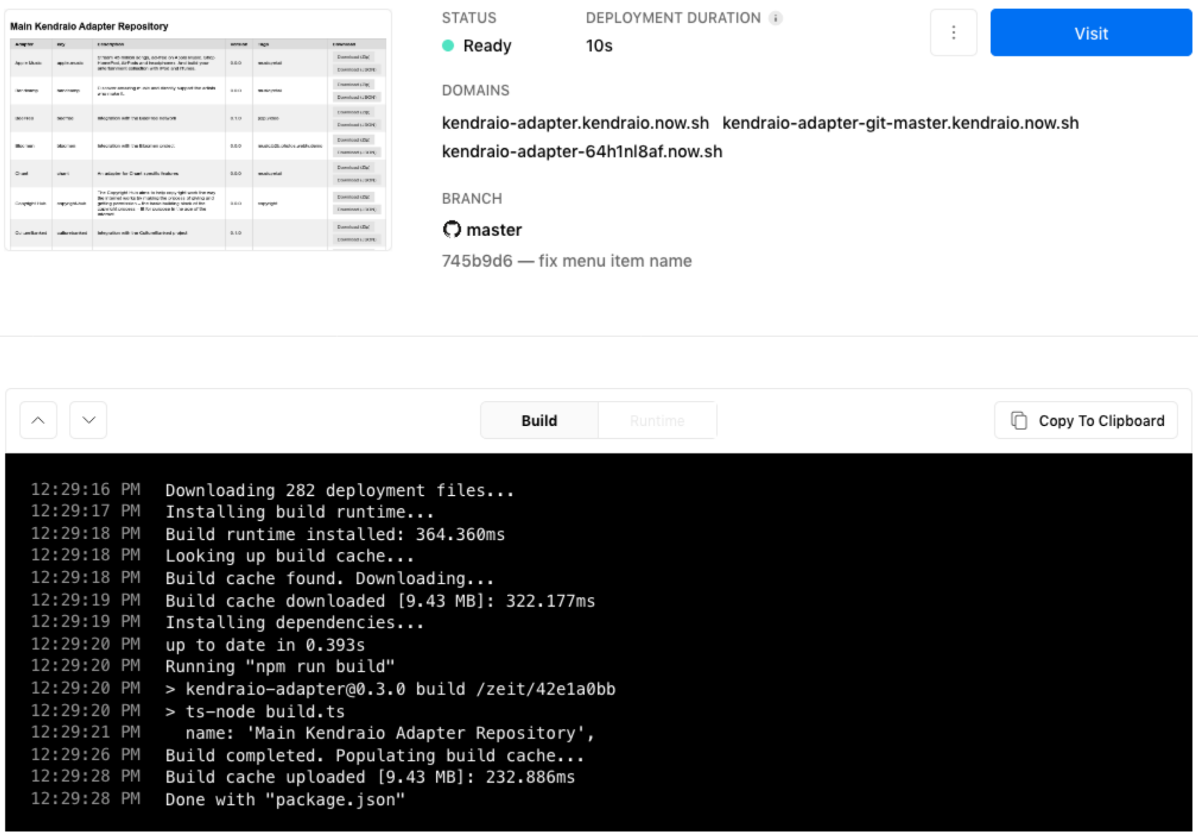
- musicalWorks.json - manage musical works via the API
- soundRecordings.json - manage sound recordings via the API
- bulkImport.json - convert and upload a bulk data file using the batch processing features of the API
- editOrganisation.json - edit an organisation (used to inform content grouping)
- editSoundRecording.json - update metadata for a sound recording
- editUser.json - edit a user's metadata
- editWork.json - edit a musical work's metadata
- exportWorks.json - download and create an export file from musical works present in the API data
- importFromFile.json - import a single data file
- importMusicalWorks.json - import musical works data
- importRecordings.json - import sound recordings data
- listGroups.json - list the available content groups (used by import features)
- Login.json - connect the App to the API and generate authentication credentials
- Register.json - create a new user record
- Search.json - search for content within the Bloomen system
- Users.json - list all users
- workVersions.json - view all revisions of a content item and restore previous versions

3.6.3 Integration Methodology

All functionality will be present and addressed in the API. The Bloomen Adapter contains the configuration that integrates this with the management portal features. The Adapter is managed as part of an Adapter Repository, where configurations can be version controlled, and published as part of an automated build process.

Changes to configuration are exported and committed to the adapter repository. A continuous integration system performs a build and deploys the packaged Adapter. This can then be installed into any running version of the App by refreshing the adapter list and updating the Adapter.

The Adapter configuration repository is built on every commit to the GitHub repo using integration with Zeit Now, as shown below:



Main Kendraio Adapter Repository

STATUS
Ready

DEPLOYMENT DURATION
10s

DOMAINS
kendraio-adapter.kendraio.now.sh kendraio-adapter-git-master.kendraio.now.sh
kendraio-adapter-64h1nl8af.now.sh

BRANCH
master
745b9d6 — fix menu item name

```
12:29:16 PM  Downloading 282 deployment files...
12:29:17 PM  Installing build runtime...
12:29:18 PM  Build runtime installed: 364.360ms
12:29:18 PM  Looking up build cache...
12:29:18 PM  Build cache found. Downloading...
12:29:19 PM  Build cache downloaded [9.43 MB]: 322.177ms
12:29:19 PM  Installing dependencies...
12:29:20 PM  up to date in 0.393s
12:29:20 PM  Running "npm run build"
12:29:20 PM  > kendraio-adapter@0.3.0 build /zeit/42e1a0bb
12:29:20 PM  > ts-node build.ts
12:29:21 PM    name: 'Main Kendraio Adapter Repository',
12:29:26 PM  Build completed. Populating build cache...
12:29:28 PM  Build cache uploaded [9.43 MB]: 232.886ms
12:29:28 PM  Done with "package.json"
```

Figure 4: Continuous Integration build of the Adapter Repository

3.6.4 Requirements fulfilled

The requirements fulfilled can be found in Table 6.

ID (from D2.2)	Name	Fulfilled
RF-MUS-1	Asset Registration Manager	X
RF-MUS-2	Batch Registration Manager	X
RF-MUS-3	Asset Search	X
RF-MUS-4	Assets Explorer	X
RF-MUS-5	Changelog	X
RF-MUS-6	Notification System	Feature moved to Music App
RF-MUS-7	Duplicate Search	Partially fulfilled
RF-MUS-8	Core Metadata Editing	X
RF-MUS-9	Edit Request Management	Feature dropped*
RF-MUS-10	Link Assets	Feature dropped*
RF-MUS-11	Merge Assets	Feature dropped*
RF-MUS-12	Merging Request Management	Feature dropped*

Table 6: Requirements fulfilled for Management Portal

* With regards to RF-MUS-9/10/11/12, these features have been dropped from the management portal as it was realised that these features would have necessitated access by users without access privileges to the management portal and hence, these features would have been more appropriate to being included within the music app directly. However, these features were not finally supported by Bloomen's blockchain-based technology, being considered low priority with respect to others in

favour of focusing on conflict management over asset management within the app. So there was nothing special to "pilot" here in the context of this blockchain project.

3.6.5 Testing Plan

During the development of the workflows for the functionality of the management portal, testing was performed using manual testing scripts. These testing scripts mirrored the requirements being implemented.

The correct configuration for the workflows was proved by checking the operations performed in the user interface caused the appropriate changes to have been made in the data, as accessed via the API.

3.7 Decentralized Rights Management Tool

3.7.1 Description and role in Bloomen system

The second iteration of this component optimally exploited the full range of capabilities developed in first iteration, **D3.2**, and managed to fully embrace the complete spirit and logic of the functional requirements of **D2.2** by offering copyright management of music assets in a decentralized environment. The basic idea of having a shared ledger between partners that raises up claim conflicts once detected by a smart contract comes into fruition in this iteration of the tool.

This component has more documentation in other deliverables within the project (**Demo 3 of D3.3**) where you can find information about its functionalities. In a nutshell, the core elements of this application are described in the following functionalities:

1. Music assets claims conflict detection, broadcasting to respective members and resolving when claims value such as dates, territories, split percentage are modified by the users (in the form of a 'Claim Update').
2. Scalable music asset claims uploading in one click through a CSV type of file.
3. Utilization of more than one CMO accounts.
4. Connection with Bloomen API in order to fetch and submit music asset data.

3.7.2 Integration with Bloomen platform

All smart contracts are written in Solidity and deployed on the public-permissioned blockchain network of Alastria ecosystem. All data exchange and requests between blockchain and other components are implemented in JavaScript, Typescript and Angular through web3.js library. Moreover, there is an ability to interact directly with the blockchain by using web3.js or a similar library. Developers are not encouraged to do so, but, nevertheless, it is supported by the platform. Furthermore, the front end is mostly developed in Typescript and Angular so it merges entirely with the Bloomen API.

More details about the tool source code can be found in the corresponding open-source GitHub repository bloomenio/bloomen-decentralized-rights-management-v2 .

3.7.3 Integration Methodology

As of 3.4.3 Smart Contract Integration Methodology being a part of this current iteration the exact integration methodology was followed here as well for the whole development of the Decentralized Rights Management Tool (see 3.4.3).

3.7.4 Requirements fulfilled

The requirements fulfilled can be found in Table 7.

ID (from D2.2)	Name	Fulfilled
RF-MUS-3	Asset Search	X
RF-MUS-4	Assets Explorer	X
RF-MUS-6	Notification System	X
RF-MUS-7	Duplicate Search	X
RF-MUS-13	Claiming	X
RF-MUS-14	Claims Explorer	X
RF-MUS-15	Conflict Resolution	X

Table 7: Requirements fulfilled for the Decentralized Rights Management Tool

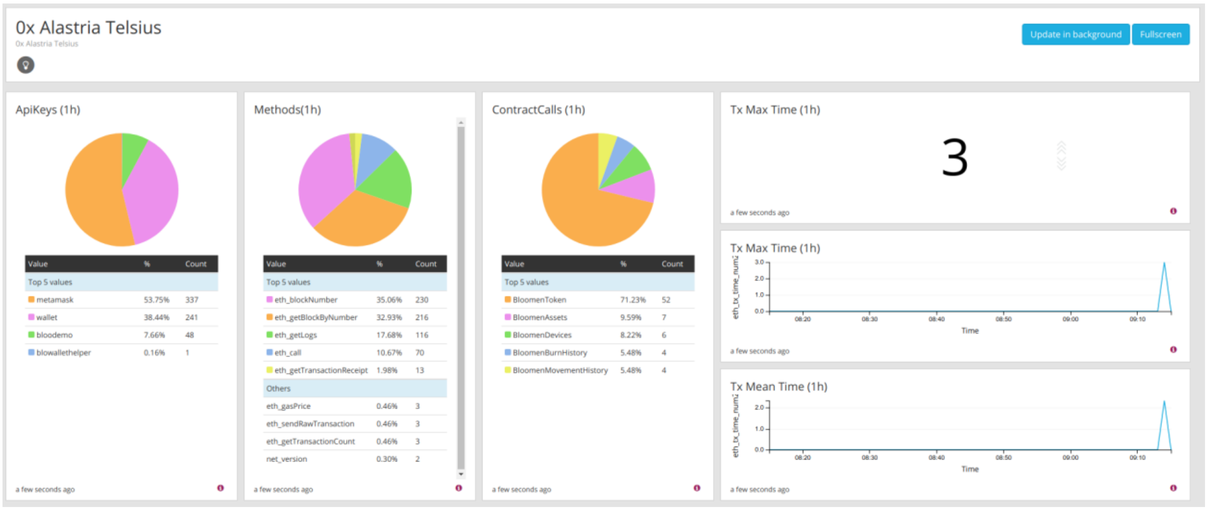
3.7.5 Testing Plan

The testing plan for the claim conflict and music assets copyright resolution aspires to make sure that the presented output meets all pre-defined requirements concerning scalability, performance, security and business. The quality of the tool's software is guaranteed by well-known methods of software organization and testing. Specifically, Truffle Suite constituted a crucial testing framework for smart contracts development. deployment, testing and review, since it provides these processes in an organized and automated procedure. Furthermore, Webstorm, the JavaScript IDE, was the core platform for managing and testing the whole full-stack (Javascript, Typescript, Angular, HTML, CSS, Solidity) application. An agile methodology based on sprints made the development of the tool easier to manage, review and address any issues and maintain its quality.

3.8 BaaS Portal - Bloomen developer Portal

3.8.1 Description and role in Bloomen system

This component is a platform that allows us to control and monitorize the access to Quorum nodes offered by Bloomen. In this first iteration we only offer access to mainnet (Telsius), testnet (Arrakis) and show the utilization of both on a per net dashboard.



Value	%	Count
Top 5 values		
🟧 metamask	53.75%	337
🟪 wallet	38.44%	241
🟩 bloodemo	7.66%	48
🟦 blowallethelper	0.16%	1

Value	%	Count
Top 5 values		
🟪 eth_blockNumber	35.06%	230
🟧 eth_getBlockByNumber	32.93%	216
🟩 eth_getLogs	17.68%	116
🟦 eth_call	10.67%	70
🟨 eth_getTransactionReceipt	1.98%	13
Others		
eth_gasPrice	0.46%	3
eth_sendRawTransaction	0.46%	3
eth_getTransactionCount	0.46%	3
net_version	0.30%	2

Value	%	Count
Top 5 values		
🟧 BloomenToken	71.23%	52
🟪 BloomenAssets	9.59%	7
🟩 BloomenDevices	8.22%	6
🟦 BloomenBurnHistory	5.48%	4
🟨 BloomenMovementHistory	5.48%	4

Figure 5: Blockchain Dashboard

This component is documented on a deliverable within the project (**D3.3 - Blockchain transactions and payment basic services - 3rd cycle**) where you can find documentation about its functionalities.

3.8.2 Integration with Bloomen platform

This component is part of the Bloomen platform.

3.8.3 Integration Methodology

Ethereum JSON RPC API.

3.8.4 Requirements fulfilled

This component is a transversal solution to the entire platform and is not related with concrete requirements.

3.8.5 Testing Plan

This software component acts transparently and uses the data collected during the Bloomen pilots. The validation process is directly related to the quality and quantity of data collected in real time.

4 Conclusions-Next Steps

The 2nd cycle of the Bloomen system is presented in this document. We are progressing in the integration process, since several modules are updated and interact successfully with each other in an integrated environment. Their progress is mainly evaluated through the table of the requirements (Tables 1-7), since, in the second cycle, many more requirements, defined in D2.2, have been fulfilled. More details are also given, in this document, concerning the testing of each component. The different components of Bloomen system have been evaluated in the three pilots, which use the components according to their scenario and approach.

In the 3rd cycle we will focus on fully integrating all core modules and cover all business requirements of the project, while, at the same time, we will use pilots' outcomes to ensure a stable and efficient system to be offered to end-users.