SMARTBEAR

CALL H2020-SC1-FA-DTS-2018-2020

Trusted digital solutions and Cybersecurity in Health and Care

TOPIC DT-TDS-01-2019

Smart and healthy living at home

SMART BEAR

"Smart Big Data Platform to Offer Evidence-based Personalised Support for Healthy and Independent Living at Home"

D6.6 (D31) – Report on integrated SMART BEAR platform v3

Due date of deliverable: 31/12/2023
Actual submission date: 21/12/2023

Grant agreement number: 857172

Start date of project: 01/09/2019

Lead contractor: CNR

Duration: 60 months

Project funded by the European Commission within the EU Framework Programme for Research and Innovation HORIZON 2020	
Dissemination Level	
PU = Public, fully open, e.g., web	X
CO = Confidential, restricted under conditions set out in Model Grant Agreement	
CI = Classified, information as referred to in Commission Decision 2001/844/EC.	
Int = Internal Working Document	

D6.6 (D31) – Report on integrated SMART BEAR platform v3

Editors
Manuel M Perez Perez (ATOS)

Contributors
Alberto Acebes (ATOS)

Reviewers
Alessandra Gargiulo (FCSR)
Alice Leporini (FCSR)

The work described in this document has been conducted within the project SMART BEAR, started in September 2019. This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 857172

Executive Summary

This deliverable details the functionalities offered, the components developed, and the tools used for building the Smart Bear Platform in its last version, V3. This deliverable complements D6.5 in providing a complete description of the SMART BEAR platform ready to be used for large-scale pilots.

Contents

List of acronyms

API	Application Programming Interface
BDA	Big Data Analytics
CDR	Clinical Data Repository
CI/CD	Continuous Implementation / Continuous Deployment
FHIR	Fast Health Interoperable Resources
HAPI	HL7 Application Programming Interface
K8s	Kubernetes
LEP	Large Scale Pilots
PoP	Pilot of Pilots
SFC	SMART BEAR Feedback Corner
SPA	Security and Privacy Assurance

List of figures

1 Introduction

This document forms part of the set of deliverables describing the different versions of the SMARTBEAR platform integration, with the following sequence:

- D6.7 Integrated SMART BEAR platform v0.1. *(already delivered on month 19)*
- D6.8 Report on integrated SMART BEAR platform v0.1. *(already delivered on month 19)*
- D6.1 Integrated SMART BEAR platform v1. *(already delivered on month 30)*
- D6.2 Report on integrated SMART BEAR platform v1. *(already delivered on month 30)*
- D6.3 Integrated SMART BEAR platform v2. *(already delivered on month 42)*
- D6.4 Report on integrated SMART BEAR platform v2.*(already delivered on month 42)*
- *D6.5 Integrated SMART BEAR platform v3. (in parallel, to be delivered on month 52)*
- ***D6.6 Report on integrated SMART BEAR platform v3. (this document, to be delivered on month 52)***

1.1 Overview

During this last period, the full integration of the platform based on a Continuous Integration/Continuous Deployment (CI/CD) methodology has been accomplished, as well as the migration of the data collected during the Pilot of Pilots (PoP). CI/CD practices are used in conjunction with version control systems, automated testing tools, and containerization technologies. Adopting CI/CD can significantly improve the efficiency, quality, and reliability of software development and deployment processes during the demanding period of large-scale pilots. The final version of the platform includes new components that were not implemented for the PoP; as these new functionalities were implemented, the complexity of the integration progressively increased. In addition, migration of the data collected during the PoP required the transfer of data from four different databases, namely the FHIR and non-FHIR databases, both based on *MariaDB*[1], the user management database and the *MongoDB*[2], used for storing real-world data collected by the sensors. In addition, we set up a plan for transferring the CI/CD infrastructure and knowledge from Atos to ICCS, as WP6 will be finalized by the end of 2023 after being extended 16 months from the original planning.

1.2 Goals of the deliverable

This deliverable aims at reporting the activities performed during the last period regarding the integration and testing activities carried out in WP6 of SMART BEAR project.

1.3 Activities follow-up

During this last period and in addition to the daily basis communication by using *Basecamp* tools, emails and ad hoc teleconferences, two different recurrent meetings were held on a weekly basis.

[1] MariaDB Server is one of the most popular open-source relational databases (*https://mariadb.org*)
[2] MongoDB, an open-source NoSQL database management program (*https://www.mongodb.com*)

Therefore, in addition to the weekly Friday follow-up meetings already set up from the beginning of the WP6 activities, we set up an additional technical meeting every Wednesday. The latter aimed at discussing but overall solving technical issues. This new type of meeting showed great effectiveness as both participants and chaired person (Atos) were exclusively focused on solving technical problems following a scrum-like methodology. Technical discussions and issues were reported and followed up using *Basecamp* services.

2 Activities performed during the last period

During this last period the PoP has continued, so the technical partners had to split their work in maintaining their components and resolving issues regarding the first prototype while advancing in the development and refinement of their components, and their adaptation to the CI/CD framework.

2.1 Maintenance of the first prototype for the PoP

Alongside the activities performed for the integration of the different components into the new K8s framework, partners also worked on maintaining the initial version of the platform to keep it operative during the PoP. In deliverable D6.4 we explained the methodology used for dealing with issues, bugs, refinements, and even new functionalities that arose from the PoP. The interaction between clinicians and technical partners were initially based on the already established channels; as the number of issues increased, we chose to work using *Trello*[3] and *Hipporello*[4] as collaborative frameworks. However, other collaborative tools were discussed for the large-scale pilots, where the number of users will increase even more. Finally, we decided to use *Basecamp* services. The main advantage is that *Basecamp* was already used by all partners from the beginning of the project, and it is useful to have a single access point for dealing with all the info, documents, and communications of the project without the need to rely on new tools that require a learning process, and, in any case, they would not be fully integrated with *Basecamp*.

2.2 Gitlab

Gitlab[5] base is *Git*[6], which is version control software. It is a widely used de facto standard for version control, providing a suite of tools for software development and collaboration and supporting *DevOps*[7]. In fact, *Gitlab* covers the entire software development lifecycle from planning to monitoring (*Figure 1*). Within the scope of SMART BEAR, we are using *Gitlab* for providing a common repository to host the code of the different components of the project and to facilitate CI/CD. Partners were assigned with privileges for managing their activities related to the project, allowing them to manage and track changes of their source code, to collaborate with other partners, and to maintain different branches for development. *Gitlab* also includes an issue-tracking system that allows technical partners to manage and prioritize tasks, track bugs, and discuss issues within the context of their codebase. As part of the CI/CD integration approach, the pipelines enable automated testing and deployment of code changes, ensuring code is tested and validated before being merged into the main codebase. Documentation is managed by using the wiki feature.

[3] Trello brings all your tasks, teammates, and tools together (*https://www.trello.com*)
[4] Hipporello, streamline business processes (*https://www.hipporello.com*)
[5] GitLab, the most comprehensive AI-powered DevSecOps Platform (*https://about.gitlab.com)*
[6] Git is a free and open-source distributed version control system (*https://git-scm.com*)
[7] DevOps is a combination of software development (dev) and operations (ops)

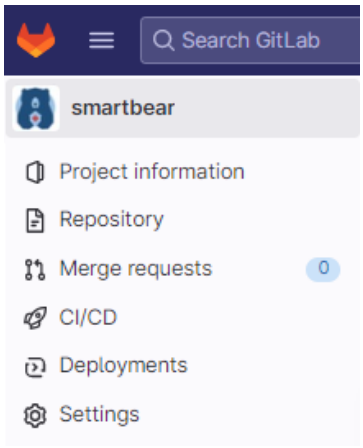

Figure 1: Main functionalities of the -SMART BEAR Gitlab.

2.3 Set up the collaborative environment

As the complexity of the integration increase and in order to monitoring the different activities, we started taking advance of web-based project management and collaboration tools. First, we created the *SMART BEAR Feedback Corner* (SFC) based on *Trello* and *Hipporello*. The SFC provides a visual way for partners to organize and prioritize tasks, issues and bugs that emerged during the PoP. SFC is organized based on boards, lists, and cards. There is one board for the project (https://trello.com/b/MZFagoOz/feedback-corner), and within this board, lists to represent different type of activities related to the project. Cards are opened within the lists for describing specific tasks, actions, or items to be addressed. However, for the specific daily technical integration activities we rely in *Basecamp*[8] it is a general project management tool that is has been used in Smart Bear from the very beginning for project management and communication, as well as a document repository. As part of the functionalities, the "To-Do Lists" are quite useful for defining action, tracking task status, setting due dates, and assigning responsibilities. Basecamp also includes message boards for project-specific discussions, enabling partners to communicate and share updates. Therefore, we extensively used Basecamp for daily management of WP6 activities to collaborate and stay organized. A screenshot summarizing the WP6 activities is shown in *Figure 2*. All the messages, to-do lists, schedule of activities as well the repo of working files were all structured and accessible from a single point, serving also as a central hub for communication.

As the number of cards increased, we decided to divide them into two specific main tabs for easing the tasks follow-up; namely the "*SMART BEAR Platform I&T*" and the "*Testing Staging Production Envs*". Once an issue is solved or a task finalized, they are archived in the "Done" tab (See *Figure 2* below).

[8] BaseCamp, project management software, online collaboration (*https://www.basecamp.com*)

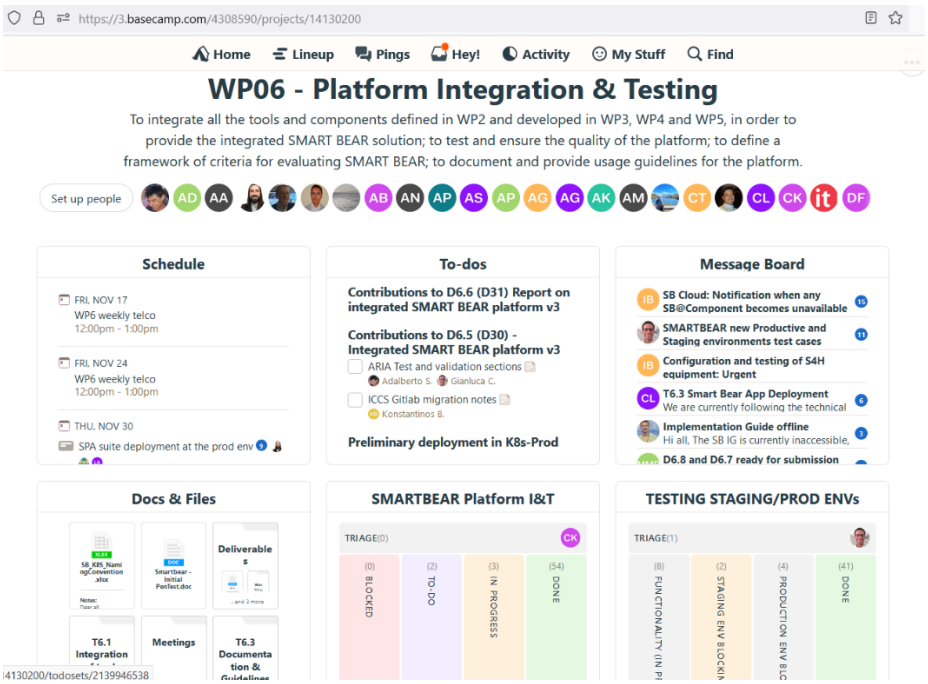

Figure 2: Screenshot of the page of Basecamp for managing WP6 activities.

The platform facilitates the complexity of the integration. *Figure 3* below shows a general view of all the tasks created during February 2023. For each of these activities, specific persons were assigned, and a thread of messages was created.

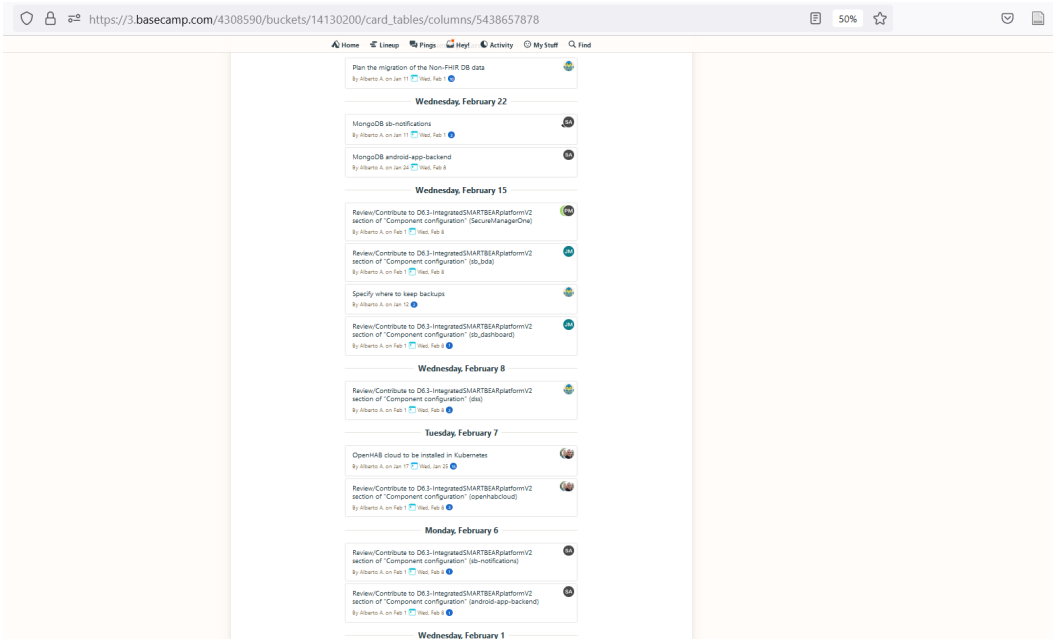

Figure 3: Screenshot of Basecamp showing a number of cards created during February 2023.

Figure 4 below shows one of the cards created for defining a pending task. In addition to the description of the task, the author can assign which partners need to be involved and set up a number of actions/steps, which are marked with a checkmark once finalized.

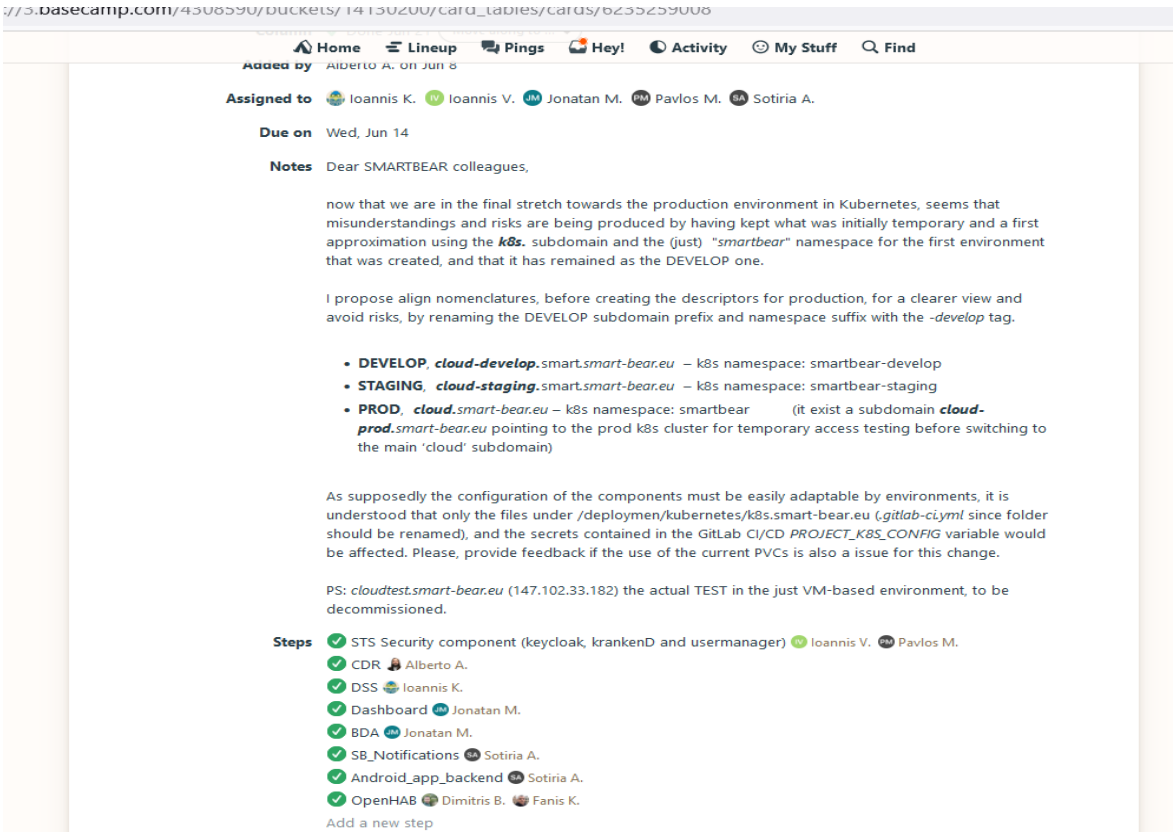

Figure 4: Example of a specific card, in which partners involved are assigned, as well a deadline a specific action that must be marked by the responsible partner once the action is completed.

As the proven usefulness of *Basecamp* and the familiarity of all consortium partners with using it, we decided to use it as the main communication and reporting issues tool for large scale pilots as well.

2.4 Setting up a new environment: the staging environment

As reported in previous deliverables, the project initially operated within two primary environments: the testing environment, focused on application development and testing, and the production environment, where applications deemed functional were manually promoted. These two environments were good enough for managing the ICT activities for the PoP. However, the platform's growing complexity, resulting from the integration of new functionalities and test cases, raised the need for an additional environment. Consequently, a staging environment was established to facilitate the testing of functionalities in an environment closely resembling the production setting.

2.5 Changes in subdomains

The expansion of execution environments on the platform (develop, staging, and prod) necessitated a refinement of the initial generic preliminary naming convention for *Kubernetes*. This adjustment aimed at providing more specificity and descriptive clarity for effective differentiation. Consequently, modifications to the certificate configuration for the *Kubernetes* API were necessary to ensure accessibility via a domain name rather than relying solely on the public IP. The *Gitlab*

configuration, which originally used the k8s. subdomain as the endpoint for reaching the *Kubernetes* API (specified in the CI/CD variable K8S_SERVER), had to be updated to align with the new nomenclature.

2.6 Docker registry credentials

Although *Gitlab* has its own registry, the deployment of most components relies on the project's *Docker*[9] private registry (*Nexus*[10]), serving as the repository for storing versions of *Docker* images. Presently, access to this registry is facilitated through a generic user, a configuration that has been updated due to policy adjustments within the hosting company.

The credentials for this access are securely stored in a secret file in *Kubernetes*[11], specific to each environment's namespace. Although altering these credentials may be considered a straightforward task, it is deemed valuable to document this process for future reference. It is important to note that the execution of this procedure should be directed towards the target namespace, and a functional *Docker* service should be available during the operation.

2.7 Integration of components previously supported by third-party services

The final version of the platform, transitioned to *Kubernetes* environments, now incorporates components and services that were previously placed in third-party infrastructure and employed as external services in earlier versions. To centralize resources, enhancing the platform's overall management and maintenance, those components were migrated to the SMART BEAR cloud. This migration involved consolidating external components into the project cluster, aligning them with the remaining elements. Furthermore, tasks such as performance monitoring, security measures, and backups were centralized through this process. Notably, external components have been seamlessly integrated into the CI/CD processes, streamlining control, and facilitating automated deployments. Despite the SB Notifications component being part of the previous implementation, its database was originally hosted on public cloud.mongodb.com. The migration process involved, among other tasks, defining necessary configurations, deployment descriptor files, configuring a new database, transferring data, and adjusting the component for execution in the final environment. A detailed description of the data migration process and its validation is provided later in this document. Similarly, the Android App Backend, initially hosted on Heroku (a cloud application platform), had its database on cloud.mongodb.com. The component has now been relocated to instances within the project's *Kubernetes* clusters. Again, the subsequent sections of this document outline the data migration process and its validation for this particular migration.

[9] Docker, accelerated Container Application Development (*https://www.docker.com*)
[10] Sonatype Nexus Repository (*https://www.sonatype.com/products/sonatype-nexus-repository*)
[11] Kubernetes, is an open-source system for automating deployment, scaling, and management of containerized applications (*https://kubernetes.io*)

2.8 Migration of the databases used during the PoP

The data migration process conducted after the Pilot of Pilots (PoP) uncovered additional specific requirements arising from managing large volume of data. Notably, the conventional utilization of resource counters supplied by the Clinical Data Repository (CDR) proved to be less efficient than expected when employing the methods specified in the existing *HAPI-FHIR*[12] standard implementation. Since this functionality is crucial for generating summaries on the main screen of the dashboard, delays in data retrieval led to timeout issues. Consequently, a strategic decision was made to address this challenge by introducing a custom extended operation. This custom operation was designed to directly interact with the database, employing a native count operation to optimize and expedite the data retrieval process.

In addition, we discussed a plan for dealing with the migration of the *FHIR* database, the non-*FHIR* database, the *MongoDB* for physiological signals and the security databases. As data migration involves moving data from one system to another and among different versions, it is therefore a critical process that requires careful planning and execution to secure data integrity and minimize downtime. Consequently, we carefully set up a roadmap for ensuring the successful migration of the different databases.

The first step was to assess and identify the specific database requirements to understand the existing database structure, data types, and interdependences as well as the target database system and potential changes between the old and new versions. A critical point was to evaluate the volume of data and estimate downtime requirements. Before executing the migration, we performed a comprehensive backup of the existing database to avoid data loss and recovery in case of problems during migration. Once the back-ups were made for each database, we ensured their reliability, tested data integrity after the back-ups, defined the best recovery strategy and finally we developed a rollback plan in case the migration encountered issues.

We realized the importance of defining a schema for data migration by creating the specific scripts needed or using the specific tools provided for migrating the data schema (tables, indexes, constraints, etc.) to the new environment. Therefore, depending on the type of database, we developed scripts or use migration tools to transfer the data from the old databases to the new ones.

After migration, we performed tests to validate the migration process and verify the data consistency and integrity between the source and target databases.

It is important to highlight the need to modify database connection strings and configurations in the application code to point to the new database and then to perform testing of the application with the new database to identify and address any compatibility issues.

As the migration requires to shut down the platform and stop the pilot, we scheduled it in agreement with the clinical partners during a short period to minimize the impact on users.

The technical documentation of the migration process is reported in D6.5, including steps taken, issues encountered, and resolutions applied.

[12] HAPI FHIR is an open-source Java-based implementation of the FHIR specification developed by the HL7

2.9 Automatic and manual testing

In order to set up a standardized procedure for testing components before their final deployment in production, a number of functional test cases were set up. As part of the testing activities, it was very important to ensure that the data migrated from the initial version based on virtual machine and the new platform was performed correctly, therefore a migration validation check list were defined and used for checking data consistency.

3 Concluding remarks

The work done in WP6 can be broadly divided in two main activities, one dealing with the initial prototype maintenance, including new functionalities and some needed refinements, and the other dealing with the setup of the *Kubernetes* framework, support for partners with their components integration and components and databases migration together with the final testing of the platform. As a result, we were able to ensure the proper functionalities of the platform during the whole duration of the PoP, as well as to set up CI/CD framework based on *K8s*, to integrate all the components of the final version of the platform, to migrate all the data from the original version to the final version, and finally to provide a battery of test for ensuring safe promotion of any update to the production environment. A number of tools were used for easing the integration, to set up the CI/CD environment, for managing daily technical activities, for establishing smooth communication channels between clinicians and technicians, and for dealing with reported issues, bugs and refinements. At the time of writing this report, the prototype is ready for starting the large-scale pilots, able to register patients, collect and store data from devices, and provide authorization and authentication mechanisms. Therefore, this prototype is able to implement most of the expected functionalities, although some sub-components of the Big Data Analytics *BDA* and *SPA* services are not yet fully migrated but will be by the end of the year. In any case, they have no impact at this early stage of the large-scale pilot.