D2.10

Development and Testing environment

ABSTRACT

The present document describes the integration activities that will be performed for the integration of the various software components of the InteropEHRate project. The document describes the integration roadmap that is followed, along with the development tools used. In addition, a summary of the integration activities that took place during the first year of the project are depicted.

Delivery Date	18th October 2019
Work Package	WP2
Task	T2.4
Dissemination Level	Public
Type of Deliverable	Other
Lead partner	BYTE

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 826106.

This document has been produced in the context of the InteropEHRate Project which has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826106. All information provided in this document is provided "as is" and no guarantee or warranty is given that the information is fit for any particular purpose.

CONTRIBUTORS

	Name	Partner
Contributors	Chrysostomos Symvoulidis	BYTE
Contributors	François Sevrin	Andaman7
Contributors	Simone Bocca	UniTN
Contributors	Thanos Kiourtis, Argyro Mavrogiorgou, Konstantinos Vidakis	UPRC
Contributors	Francesco Torelli, Alessio Graziani	ENG
Reviewers	Salima Houta, Marcel Klötgen	FRAU
Reviewers	Simone Bocca	UNITN

LOG TABLE

Version	Date	Change	Author	Partner
0.1	2019-08-02	Provided Table of Contents	Chrysostomos Symvoulidis	BYTE
0.2	2019-08-06	Contribution in Chapters 1 and 2	Chrysostomos Symvoulidis	BYTE
0.3	2019-08-09	Contribution in Chapters 1, 2 and ANNEX	Chrysostomos Symvoulidis	BYTE
0.4	2019-09-06	Contribution in Chapter 2	François Sevrin	A7
0.5	2019-09-16	Review and contribution to all sections	Francesco Torelli	ENG
0.6	2019-09-17	Changes and contribution across the document	Dimosthenis Kyriazis	UPRC
0.7	2019-10-02	Contribution in Chapter 2 , Section 2.2.2	Simone Bocca	UniTN
0.8	2019-10-07	Contribution in Chapter 2	Alessio Graziani	ENG
0.9	2019-10-08	Provided deliverable for internal review	Chrysostomos Symvoulidis	BYTE
1.0	2019-10-09	First internal review	Salima Houta,	FRAU

			Marcel Klötgen	
1.1	2019-10-09	Second internal review	Simone Bocca	UNITN
1.2	2019-10-14	Quality review	Argyro Mavrogiorgou	UPRC
1.3	2019-10-16	Final check	Laura Pucci	ENG
1.4	2019-10-18	Addressed final check comments	Chrysostomos Symvoulidis	BYTE
vFinal	2019-10-18	Final version for submission	Laura Pucci	ENG

ACRONYMS

Acronym	Term and definition
CI	Continuous Integration
CD	Continuous Delivery / Continuous Deployment
D2D	Device to Device protocol
FHIR	Fast Healthcare Interoperability Resources
GUI	Graphical User Interface
HAPI	Health API
HCP	Healthcare professional
HDI	Health Data Integration
HR	Health Record
M-D2D-E	Mobile D2D HR Exchange
M-D2D-SM	Mobile D2D Security Management
M-R2D-SM	Mobile R2D Security Management
R2D	Remote to Device protocol
S-HER	Smart Electronic Health Record
S-R2D-SM	Server R2D Security Management
T-D2D-E	Terminal D2D HR exchange
T-D2D-SM	Terminal D2D Security Management
T-R2D-SM	Terminal R2D Security Management
URL	Uniform Resource Locator
VCS	Version Control System
WP	Work Package

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1. Scope of the document

The current document presents the development and integration steps currently planned by the technical partners of the InteropEHRate Consortium, in order to produce the InteropEHRate platform. This deliverable is focused mainly on the description of the development and integration approach that is put in place and followed in the project. The deliverable is described as "Other" because the main delivery points regard the usage of the software repositories provided in the project's git.

1.2. Intended audience

This document is mainly intended for InteropEHRate developers, since it describes the tools and technologies used for the implementation of the various components, along with the integration roadmap agreed by the technical partners in order to realise the integrated InteropEHRate platform.

1.3. Structure of the document

This document is structured as follows:

- Section 1 (this section) describes the goals and structure of the document, and its relation to other reports.
- Section 2 "Integration approach and Hardware infrastructure" provides a description of the approach that is followed regarding the development of the software components of the InteropEHRate platform, including the tools used for this process as well.
- Section 3 "Integration plan" depicts the roadmap that is going to be followed throughout the project in terms of integration efforts and planning.
- "Conclusions and next steps" wraps up this deliverable and presents the next steps that are going to be followed.
- The Annex "First year integration roadmap" presents in a more concrete way how the development and integration steps that are described in the previous chapters are realized during the first year of the project, with references to the project's Gitlab.

1.4. Updates with respect to previous version (if any)

This document describes the development and integration approach of InteropEHRate and the currently planned steps towards the implementation of the overall integrated InteropEHRate platform. According to the described approach as presented in [D2.4], future updates of the planned steps and their status of advancement will be reported on the project issue tracker (as introduced in Section 2.2.5). Details on the implementation of the different components and the planned steps will be documented in the corresponding deliverables of the project.

2. INTEGRATION APPROACH AND HARDWARE INFRASTRUCTURE

2.1. Integration approach

This chapter describes the approach that has been agreed by the technical members of the consortium of the InteropEHRate project, for the development and integration of the components towards the overall InteropEHRate platform. The agreed process for development and integration is described below, while the adopted tools are described in the next sections.

As described in deliverable [D2.1], the starting point of the InteropEHRate development process is a set of user scenarios mainly defined by the final users. Starting from these scenarios, the technical partners agree with the final users on a set of user requirements (implied from the defined scenarios) and on their priority.

Each year, the technical partners review the user requirements and on the base of user priorities and technical constraints they plan the user requirements to be addressed during the year. Each *user requirement* targets the behaviour of a single user application or service with respect to a particular behaviour offered to the final user, as described in deliverable [D2.1].

During the integration and development planning the user requirements are further divided into more granular elements called “functionalities” and “tasks”. A functionality is a fine-grained behaviour or operation offered by a single software component, to be consumed by other components (in this case the functionality may by a method of a library) or by the final user (in this case a functionality may be a portion of offered GUI and related behaviour). Typically each user requirement requires the implementation of several functionalities.

While a user requirement may require one or more years of work to be fully implemented, a functionality may be implemented in a few months of development. A task is a development activity that is needed in order to implement a functionality or a requirement. A task may deliver *a functionality* or *a portion of it*. A task should be sufficiently fine grained to be implemented within a maximum period of 2 weeks, called sprint according to agile methodologies [Scrum]. The sprints are a short period of time, all having the same duration, where the development and integration team focuses on completing a set of work items, similar to the sprints in the agile development methodology. The sprints in the InteropEHRate project have duration of two weeks.

However, apart from these similarities, the InteropEHRate development process cannot be considered an agile process, but just an iterative process where requirements and development tasks are reviewed periodically.

At the beginning of each sprint the developers meet to plan the tasks to be performed to progress the realization of a specific sprint. First of all, the results of the previous sprint are reviewed to confirm their completion. If any task has not been completed yet, it is moved to the new starting sprint or further analysed to understand if it needs to be further split.

The meaning of requirements is also reviewed during this planning phases and if needed further clarifications are asked to the lead of the requirements. On the base of these clarifications and of technical opportunities, the tasks for the new sprint are then fixed. During the following weeks the advance of the agreed tasks is then monitored.

In order to make sure the development process is followed, the project is divided into several Work Packages (WP) that under them, a main goal of the project is set to be fulfilled. Each WP is assigned to a project partner whose responsibility is to coordinate the design and implementation activities within that WP. Consequently, each WP is further divided into Tasks, which is assigned to a project partner as well whose responsibility is the design and implementation of a defined part of the overall WP goal.

In order to clarify the responsibilities, each user requirement is assigned to a technical lead. The technical lead associated to a requirement is accountable for the planning of the features and tasks needed for the implementation of that requirement. While this planning is fixed on a sprint by sprint basis, it is recommended that the technical lead of a requirement prepares a draft global plan specifying the tasks to be implemented in future sprints in order to complete the implementation of the requirement. It is up to the technical lead to decide if to split the requirement in different features.

Each task is assigned to a technical lead, i.e. the person responsible for the completion of the task by the **requirement leader**. The technical lead of a task is also responsible for reporting the status of the task to its requirement leader who is responsible for monitoring the progress of the implementation of the full requirement and to solve integration issues related to that requirement. The lead of integration and development is responsible to assure that the entire integration and development process is correctly understood and executed and that any needed improved to the process is agreed and applied when necessary.

The above-mentioned plan is managed through a GitLab issue tracker (described in Section 2.2.5). As all planning and reporting is managed through the project's issue tracker, it is clear that it is not just a bug tracking tool, but a system responsible to supervise the overall procedure. For the time-being the access is limited to the consortium partners.

The structure of the development and integration approach of InteropEHRate is graphically depicted in Figure 1 below:

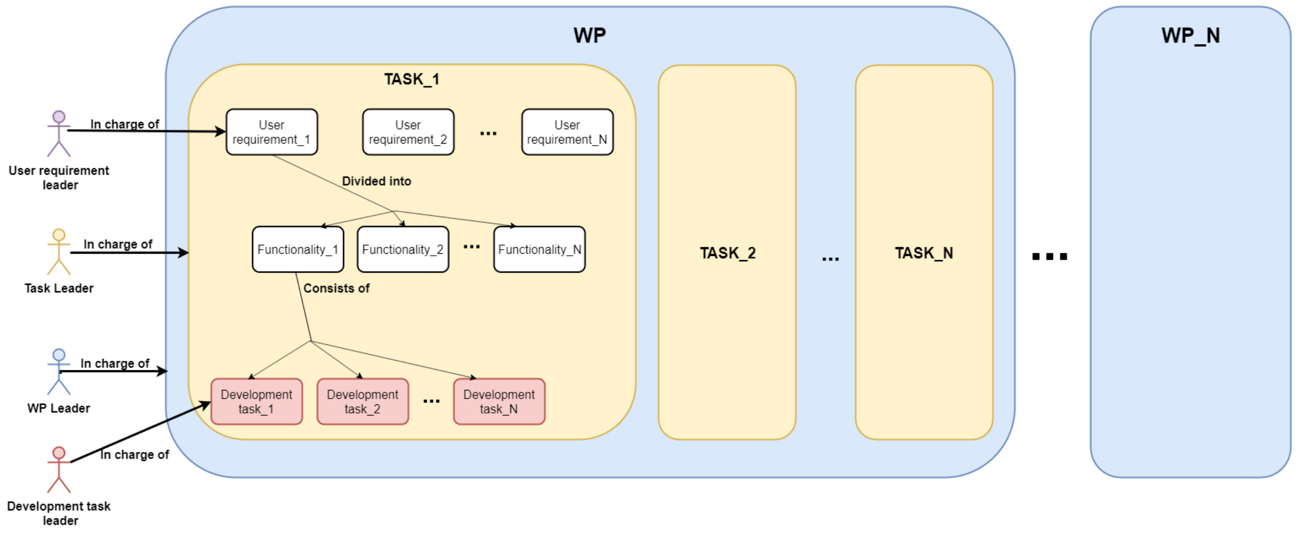


Figure 1 - Roles and responsibilities

Another important notice is that the roadmap (i.e. the plan of which requirements will be targeted and of which tasks will be implemented during each year, sprint by sprint) is updated *continuously*. The target of each year has been set, but adjustments according to the progress may exist. More specifically, the first year is mainly focused on the first scenario (Device to Device HR exchange), the second year will be focused on the second scenario (Remote to Device HR exchange), while the third year is focused on the third scenario (Research protocol). But this does not mean that once a year is over, the corresponding scenario is complete. On the contrary, throughout the project all three scenarios will constantly be updated.

In addition, although it is not strictly part of the implementation process, it is important to note that a *focus group* consisting of representatives of the final users (i.e. patients, nurses, medical doctors and researchers) is set in order to increase the user satisfaction, through the co-design of the GUIs and the functionalities of the InteropEHRate framework.

For better understanding of the integration approach, in the Annex the integration roadmap of the first year is shown.

2.2. Tools and techniques

This chapter describes the tools and the techniques used for the development of the components of the InteropEHRate platform.

2.2.1. Version Control System (VCS)

The consortium has selected GitLab as the main Version Control System (VCS). A private deployment has been concluded during the first year of the project on UPRC premises, and the corresponding URL to the InteropEHRate GitLab instance is the following: http://iehrgitlab.ds.unipi.gr/.

A private "InteropEHRate" group in this GitLab is created which is organized into several sub-groups. Each sub-group regards a macro-component as also presented in the architecture deliverable [D2.4]. Under each group a project that regards a component is created. More specifically the structure of the "InteropEHRate" group is presented below:

- Health tools
  - Data mapping tool
  - Knowledge management tool
- InteropEHRate Research Services
  - Server Research Data Sharing
  - Server Encrypted communication
- InteropEHRate Health Services
  - Machine Translation
  - Label Translation
  - FHIR Export
  - FHIR Import
  - Information Extraction and Data Formalization
  - FHIR Data Mapping
  - Legacy Data import
  - S-EHR Localisation platform
  - S-EHR Integration platform
  - HR Data Index
  - Server R2D HR Exchange
  - Server Encrypted communication
  - R2D Security management
- Reference HCP app
  - HCP App
  - Terminal R2D Security Management
  - Terminal D2D Security Management
  - Terminal Encrypted Communication
  - Terminal R2D HR Exchange
  - Terminal D2D HR Exchange
- S-EHR Cloud
- S-EHR Message Broker
- S-EHR Mobile app
  - S-EHR app
  - R2D Security Management
  - D2D Security Management
  - Encrypted Storage
  - Encrypted Communication
  - R2D HR Exchange
  - D2D HR Exchange
  - Research Data Sharing
  - Anonymization and Aggregation
- Interoperability profile

In addition, two more *projects* are created in order to facilitate on the organization of the development phase: the “issues” project, which is the main project used for the management of the integration and implementation roadmap agreed among all developers and the "Website activities", which is used for tracking down the activities related to the implementation of the project's website. The "issues" project will be further described in Section 2.2.6.

The developer's team have set a list of ground rules that need to be followed in order to have a homogenized result, with respect to the implementation that are also documented in the 'issues' task of the gitlab's server. These rules include:

- The structure of the code repositories
    - It has been agreed that the overall structure of every repository should be the same for homogeneity reasons.
- The naming of the of the packages and the interfaces of each developed component
    - All the components developed for the InteropEHRate project should be placed under a certain package.
- The branches in each repository
    - A "develop" branch is used for the development, where a "master" branch is used for the official releases of the components of the InteropEHRate platform.

2.2.2. Build tools

This chapter presents the tools that are used or will be used throughout the project for the development and integration of the various components. For each component, an exhaustive list of the tools is provided.

2.2.2.1. R2D Protocol

The R2D library for mobile is implemented using the Java programming language and is compatible with the Android operating system. The list of tools used for development contains the following:

- Android Studio 3.4.2;
- Android emulator;
- Android compatible smartphone;
- JUnit for unit testing;
- Gradle and Maven (Nexus) for dependency management.

The R2D server is implemented as a Java application built to run as a Java web application hosted inside a web container. The list of tools used for development includes:

- Eclipse 2019 (application is developed using standard project configuration as defined by Maven and adopted by several IDEs);
- JUnit for testing;
- Maven for dependency management;
- Tomcat as web container and HTTP server;
- Derby as database;
- Jenkins for continuous integration;

2.2.2.2. D2D Protocol

The D2D protocol specifies a series of specified bluetooth messages regarding the information that is being exchanged and healthcare related data between a healthcare practitioner (utilizing a web application (HCP app)) and a citizen (utilizing a mobile application (S-EHR app)), without the usage of internet connection.

In order for the HCP app to use the D2D protocol, the Terminal D2D Exchange (T-D2D-E) component is implemented as a Java library. In more detail, the T-D2D-E is a Maven [Apache Maven] project following the standard maven structure. The pom.xml file contains information about the project and configuration details used by Maven to build the project, containing default values such as the build directory.

- For the development of the T-D2D-E component, Java is currently used as the main programming language.
- For the development and testing of the T-D2D-E component, the Netbeans IDE is used.
- For achieving the Bluetooth connection, as well as the data exchange process, the BlueCove Java library is being used, provided as a Maven dependency in the pom.xml file of the project.
- For the transfer of data (i.e. FHIR Objects), the HAPI-FHIR [HAPI FHIR] library is being used, provided as a Maven dependency in the pom.xml file of the project.

In order for the S-EHR app to use the D2D protocol,the Mobile D2D HR Exchange (M-D2D-E) component is implemented as an Android Java library. In more detail, the M-D2D-E is a Gradle [Gradle] project following the standard Gradle structure. The Gradle file contains information about the project and configuration details used by Gradle to build the project, containing default values such as the build directory.

- For the development of the M-D2D-E component, Java for Android is currently used as a programming language.
- For the development and testing of the M-D2D-E component, the Android Studio IDE is used.
- For the transfer of data (i.e. FHIR Objects), the HAPI-FHIR library is being used, provided as a Gradle dependency in the Gradle file of the project.

2.2.2.3. S-EHR App

The InteropEHRate S-EHR App is embedded in the current Andaman7 app as a module.

- The S-EHR App is written in Java for Android.
- For the development of the S-EHR module, Kotlin for Android [Kotlin-Android] is currently used as the programming language.
- For the development and testing of the S-EHR app the Android Studio IDE is also used.
- For dependency injection Dagger2 [Dagger] framework is being used, provided as a Gradle dependency in the Gradle file of the project.
- The database is managed using Realm Framework for Android [Realm], provided as a Gradle dependency in the Gradle file of the project.
- For the retrieval of the data, the HAPI library is being used, provided as a Gradle dependency in the Gradle file of the project.

2.2.2.4. HCP Application

The HCP application is designed to run on the Java Virtual Machine being developed using Java and Kotlin programming languages having Spring as framework for development. The list of tools used for development and integration is:

- Maven for build automation;
- JUnit for unit testing;
- Selenium for automated testing, HCP being a web application;
- There is no close dependency on a certain Integrated Development Environment. Depending on the preferences of the developers can be used: Intellij IDEA, Eclipse or Spring STS;
- Jenkins for continuous integration;
- Nexus for dependency management.

2.2.2.5. Conversion and Translation tools

The conversion and translation tools rely on an innovative knowledge-based data integration platform, shown as *Health Data Integration (HDI) Platform*. This allows the conversion of local EHR formats to the interoperable S-EHR representation and of their translation into multiple European languages.

The HDI platform, developed in Java using Spring framework and Hibernate, uses different tools, called InteropEHRate Health Tools (IHT) for the conversion and translation process.

- Data Mapping Tool: Developed in Java, it allows the conversion of data between different formats. It allows also to write short Python scripts to manage the data conversions.
- Knowledge Management Tool: Developed in JavaScript - CoffeeScript [CoffeeScript], allows the management of the knowledge of data. This tool provides all the operations needed for importation, definition and description of the knowledge of data.

The database used by the HDI platform is managed using PostgreSQL [PostgreSQL], while another database in support to the Knowledge Management Tool is managed through MongoDB [MongoDB].

Inside the HDI platform the different libraries and software components are used as Maven dependencies included in the pom file of projects.

2.2.2.6. Security libraries

In order for the D2D and R2D protocols to be secure, the Terminal D2D Security Management (T-D2D-SM), Mobile D2D Security Management (M-D2D-SM), Terminal R2D Security Management (T-R2D-SM) and Server R2D Security Management (S-R2D-SM) components are implemented as a Java libraries.

The T-D2D-SM, T-R2D-SM and S-R2D-SM are Maven projects following the standard maven structure. The pom.xml file contains information about the project and configuration details used by Maven to build the project, containing default values such as the build directory.

- For the development of the components, Java is used as a programming language.
- For the development and testing of the components IntelliJ IDEA IDE [IntelliJ IDEA] is used.

- For cryptographic functions java.security and javax.crypto Packages are being used.

The M-D2D-SM and M-R2D-SM are a Gradle projects following the standard Gradle structure. The Gradle file contains information about the project and configuration details used by Gradle to build the project, containing default values such as the build directory.

- For the development of the components, Java for Android is currently used as a programming language.
- For the development and testing of the component Android Studio IDE is used.
- For cryptographic functions java.security and javax.crypto Packages are being used.

2.2.3. Testing tools

InteropEHRate regards a project that various technologies and frameworks co-exist. In order to make sure that the produced software works according to plan several tests are executed. These include unit tests, Android UI tests, Automation tests and integration tests.

The tests on component level (unit tests, Android UI tests, browser automation tests) should be written and executed by the leading developer of the corresponding software and are configured to be executed after a push or commit in the master branch using the GitLab Runners [GitLab Runner]. When it comes to integration tests, these should be written by the leading developers of the software mechanisms to be integrated.

Each one of these tests and the tools used for them are explained in detail in the chapters below.

2.2.3.1. Unit testing

Unit testing regards the software tests that investigates whether a developed components behaves the way it is supposed to. Unit testing is a very important step, since it should be performed before the integration of the component with the rest of the InteropEHRate platform. The responsible for running these tests is the developer or the team responsible for the implementation of a component. Most of the developed applications are written in Java, hence **JUnit** [JUnit] has been chosen as the main tool for Unit testing.

2.2.3.2. Android UI tests

With Android UI tests, the developer can ensure that the functional requirements of developed application are met. This type of test tests the user interactions with the application, and is used to identify unexpected behavior that may lead to a poor user experience. For this reason, **Espresso** [Espresso] has been agreed on using as the primary Android UI testing tool.

2.2.3.3. Browser automation testing

The software applications in the InteropEHRate project are/will be written as web applications. For this reason, tests that will assess the interaction and responsiveness of the application with the user is necessary. Thus, **Selenium** [Selenium] has been selected as the automation testing tool used for such applications, like the reference HCP app.

2.2.3.4. Integration / System testing

This type of test is used to ensure that the individually developed components of the InteropEHRate platform work well in combination. Integration tests can expose bugs in the *interfaces* and in the *interaction* between integrated components and / or systems. Similarly, system tests will also be written for cases where developed components of the InteropEHRate platform communicate with external services (e.g. Healthcare Organization Information Systems).

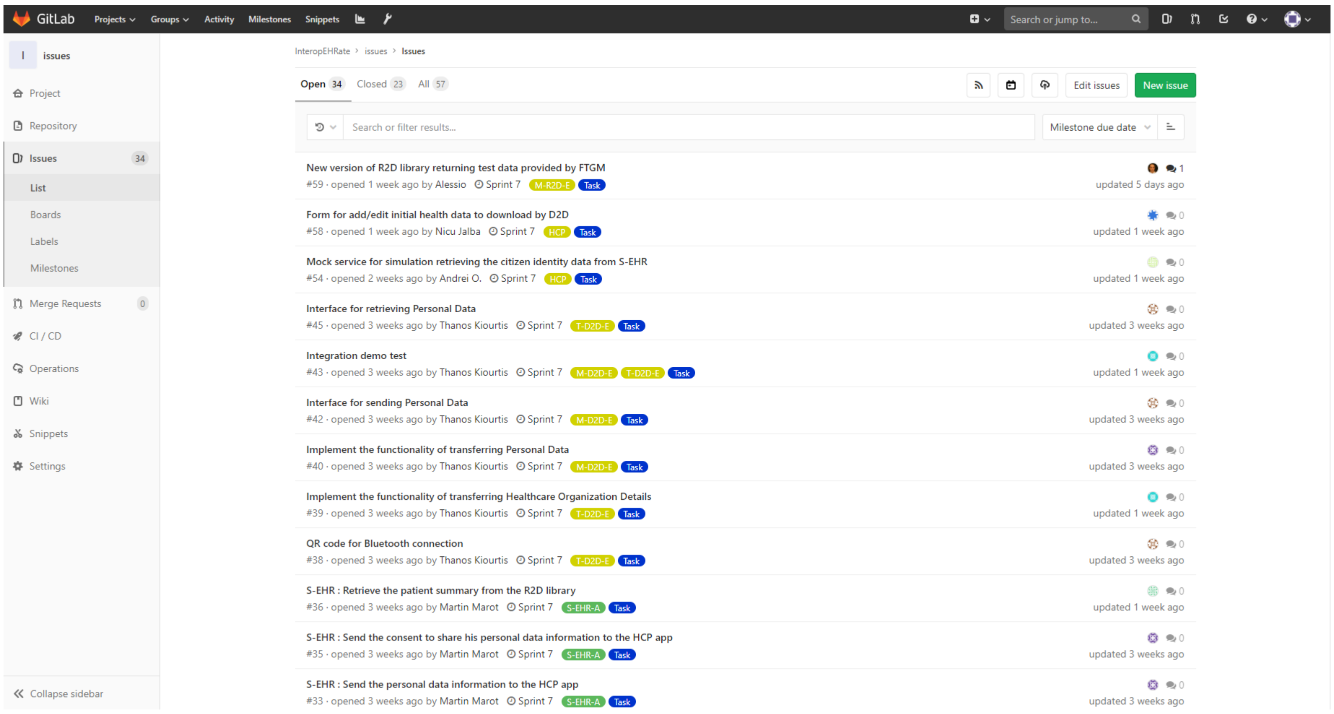
2.2.4. Continuous integration

The deployment of the InteropEHRate platform is based on a Continuous Integration (CI) process. Continuous Integration regards the software engineering practice, with which the developed software can be released on any occasion, through the production of code in fixed time periods. Continuous Integration is considered to be a best practise for the automation of software integration, and is comprised of a set of techniques that ensure the quality of the deployed product. With Continuous Integration, errors can be identified rapidly and integration issues are minimized.

The Continuous Integration approach in the InteropEHRate project starts with committing the code to GitLab. The code is then built automatically using Jenkins [Jenkins]. Once the build is complete, a set of automated unit tests, which are under the responsibility of each component's developer, are triggered. If the tests are successful the code is pushed to the master branch, which in our case represents the versioning branch.

2.2.5. Issue tracking

As already mentioned, GitLab issue tracker is the toolset that the InteropEHRate project uses. It is located in the private "InteropEHRate" group of GitLab as described in Section 2.2.1. A screenshot of the issue tracker is depicted in Figure 2:



InteropEHRate › issues › Issues

Open 34 Closed 23 All 57

Search or filter results...

New version of R2D library returning test data provided by FTGM
#59 · opened 1 week ago by Alessio Sprint 7 M-R2D-E Task — 1 — updated 5 days ago

Form for add/edit initial health data to download by D2D
#58 · opened 1 week ago by Nicu Jalba Sprint 7 HCP Task — 0 — updated 1 week ago

Mock service for simulation retrieving the citizen identity data from S-EHR
#54 · opened 2 weeks ago by Andrei O. Sprint 7 HCP Task — 0 — updated 1 week ago

Interface for retrieving Personal Data
#45 · opened 3 weeks ago by Thanos Kiourtis Sprint 7 T-D2D-E Task — 0 — updated 3 weeks ago

Integration demo test
#43 · opened 3 weeks ago by Thanos Kiourtis Sprint 7 M-D2D-E T-D2D-E Task — 0 — updated 1 week ago

Interface for sending Personal Data
#42 · opened 3 weeks ago by Thanos Kiourtis Sprint 7 M-D2D-E Task — 0 — updated 3 weeks ago

Implement the functionality of transferring Personal Data
#40 · opened 3 weeks ago by Thanos Kiourtis Sprint 7 M-D2D-E Task — 0 — updated 3 weeks ago

Implement the functionality of transferring Healthcare Organization Details
#39 · opened 3 weeks ago by Thanos Kiourtis Sprint 7 T-D2D-E Task — 0 — updated 1 week ago

QR code for Bluetooth connection
#38 · opened 3 weeks ago by Thanos Kiourtis Sprint 7 T-D2D-E Task — 0 — updated 3 weeks ago

S-EHR : Retrieve the patient summary from the R2D library
#36 · opened 3 weeks ago by Martin Marot Sprint 7 S-EHR-A Task — 0 — updated 1 week ago

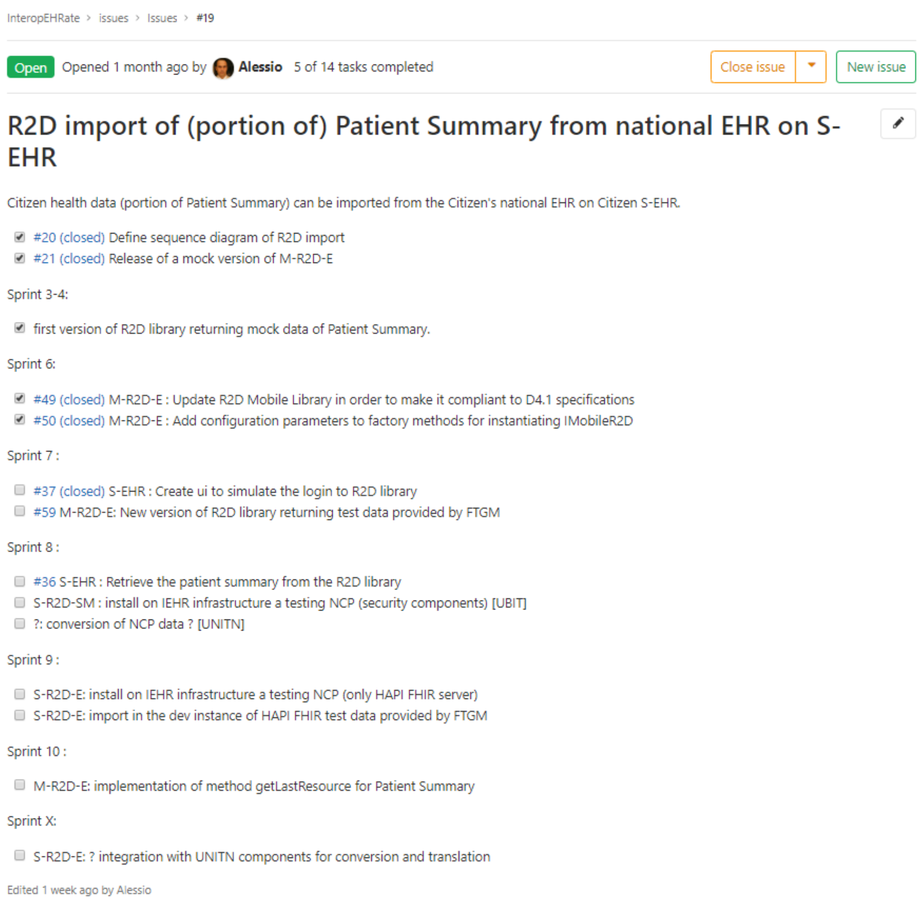
S-EHR : Send the consent to share his personal data information to the HCP app
#35 · opened 3 weeks ago by Martin Marot Sprint 7 S-EHR-A Task — 0 — updated 3 weeks ago

S-EHR : Send the personal data information to the HCP app
#33 · opened 3 weeks ago by Martin Marot Sprint 7 S-EHR-A Task — 0 — updated 3 weeks ago

Figure 2 - GitLab issue tracker

In order to better manage the issues, a list of labels is created and used to categorize the issues. These labels include:

- **The "Requirement" label**: An issue tagged as "requirement" represents a *user requirement* that needs to be implemented. In the description of each requirement the corresponding Tasks that form it are presented, divided into their corresponding sprints. A checked task in the description of a requirement, suggests that it is complete. An example "Requirement" is presented in Figure 3:



InteropEHRate › issues › Issues › #19

Open Opened 1 month ago by Alessio 5 of 14 tasks completed

Close issue New issue

R2D import of (portion of) Patient Summary from national EHR on S-EHR

Citizen health data (portion of Patient Summary) can be imported from the Citizen's national EHR on Citizen S-EHR.

- [x] #20 (closed) Define sequence diagram of R2D import
- [x] #21 (closed) Release of a mock version of M-R2D-E

Sprint 3-4:

- [x] first version of R2D library returning mock data of Patient Summary.

Sprint 6:

- [x] #49 (closed) M-R2D-E : Update R2D Mobile Library in order to make it compliant to D4.1 specifications
- [x] #50 (closed) M-R2D-E : Add configuration parameters to factory methods for instantiating IMobileR2D

Sprint 7 :

- [ ] #37 (closed) S-EHR : Create ui to simulate the login to R2D library
- [ ] #59 M-R2D-E: New version of R2D library returning test data provided by FTGM

Sprint 8 :

- [ ] #36 S-EHR : Retrieve the patient summary from the R2D library
- [ ] S-R2D-SM : install on IEHR infrastructure a testing NCP (security components) [UBIT]
- [ ] ?: conversion of NCP data ? [UNITN]

Sprint 9 :

- [ ] S-R2D-E: install on IEHR infrastructure a testing NCP (only HAPI FHIR server)
- [ ] S-R2D-E: import in the dev instance of HAPI FHIR test data provided by FTGM

Sprint 10 :

- [ ] M-R2D-E: implementation of method getLastResource for Patient Summary

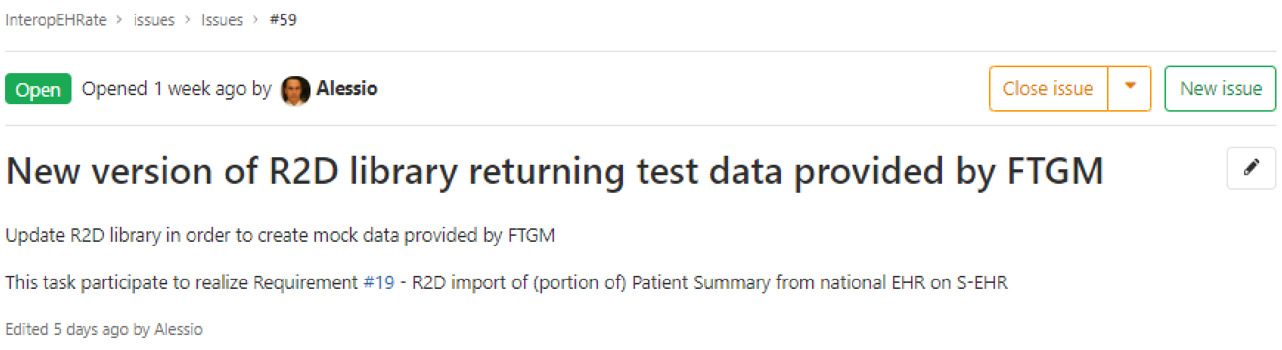
Sprint X:

- [ ] S-R2D-E: ? integration with UNITN components for conversion and translation

Edited 1 week ago by Alessio

Figure 3 - "Requirement" labelled issue sample

- **The "Task" label:** A task refers to a work item that a specific developer should complete within a single sprint. Several tasks together implement a requirement. The tasks are the granular units that are developed in sprints. A task is associated with one requirement only. A sample "Task" labeled issue is presented in Figure 4:



InteropEHRate › issues › Issues › #59

Open Opened 1 week ago by Alessio

Close issue New issue

New version of R2D library returning test data provided by FTGM

Update R2D library in order to create mock data provided by FTGM

This task participate to realize Requirement #19 - R2D import of (portion of) Patient Summary from national EHR on S-EHR

Edited 5 days ago by Alessio

Figure 4 - "Task" labelled issue sample

- **Macro-components labels:** This type of label is used to associate a requirement and its corresponding tasks to the analogous macro-component. Each macro-component realizes one or more user requirements. Each user requirement is offered by a single macro-component. A macro-component may be composed of different nested components not directly associated to user requirements. Macro-components and nested components are described in deliverable [D2.4]. The table below depicts the list of macro-component labels:

Macro-component label	Description
HCP-A	Reference HCP App
IEHR-P	InteropEHRate Profile
IHR	InteropEHRate Research Services
HIS	InteropEHRate Health Services
IHT	InteropEHRate Health Tools
S-EHR-A	S-EHR Mobile App
S-EHR-B	S-EHR Message Broker
S-EHR-C	S-EHR Cloud

Table 1 - Macro-component labels

- **Components labels:** This type of label is used to associate an issue (Requirement or Issue) with the corresponding nested component. Table 2 presents the components labels and the label of their parent macro-component.

Component label	Description	Macro-component label
M-R2D-SM	Mobile R2D Security Management	**S-EHR-A**
M-D2D-SM	Mobile D2D Security Management	
MES	Mobile Encrypted Storage	
MEC	Mobile Encrypted Communication	
M-R2D-E	Mobile R2D HR Exchange	

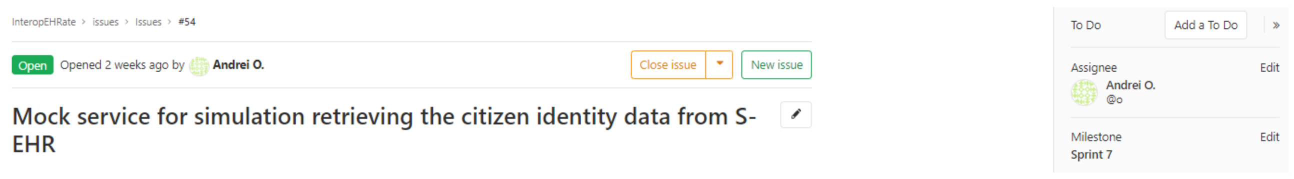
M-D2D-E	Mobile D2D HR Exchange	
M-RS	Mobile Research Data Sharing	
M-AA	Mobile Anonymization and Aggregation	
SEC	Server Encrypted Communication	**IRS**
S-RS	Server Research Data Sharing	
T-KM	Knowledge Management Tool	**IHT**
T-DM	Data Mapping Tool	
S-R2D-SM	Server R2D Security Management	**IHS**
SEC	Server Encrypted Communication	
S-R2D-E	Server R2D HR Exchange	
HRDI	HR Data Index	
S-IP	S-EHR Integration Platform	
S-LP	S-EHR Localisation Platform	
S-LDI	Legacy data import	
S-DM	FHIR Data Mapping	
S-IE	Information Extraction and Data Formalisation	
S-EX	FHIR Export	
S-IM	FHIR Import	
S-LT	Label Translation	
HCP	HCP App	**HCP-A**
T-R2D-SM	Terminal R2D Security Management	
T-D2D-SM	Terminal D2D Security Management	
TEC	Terminal Encrypted Communication	
T-R2D-E	Terminal R2D HR Exchange	
T-D2D-E	Terminal D2D HR Exchange	
S-EHR-B	S-EHR Message Broker	**S-EHR-B**

S-EHR-C	S-EHR Cloud	**S-EHR-C**

Table 2 - Components labels

Note that some macro-components, including the S-EHR Cloud and the S-EHR Message Broker do not have any nested components constituting them. In such cases, only macro-component labels exist.

Finally, each task is assigned to a person (i.e. a developer responsible to undertake it) and tagged with the sprint where it needs to be implemented. If a task is undertaken by more than one person, the assignee is the main contributor, that is considered accountable for the completion, where the others are placed as participants. In the figure below, a sample task with the person in charge is depicted, along with the sprint where this task should be finished.


InteropEHRate > issues > Issues > #54

Open Opened 2 weeks ago by Andrei O.

Close issue | New issue

Mock service for simulation retrieving the citizen identity data from S-EHR

To Do | Add a To Do

Assignee: Andrei O. @o — Edit

Milestone: Sprint 7 — Edit

Figure 5 - A task assigned to a person

3. CONCLUSIONS AND NEXT STEPS

This document presented the development and integration roadmap of the InteropEHRate project, as it is agreed by the technical partners of the Consortium. It described the tools and techniques used / will be used throughout the project in order to deliver the integrated InteropEHRate platform.

Additionally, the work that has been done towards that direction in the first year of the InteropEHRate project is also presented in the Annex to this deliverable. Though this will be the only deliverable depicting the work done in the Development and Integration task, its outcomes will be provided through the development-related deliverables of the project.

REFERENCES

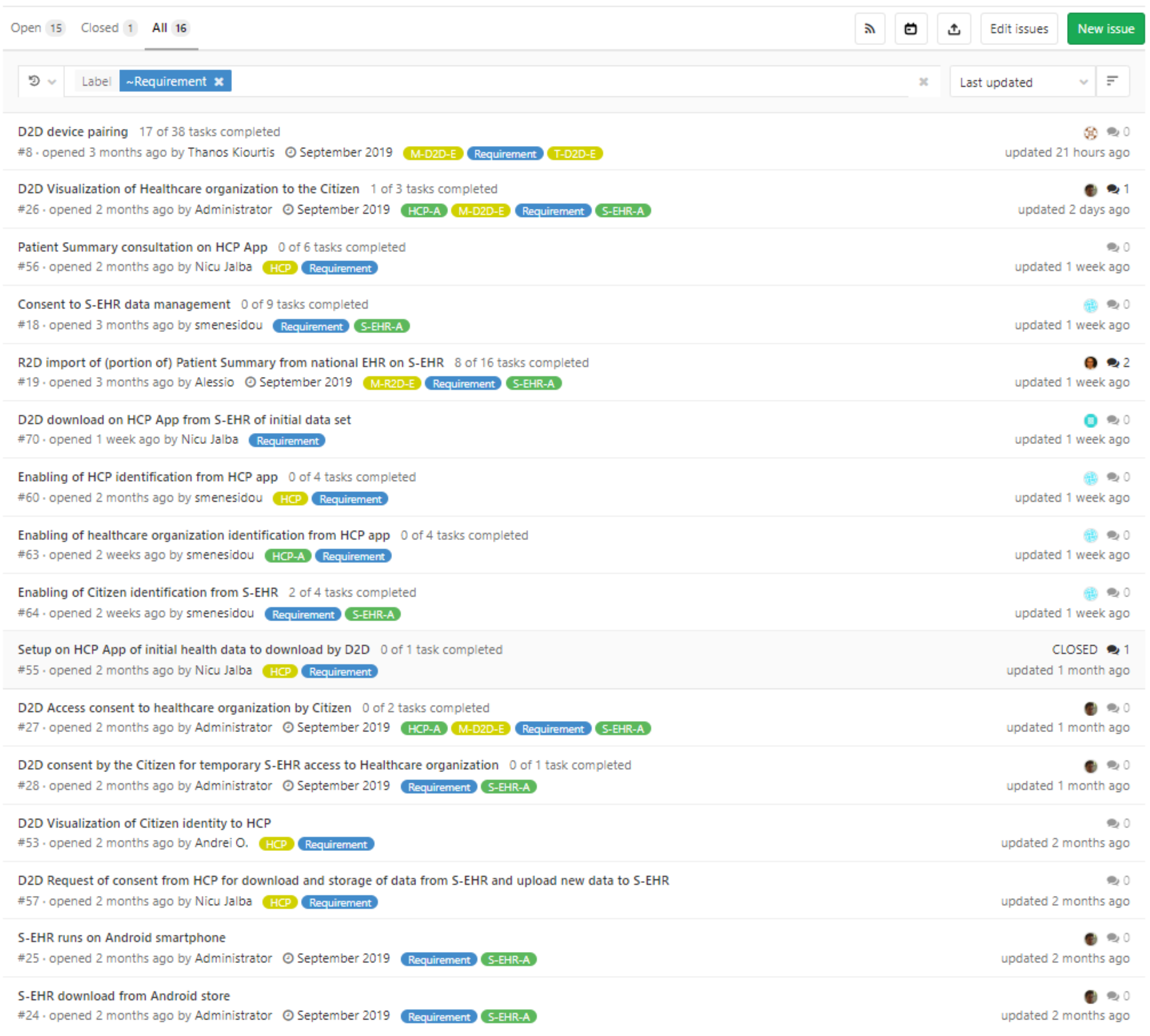
- **[Apache Maven]** Apache Maven. Website: https://maven.apache.org/
- **[CoffeeScript]** CoffeeScript. Website: https://coffeescript.org/
- **[Dagger]** Dagger. Website: https://dagger.dev
- **[D2.1]** InteropEHRate Consortium, User Requirements for cross-border HR integration - V1, 2019. www.interopehrate.eu/resources
- **[D2.4]** InteropEHRate Consortium, InteropEHRate Architecture - V1, 2019. www.interopehrate.eu/resources
- **[Espresso]** Espresso Android UI tests. Website: https://developer.android.com/training/testing/espresso
- **[Gradle]** Gradle Build Tool. Website: https://gradle.org/
- **[GitLab Runner]** Configuring GitLab Runners. Website: https://docs.gitlab.com/ee/ci/runners/
- **[HAPI FHIR]** HAPI FHIR. Website: https://hapifhir.io/
- **[IntelliJ IDEA]** IntelliJ IDEA. Website: https://www.jetbrains.com/idea/
- **[Issue tracker]** InteropEHRate issues. Website: http://iehrgitlab.ds.unipi.gr/interopehrate/issues/issues
- **[Jenkins]** Jenkins. Website: https://jenkins.io/
- **[JUnit]** JUnit 5. Website: https://junit.org/junit5/
- **[Kotlin-Android]** Kotlin and Android. Website: https://developer.android.com/kotlin/
- **[MongoDB]** MongoDB. Website: https://www.mongodb.com/
- **[PostgreSQL]** PostgreSQL. Website: https://www.postgresql.org/
- **[Realm]** Realm: Create reactive mobile apps in a fraction of the time. Website: https://realm.io/
- **[Scrum]** Scrum. Website: https://www.scrum.org
- **[Selenium]** Selenium - Web Browser Automation. Website: https://www.seleniumhq.org/

ANNEX: EXAMPLE OF PORTION OF YEAR 1 ROADMAP

This Annex depicts a portion of the integration roadmap that took place during the first year of the InteropEHRate project. It is used to showcase how the technical partners collaborated and how the work proceeded in order to have a complete integrated InteropEHRate platform.

As already declared, the roadmap in the InteropEHRate project is in fact a multilevel concept. The procedure starts with the identification of the user requirements as these are extracted by the Use Cases. In our case, the user requirements were derived from the first scenario as it is described in the corresponding deliverable [D2.1]. These roll down to a more granular concept; the *features* that basically regard the functionalities a software provides to its end user. These features are finally translated into the tasks that need to be developed by the technical partners. An important note at this point is that the features are not the same thing as the tasks. In fact, a feature may need more than one tasks to be implemented in order to be considered as completed.

In order to minimize the confusion with multi-labeled issues, it is agreed that only the *requirements* and the *tasks* are labeled in the GitLab issue tracker. The list of the requirements to be implemented during the first year of the project can be seen on the issue tracker easily by filtering the search engine to show only the issues that are labeled as ‘Requirement’. These are presented in Figure 6 below:


Open 15 Closed 1 All 16

Edit issues New issue

Label ~Requirement

Last updated

D2D device pairing 17 of 38 tasks completed 0
#8 · opened 3 months ago by Thanos Kiourtis September 2019 M-D2D-E Requirement T-D2D-E
updated 21 hours ago

D2D Visualization of Healthcare organization to the Citizen 1 of 3 tasks completed 1
#26 · opened 2 months ago by Administrator September 2019 HCP-A M-D2D-E Requirement S-EHR-A
updated 2 days ago

Patient Summary consultation on HCP App 0 of 6 tasks completed 0
#56 · opened 2 months ago by Nicu Jalba HCP Requirement
updated 1 week ago

Consent to S-EHR data management 0 of 9 tasks completed 0
#18 · opened 3 months ago by smenesidou Requirement S-EHR-A
updated 1 week ago

R2D import of (portion of) Patient Summary from national EHR on S-EHR 8 of 16 tasks completed 2
#19 · opened 3 months ago by Alessio September 2019 M-R2D-E Requirement S-EHR-A
updated 1 week ago

D2D download on HCP App from S-EHR of initial data set 0
#70 · opened 1 week ago by Nicu Jalba Requirement
updated 1 week ago

Enabling of HCP identification from HCP app 0 of 4 tasks completed 0
#60 · opened 2 months ago by smenesidou HCP Requirement
updated 1 week ago

Enabling of healthcare organization identification from HCP app 0 of 4 tasks completed 0
#63 · opened 2 weeks ago by smenesidou HCP-A Requirement
updated 1 week ago

Enabling of Citizen identification from S-EHR 2 of 4 tasks completed 0
#64 · opened 2 weeks ago by smenesidou Requirement S-EHR-A
updated 1 week ago

Setup on HCP App of initial health data to download by D2D 0 of 1 task completed CLOSED 1
#55 · opened 2 months ago by Nicu Jalba HCP Requirement
updated 1 month ago

D2D Access consent to healthcare organization by Citizen 0 of 2 tasks completed 0
#27 · opened 2 months ago by Administrator September 2019 HCP-A M-D2D-E Requirement S-EHR-A
updated 1 month ago

D2D consent by the Citizen for temporary S-EHR access to Healthcare organization 0 of 1 task completed 0
#28 · opened 2 months ago by Administrator September 2019 Requirement S-EHR-A
updated 1 month ago

D2D Visualization of Citizen identity to HCP 0
#53 · opened 2 months ago by Andrei O. HCP Requirement
updated 2 months ago

D2D Request of consent from HCP for download and storage of data from S-EHR and upload new data to S-EHR 0
#57 · opened 2 months ago by Nicu Jalba HCP Requirement
updated 2 months ago

S-EHR runs on Android smartphone 0
#25 · opened 2 months ago by Administrator September 2019 Requirement S-EHR-A
updated 2 months ago

S-EHR download from Android store 0
#24 · opened 2 months ago by Administrator September 2019 Requirement S-EHR-A
updated 2 months ago

Figure 6 - Requirements of the first year

As depicted in Figure 6, each requirement has an assignee which regard the requirement leader. Also, in case a requirement is complete, it is highlighted.

In order to demonstrate the integration roadmap of InteropEHRate, a user requirement is chosen to be presented. Thus, the '**D2D device pairing**' requirement is selected. What is going to be presented in this annex is:

- The way a user requirement is presented in the GitLab issue tracker
- How it is split into several tasks
- How these tasks are managed and placed into Sprints, and
- What are the key roles of the partners in the development of the requirement

As we mentioned above, the requirement that will showcase the integration roadmap of InteropEHRate for its first year is called '**D2D device pairing**'. Upon the completion of this requirement, the two core components of InteropEHRate for the first scenario (i.e. the S-EHR mobile app and the HCP app) should be paired.

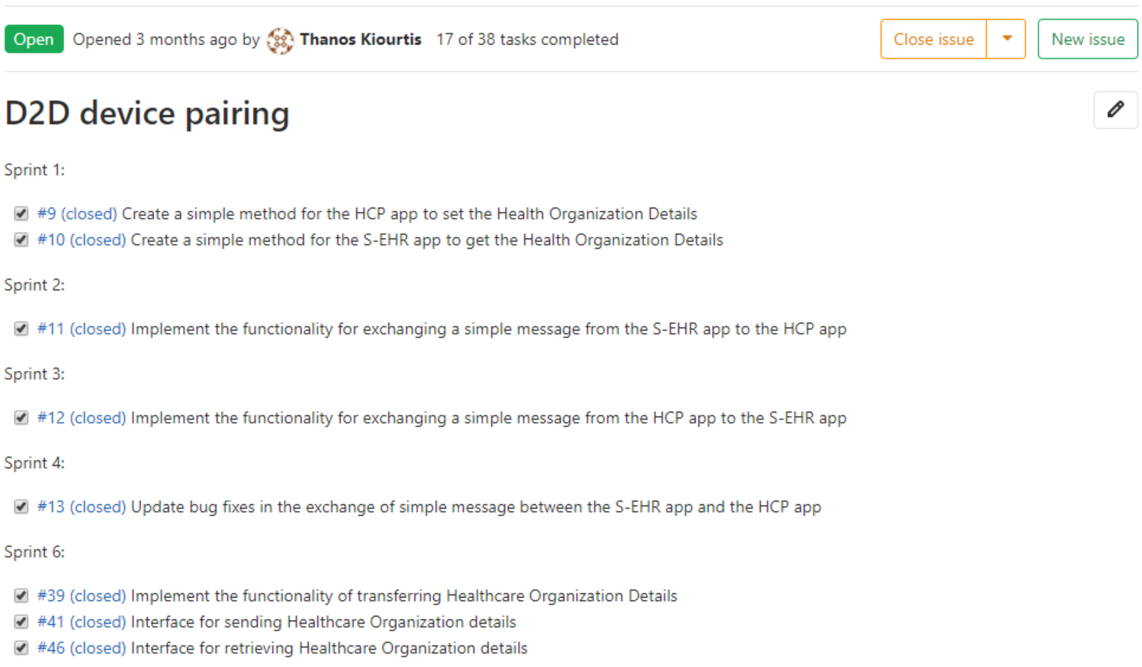
The identified tasks needed for the completion of this requirement are the ones presented in the table below:

D2D device pairing	
Sprint	**Task name**
Sprint 1	Create a simple method for the HCP app to set the Health Organization Details
	Create a simple method for the S-EHR app to get the Health Organization Details
Sprint 2	Implement the functionality for exchanging a simple message from the S-EHR app to HCP app
Sprint 3	Implement the functionality for exchanging a simple message from the HCP app to th EHR app
Sprint 4	Update bug fixes in the exchange of simple message between the S-EHR app and the app
Sprint 6	Implement the functionality of transferring Healthcare Organization Details
	Interface for sending Healthcare Organization details
	Interface for retrieving Healthcare Organization details
Sprint 7	QR code for Bluetooth connection
	Implement the functionality of transferring Personal Data
	Interface for sending Personal Data
	Interface for retrieving Personal Data
Sprint 8	Integration demo test
Sprint 9	Implement the functionality of transferring portion of patient summary
	Interface for retrieving portion of Patient Summary
	Interface for sending portion of Patient Summary
Sprint 11	Define listener for S-EHR app upon receiving organization data
Sprint 12	Implement the listener for S-EHR App upon receiving organization data

	Define listener for HCP App upon receiving demographic data
	Implement the listener for HCP App upon receiving demographic data
Sprint 13	Define listener for S-EHR App upon receiving Evaluation Data
	Define listener for HCP App upon receiving patient summary
Unknown sprint for the time being	Provide HCP identity certificate to the S-EHR app
	Update the QR code including the digital signed organization info
	Provide method for signing organization info - HCP App
	Expose API for signing organization info - HCP App
	Place the signed organization info in the QR code
	Provide method for verifying organization info - S-EHR App
	Expose API for verifying organization info - S-EHR App
	Integrate the API regarding the verified organization info with the S-EHR App
	Define listener for S-EHR App upon requesting patient summary
	Define listener for S-EHR App upon receiving consent and requesting consent answer
	Define listener for HCP App upon requesting evaluation data
	Fix closeConnection method from the S-EHR App side
	Define listener for HCP App and S-EHR App upon closing the connection
	Define interface for HCP App that will have all the required listeners
	Define interface for S-EHR App that will have all the required listeners

Table 3- 'D2D device pairing' requirement's associated tasks

In the figure below you should see how these tasks are shown in the description of the requirement:


Open Opened 3 months ago by Thanos Kiourtis 17 of 38 tasks completed Close issue New issue

D2D device pairing

Sprint 1:

- [x] #9 (closed) Create a simple method for the HCP app to set the Health Organization Details
- [x] #10 (closed) Create a simple method for the S-EHR app to get the Health Organization Details

Sprint 2:

- [x] #11 (closed) Implement the functionality for exchanging a simple message from the S-EHR app to the HCP app

Sprint 3:

- [x] #12 (closed) Implement the functionality for exchanging a simple message from the HCP app to the S-EHR app

Sprint 4:

- [x] #13 (closed) Update bug fixes in the exchange of simple message between the S-EHR app and the HCP app

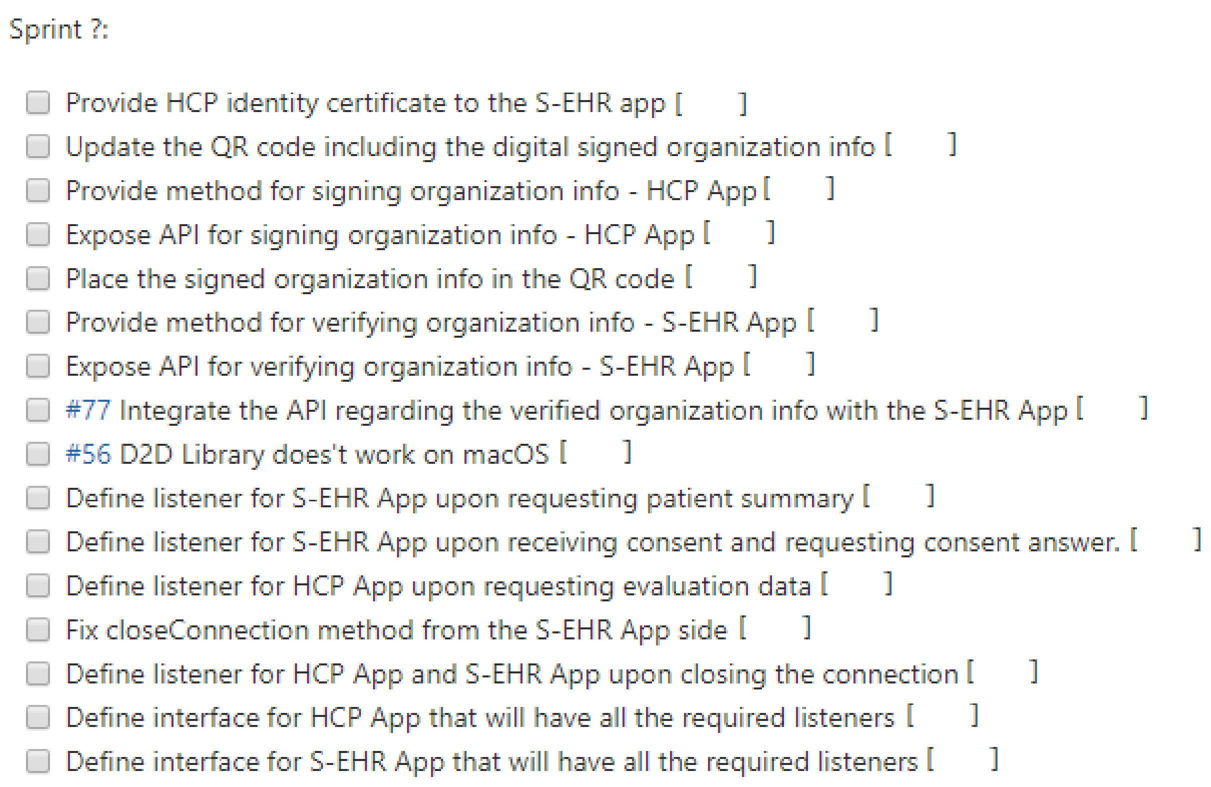
Sprint 6:

- [x] #39 (closed) Implement the functionality of transferring Healthcare Organization Details
- [x] #41 (closed) Interface for sending Healthcare Organization details
- [x] #46 (closed) Interface for retrieving Healthcare Organization details

Figure 7 - 'D2D device pairing' requirement in GitLab

As you can see, all the tasks are linked to the requirement with their ID. When a task is complete the requirement leader should make sure to check the checkbox of the corresponding task in order to close it.

As already mentioned, all the tasks needed for the completion of each requirement should be identified prior to the start of the development. But for some tasks the sprint may not be easy to address. Such tasks are also presented in the description of the requirement with a minor difference; They are placed under a '?' sprint for the moment and the *requirement leader* indicates the *task leader* by writing their name in the end of the task next to the task name, until the exact sprint of implementation is identified. This is depicted in Figure 8.



Sprint ?:

- ☐ Provide HCP identity certificate to the S-EHR app [ ]
- ☐ Update the QR code including the digital signed organization info [ ]
- ☐ Provide method for signing organization info - HCP App [ ]
- ☐ Expose API for signing organization info - HCP App [ ]
- ☐ Place the signed organization info in the QR code [ ]
- ☐ Provide method for verifying organization info - S-EHR App [ ]
- ☐ Expose API for verifying organization info - S-EHR App [ ]
- ☐ #77 Integrate the API regarding the verified organization info with the S-EHR App [ ]
- ☐ #56 D2D Library does't work on macOS [ ]
- ☐ Define listener for S-EHR App upon requesting patient summary [ ]
- ☐ Define listener for S-EHR App upon receiving consent and requesting consent answer. [ ]
- ☐ Define listener for HCP App upon requesting evaluation data [ ]
- ☐ Fix closeConnection method from the S-EHR App side [ ]
- ☐ Define listener for HCP App and S-EHR App upon closing the connection [ ]
- ☐ Define interface for HCP App that will have all the required listeners [ ]
- ☐ Define interface for S-EHR App that will have all the required listeners [ ]

Figure 8 - Tasks of the 'D2D device pairing' requirement not yet placed in sprints

Moving on to the tasks, each task is also assigned to a task leader, similar to the way a requirement is assigned to a requirement leader.