Self Assessment Towards Optimization of Building Energy

Deliverable 2.2

Interfaces between platform, services and stakeholders

Deliverable Lead: FC.ID

Deliverable due date: 31/10/2021

Actual submission date: 02/12/2021

Version: 1.0

This project receives funding in the European Commission's Horizon 2020 Research Programme under Grant Agreement Number 957128.

Document Control Page	
Title	Interfaces between platform, services and stakeholders
Editor	FC.ID
Description	Describes the main interfaces that will be present in the SATO platform to enable its main stakeholders and software components to interact with each other.
Contributors	FC.ID, CORE, EKAG
Creation date	23/07/2021
Type	Report
Language	English
Audience	☒ public ☐ confidential
Review status	☐ Draft ☐ WP leader accepted ☒ Coordinator accepted
Action requested	☐ to be revised by Partners ☐ for approval by the WP leader ☐ for approval by the Project Coordinator ☐ for acknowledgement by Partners

Table of Contents

List of Figures

Revision history

Version	Author(s)	Changes	Date
V0.1	Vinicius Cogo (FC.ID)	Initial version	15/09/2021
V0.2	Vinicius Cogo (FC.ID)	First draft for partner review	16/11/2021
V0.3	Manthos Kampourakis (CORE), Thomas Fehr (EKAG), Michael Liniger (EKAG)	Reviewed version from partners	25/11/2021
V1.0	Vinicius Cogo, Pedro Ferreira (FC.ID)	Final check, formatting, and minor corrections before submission	30/11/2021

DISCLAIMER

The sole responsibility for the content of this publication lies with the SATO project and in no way reflects the views of the European Union.

EXECUTIVE SUMMARY / ABSTRACT / SCOPE

The SATO platform is a complete solution that encompasses communication, computing, storage, and energy management service components, supporting a self-assessment framework and several energy management self-optimization services. These services take advantage of building data and assessments to increase energy flexibility, efficiency, and user satisfaction.

Deliverable D2.2 presents the main interfaces that will be present in the SATO platform to enable its main stakeholders and software components to interact with each other. These interfaces are categorized into six types for different purposes, namely: Graphical User Interfaces (GUI), Command-Line Interfaces (CLI), Web Application Programming Interfaces (Web API), Publisher/Subscriber Interfaces, Filesystem Interfaces, and other (e.g., email and SMS). This deliverable is enclosed by a figure with the positioning of each interface type in the conceptual components of the SATO platform architecture.

Deliverable D2.2 is a direct output of Task 2.1 (*Development of the SRI enabled SATO Platform concept*) from WP2 (*Development of integrated technical Platform for SATO*). It provides guidance for the ongoing development in WP2 and will guide the specification and integration of concrete internal components associated with WP3 (*Development of SATO self-assessment framework*), WP4 (*Development of SATO self-assessment and self-optimization services toolbox*), and WP5 (*Development of SATO User Interaction and Interfaces*).

1. Introduction

1.1. Motivation

The SATO project proposes a platform focused on the self-assessment and optimization of energy- and comfort-related aspects of buildings. This platform will integrate all energy-related Equipment and Building Components (EBCs) and will provide means for the main actors of the related ecosystem to monitor and manage these components through the appropriate interfaces. Interfaces, in the context of this deliverable, are shared boundaries among different components of the SATO platform and the main foreseen stakeholders.

Figure 1 presents a simplified overview of the SATO platform architecture, showing its main blocks of components, while the internal components in each of these blocks can be seen in Figure 10, which were also described in detail in Deliverable D1.4 (*Description of the system architecture of the SATO platform*).

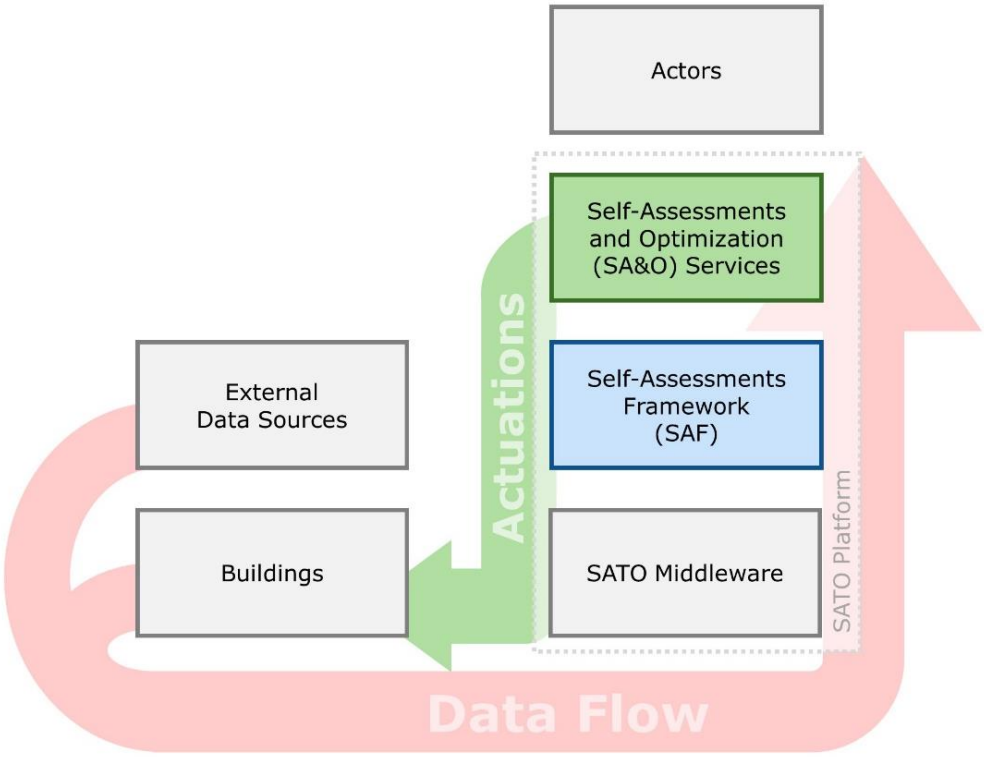

Figure 1: Overview of the main building blocks of the SATO platform.

The main stakeholders in the SATO project were identified in the Deliverable D1.1 (*Role of Actors and Design of Stakeholder Framework*), which are summarized as the following human actors:

- Building occupants
- Facility/building managers
- Grid operators
    - Aggregators
    - Energy providers
    - Distribution System Operators (DSO)
    - Transport System Operators (TSO)

Additionally, several SATO software/services were described in the same Deliverable D1.1 and will be used in this document, namely:

- SATO platform
- SATO Web App (Web, Android, iOS)
- Self-Assessment Framework (SAF)
- Self-Optimization Services (SOS)
- Appliances (APL)
- Building Management Systems (BMS)
- Facility Management Systems (FMS)
- Web-based interface
- BIM model

1.2. Objectives

The main objective of this Deliverable D2.2 is to outline and specify the different interfaces between the components of the SATO platform, services, and stakeholders. This deliverable is a direct output of Task T2.1 (*Development of the SRI enabled SATO platform concept*) from the WP2 (*Development of integrated technical platform for SATO*) and will serve as an input for the development of the SATO platform in Task D2.5 (*Development and integration of the SATO platform*) and the subsequent integration tasks in other work packages. The results from this deliverable conclude the work of Task T2.1 from the WP2.

1.3. Structure of the Document

This document is divided into four sections. The present Section 1 introduces the deliverable and provides an overview of the components and stakeholders we consider in the SATO Platform. Section 2 presents the background context of the many interfaces the SATO platform will consider. Section 3 places the different categories of interfaces in the appropriate positioning within the SATO platform. Finally, Section 4 summarises the results from this deliverable and concludes the document by providing guidance for the subsequent steps related with the development of the SATO platform.

2. Background

A computer interface is a generic concept that refers to any shared boundary that enables two or more components of a computer system to communicate with each other and interact to exchange information. More specifically, in the context of this deliverable, examples of components interacting through interfaces include the different internal software components of the SATO platform and the main foreseen stakeholders presented in the previous section.

In this section, some common categories of interfaces that have been considered in the development of the SATO platform are presented. It starts in Section 2.1 with user interfaces that enable human beings to interact with the system through graphical and command-line interfaces. In Section 2.2, the focus is pivoted to application-programming interfaces (APIs), which are interfaces that enable software components to interact with each other systematically. Finally, Section 2.3 describes other interfaces that may be integrated in the SATO platform to improve the communication with stakeholders for them taking faster informed actions.

2.1. User Interfaces

User interfaces refer to the specific category of interfaces that enable the interaction between humans and the computer. Generically, it may include many modalities of interactions, such as graphics, command lines, sound, position, movement, etc. In this deliverable, the focus will be solely on the first

two of them (graphical and command line interfaces) since they are the most prominent modalities for the foreseen interactions between the stakeholders and the SATO platform.

2.1.1. Graphical User Interfaces

Graphical user interfaces (GUI) are a type of user interface that enables users to interact with a computing system through graphical elements (e.g., icons, menus, pictures) rather than text-based ones. This type of interfaces can be adapted to work not only with computers but also with handheld mobile devices. In the SATO Project, the main graphical user interfaces are threefold: BIM desktop interfaces, Web-based graphical interfaces, and Mobile-based graphical interfaces.

SATO BIM Desktop Software

CYPE develops technical software that enables architecture, engineering, and construction professionals to design and analyse many structure types – e.g., from reinforced concrete and steel structures to soil retention elements, passing through foundations and project management. They have a team with more than 200 professionals, being more than one hundred of them developers. Their solutions promote the use of open formats for many different disciplines such as building information modelling (BIM), building energy modelling (BEM), energy efficiency, structures, acoustics, lighting, etc. Figure 2 presents an example of a 3D building model being visualized and edited in CYPE's original BIM (and related) desktop programs [1].

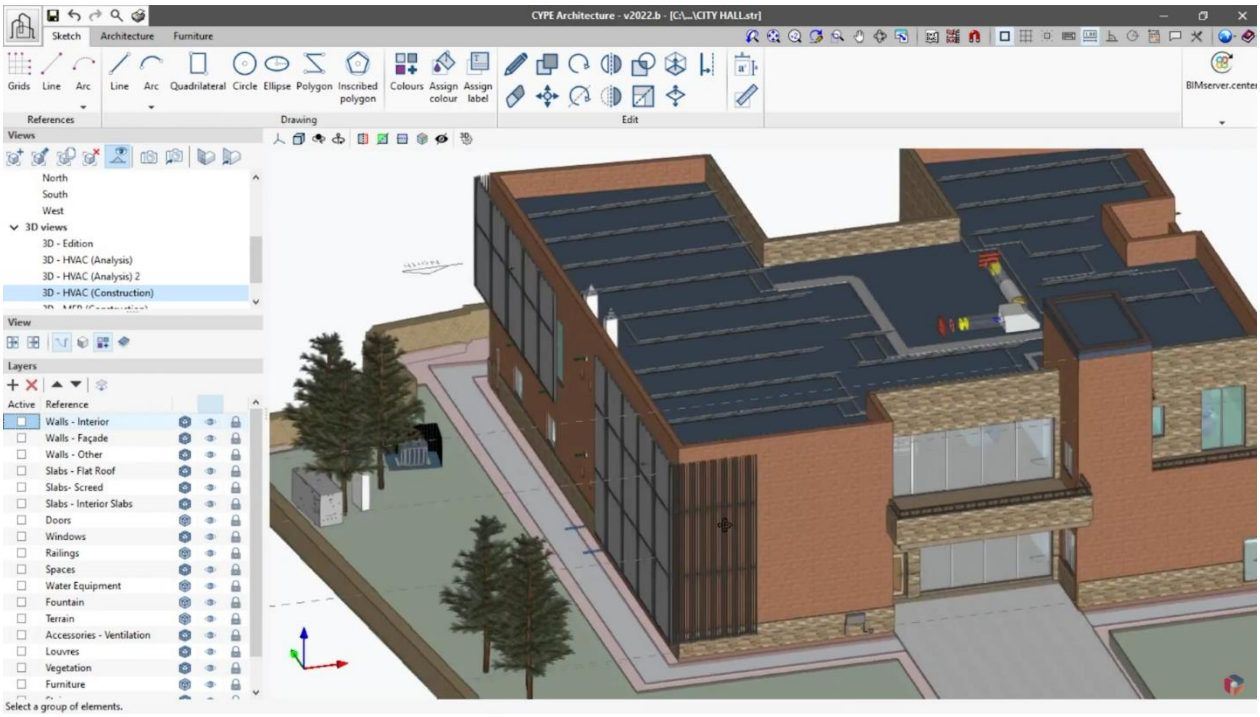

Figure 2: Example of the interface of the original CYPE BIM desktop software.

CYPE's large databases of source codes have been used to compose the initial stages of a prototype for the SATO BIM Desktop software with many plugged-in features. This separate desktop interface is being developed specifically in the context of the SATO project with focus on energy- and comfort-related assessments and optimizations. Figure 3 presents an initial overview of the graphical user interface from the SATO BIM desktop tool. This tool will enable the aggregated and disaggregated analysis and visualization of the assessments in non-residential buildings of the various applicable scales, as well as setting locations and specifications of energy-consuming equipment, sensors, and actuators.

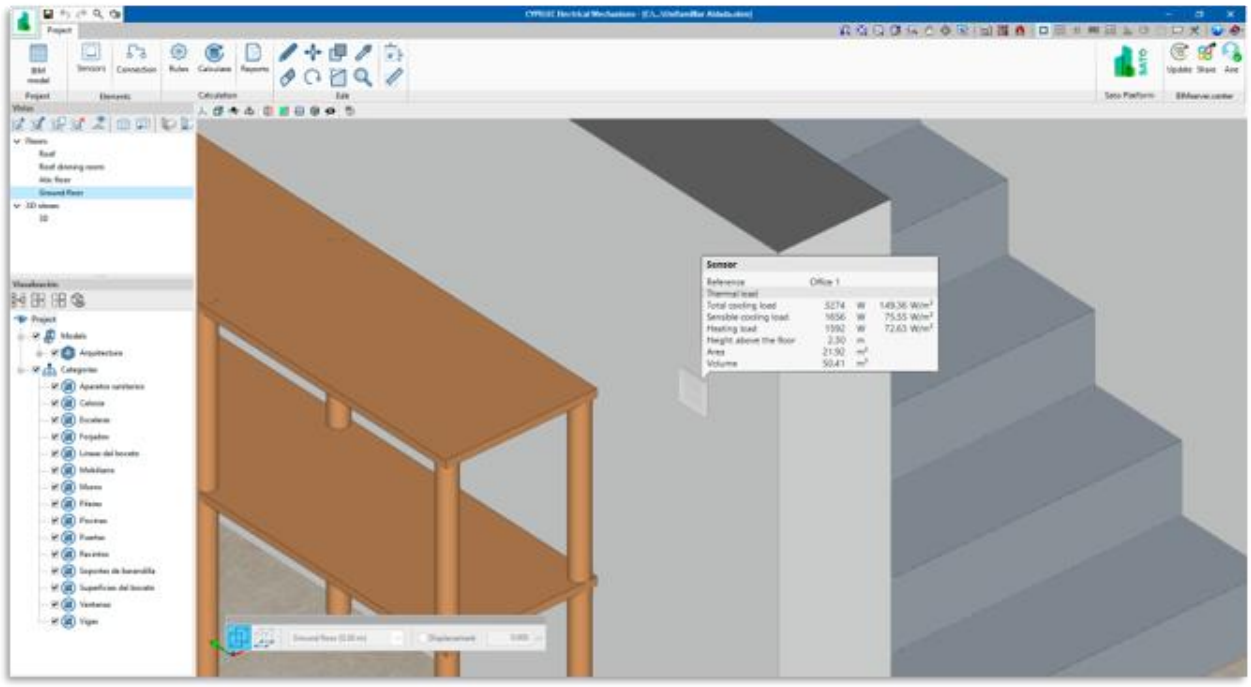

Figure 3: Example of the SATO BIM interface presenting the latest measurement of a sensor in the building model.

This tool will provide building owners and managers the virtual position of sensors and actuators in the respective building locations. This feature facilitates visualizing all devices that are currently integrated to the SATO platform in every SATO pilot. For instance, the device depicted in Figure 3 can interactively be selected in the SATO BIM tool to highlight its latest (few) reading(s). Other visualization modes (e.g., a 2D floorplan layout) or even space-based listings of devices can be implemented if their added-value justifies their implementation.

In the near future, the interface intends to make these latest/current measurements to be fetched on-demand by the desktop tool from the storage components in the SATO platform. This dynamic data will be fetched from the platform only when an actor wants to observe this data in the BIM model or to plot some graphs based on recent history. This desktop tool will also have the possibility of presenting simple charts from a subset of measurements (as can be seen in Figure 4) from the Refined storage component or colouring sections of the building according to the results from some self-assessments (as can be seen in Figure 5, which could be extremely useful for building managers) obtained from the Results storage component. All the possible interactions with the SATO BIM tool will be described in the Deliverable D5.1 (*BIM-based Interactive Applications Design*) from Task T5.1 (*Definition of actors interaction with SATO A&O services through BIM-based interfaces*) and Deliverable D5.3 (*BIM-based Interactive Applications*) from Task T5.3 (*Development of BIM-based interfaces for A&O services*).

In the design or remodelling phases of a building, the SATO BIM tool may also be able to suggest the best positioning of new devices to be inserted in the building to support some additional self-assessment and optimization. More specifically, this algorithm will be developed in Task T3.3 (*Development of BIM-based sensor placement assessment for desired level of SA&O of building and appliances*) and will be described in detail in Deliverable D3.3 (*BIM-based sensor location and placement assessments*).

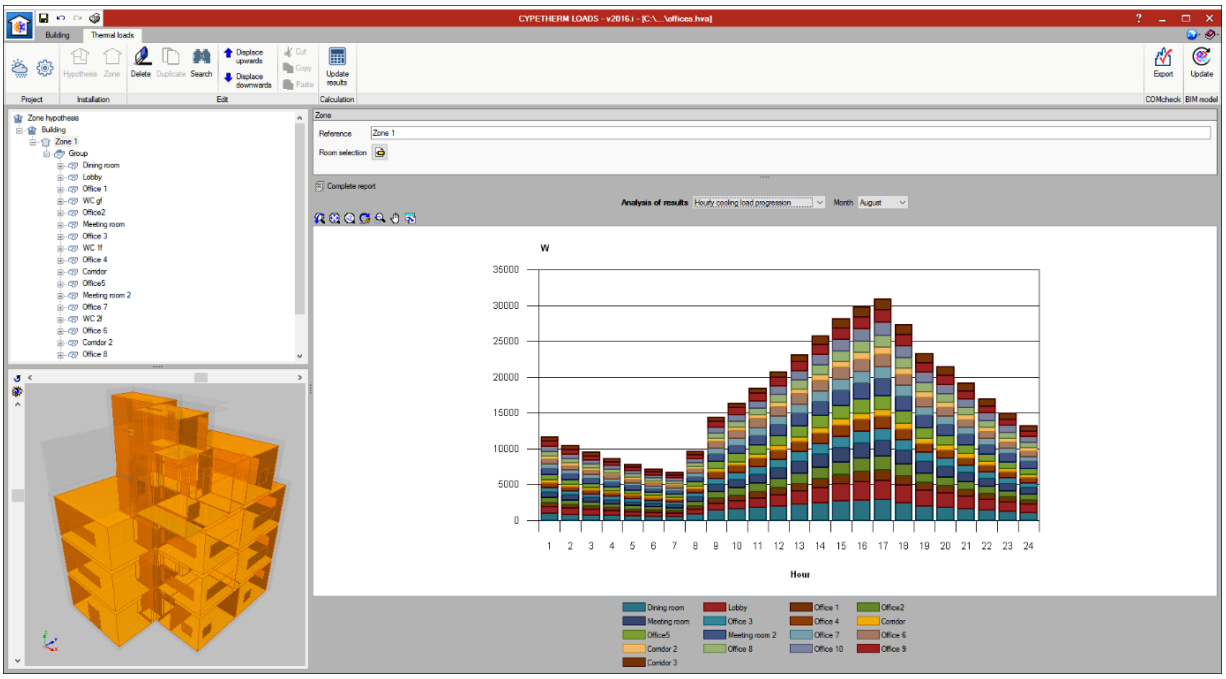

Figure 4: Example of an energy-consumption chart plotted in one of the CYPE BIM tools.

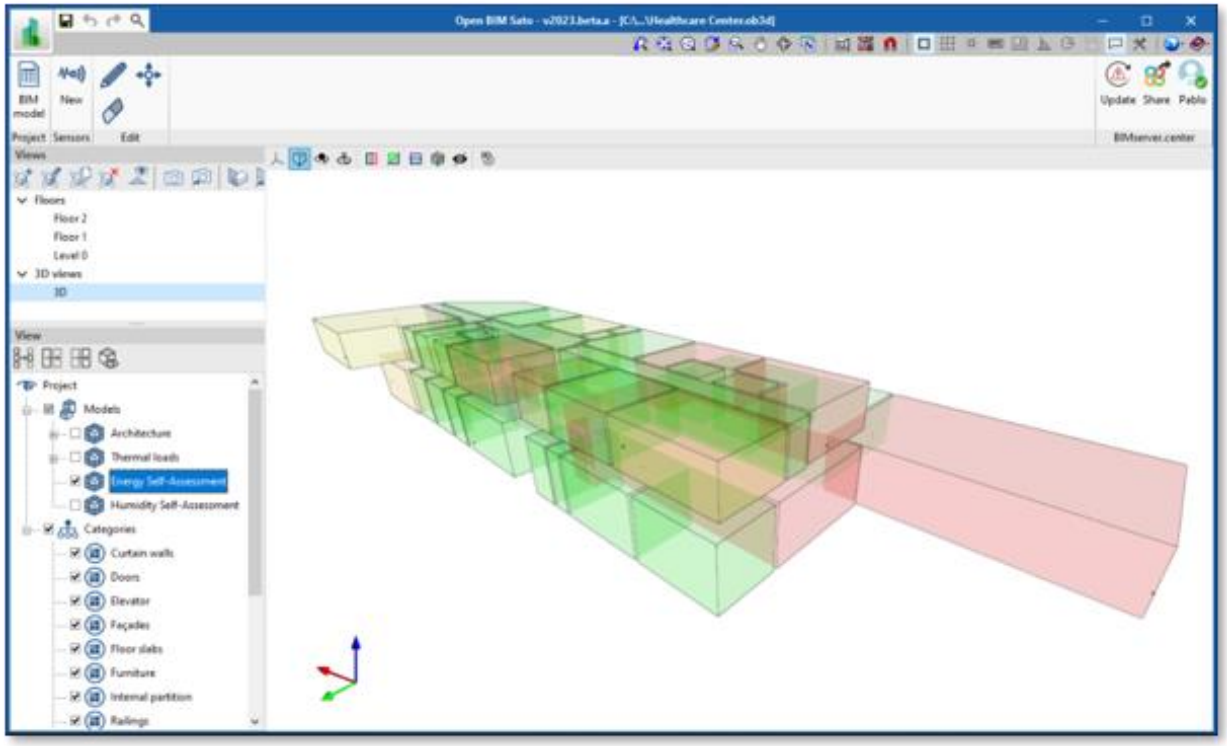

Figure 5: Example of self-assessments being presented in the SATO BIM tool from CYPE.

Many aspects of the SATO BIM desktop tool will still be defined and presented in the mentioned deliverables, but some interesting features can be mentioned in this section. For instance, an up-to-date version of the BIM model (or the dynamic values associated with device measurements) representing the whole building can be prepared periodically (e.g., every 15 minutes) and stored in the Result Storage component of the SAF, which is subsequently stored in the BIM cloud from CYPE. Alternatively, it can also be produced on-demand. Note that all data events will be stored in (and provided by) the appropriate

storage components of the SATO platform. Exporting this BIM model is only a visualization mechanism that replicates the current status of the building and its devices. The SATO platform will differentiate the static data associated to the building model from the dynamic data coming from sensors and devices, which can be considered as an additional layer that can be fetched separately.

Having access to such an enriched interface facilitates the job of building managers since they can evaluate the current state and behaviour of the building before making decisions, as well as overview the impact of recent actuations and decisions on the building and its devices through the time.

Having desktop tools as the SATO BIM tool present several advantages in terms of performance since it can employ more computational resources from the computer where it is installed compared to other type of applications that will be seen in the subsequent sections. This advantage vouches for the ability of editing complex 3D models in this type of tool. However, they need to be installed in the computer and may become dependent on the platform and software stack it uses. It is hard to make software interoperable for multiple operating systems. With this limitation in mind, SATO will also explore other interface types that facilitate simple tasks (e.g., visualizing the BIM model) in multiple operating systems and devices.

Mobile- and Web-based SATO App

Many other graphical user interfaces will be developed within the context of the SATO project to support specific interactions with the human players. Mobile and web graphical user interfaces are grouped in this section because SATO will prioritize cross-platform tools that can seamlessly be presented in traditional web browsers in desktop computers as well as in more modest browsers and frameworks in mobile devices. This flexibility is possible due to modern trends in web development that promotes preparing responsive interfaces that can adapt to different screen sizes and orientations.

One of these additional interfaces is a web-based BIM visualization tool provided by CYPE that can be embedded in websites and mobile apps. Figure 6 exemplifies this interface in two different mobile devices. Although this interface does not provide editing capabilities, it enables many stakeholders to visualize and interact with existing 3D models of the buildings. Examples of interactions include (but are not limited to) visualizing some assessments and a few latest measurements, as well providing simple actuation commands wherever possible.

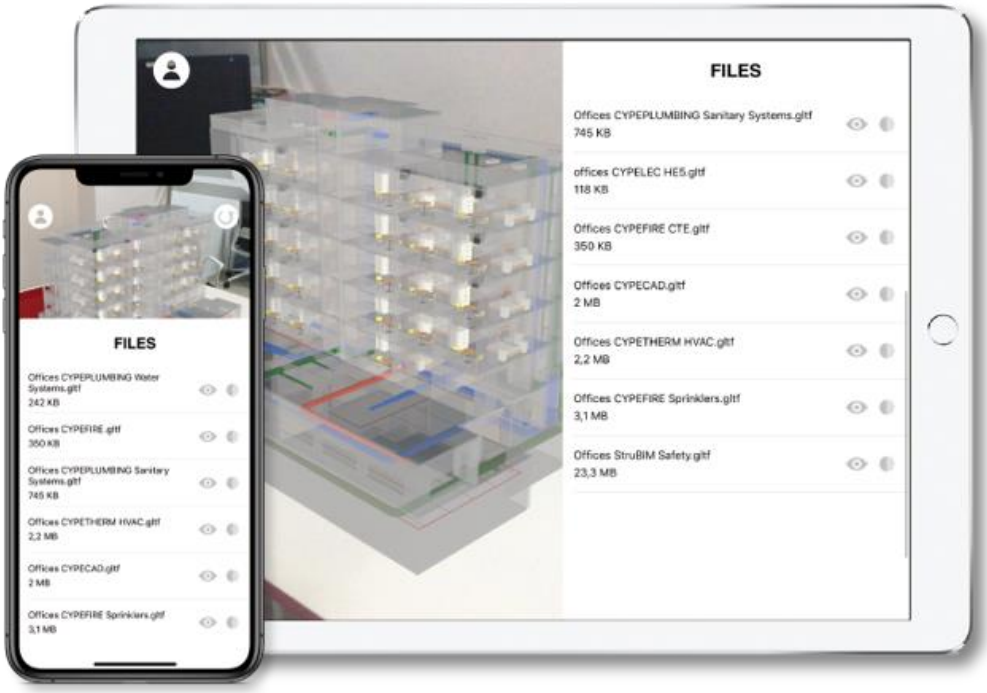

Figure 6: Mobile-base BIM visualization tool.

Monitoring dashboards are another class of web-based interfaces that will be extremely useful for the SATO project. This interface type is very adaptive and enables stakeholders to visualize multiple metrics, measurements, and assessments in near real-time. Examples of dashboard frameworks that can be used in the SATO platform include the Kibana [2], Logstash [3], and Grafana [4] (see Figure 7 for some graphical user interfaces made with these tools). Traditionally, these tools provide multiple chart types (e.g., lines, bars, pies, heat maps, point maps) and are very easy to integrate with in IoT environments. However, they also enable producing specialized data files (e.g., CSV tables) that can be downloaded and processed in other workflows.

Figure 7: Examples of web-based dashboards from Kibana, Grafana, and Logstash.

This type of interface complements the web-based BIM interface. Their differences reside in the fact that the BIM interface enables a 3D visualization of results, by locating them physically in the building. In a dashboard component, the results appear more in an aggregated or summarized version represented by numbers and charts. In both cases, data can be dynamically updated for visualization. Notwithstanding, the self-assessments services can provide updates on the observed metrics at a higher frequency using dashboards than the BIM interface because the former requires simpler calculations and results formats to be read.

Another type of graphical user interface that will be used in SATO is the mobile application approach. This type of interface enables users to interact with the SATO platform through handheld mobile devices for simpler tasks. Figure 8 depicts an example of the design process of a mobile application with its interface elements (e.g., figures, fields, menus). The SATO project foresees the design and development of generic mobile applications (e.g., both for Android and iOS) with well-known web app frameworks that can also be seamlessly used for mobile applications (e.g., the Meteor framework [5]). It will enable for instance building occupants to provide their preferences for the SATO platform that will respond and prepare the building according to these preferences, trying to optimize the building occupants' comfort.

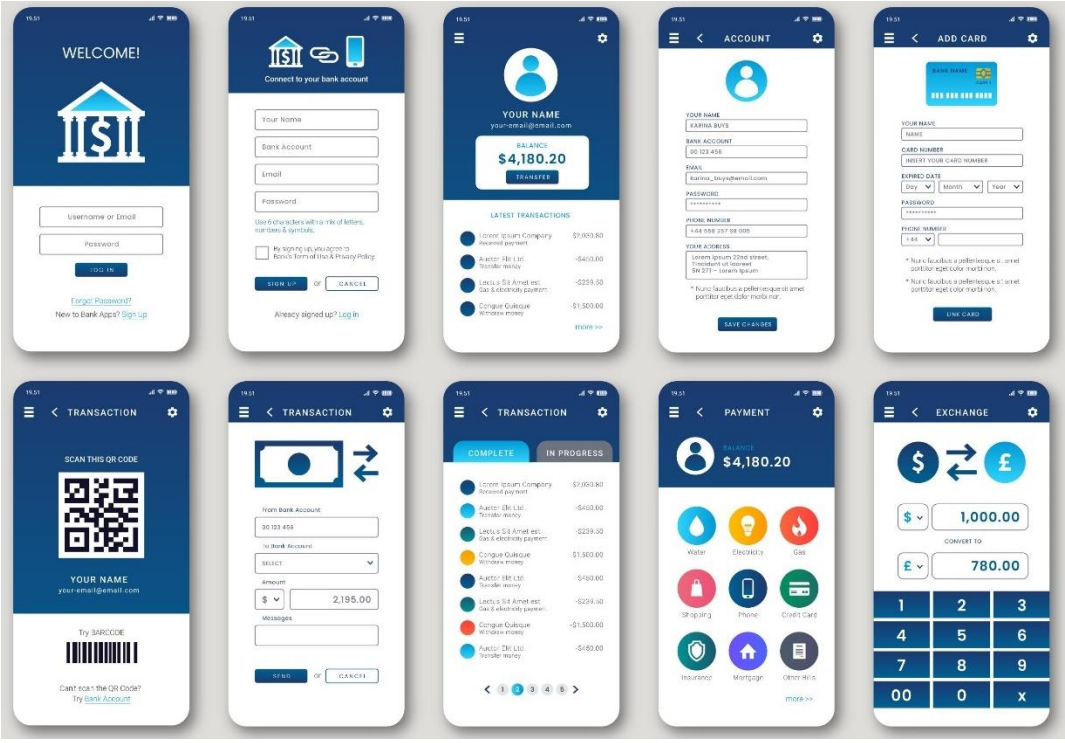

Figure 8: Example of a user interface of a mobile application. Image retrieved from FreePik.

These web-based and mobile interfaces will be designed and developed mostly in work package WP5 (*Development of SATO User Interaction and Interfaces*) of the SATO project. Their complete description will be available in Deliverables D5.2 (*Web-based Interactive Applications Design*) and D5.4 (*Web-based Interactive Applications*). They will provide the desired level of interaction with the self-optimization services to the stakeholders involved in the building's energy management and occupation. These stakeholders will be able to provide their preferences and configurations that, ultimately, will be considered when the self-optimization services decide upon the most appropriate controls for buildings and their devices through passive and active feedbacks.

2.1.2. CLI – Command-line interfaces

A Command Line Interface (CLI) is another type of user interface that connects a user to a computer program or operating system. Through a CLI, users interact with a system or application by typing text commands with the appropriate arguments and formats. The command is typed on a specific line following a visual prompt from the computer. The system responds to the text, and the user may then type on the next command line that appears. Through this pattern of command and response interactions, the user can issue a series of commands that are executed by the system or program.

Traditionally, these interfaces are used for installing, configuring, and managing applications rather than using them. However, there are CLIs that enable application users also to access or modify data stored in the application. Both cases are being considered in the SATO project. Examples of CLI that may be important for the SATO platform and its users include (but are not limited to) the CLI from the Docker [6] (a container management solution), Apache Kafka [7] (an event-based communication platform), and Apache Hadoop [8] (a MapReduce processing framework).

2.2. Application Programming Interfaces

Generically, an API is a set of definitions and protocols for building and integrating software applications. In this section, the focus will be on three main types of interfaces: Web APIs, Operating System interfaces, and Software libraries.

2.2.1. Web APIs

A REST API (also known as RESTful API) is a web-based application programming interface (API or web API) that conforms to the constraints of REST architectural style [9]. Usually, APIs become considered RESTful by providing a client-server architecture with stateless communication between the client and the server and implement the at least four methods for each resource: GET (i.e., retrieve a resource), PUT (update it), POST (create it), and DELETE (remove it).

One of the main REST APIs in the SATO platform will be the one from the Devices Services component, which provides a unifying way for managing and providing information about the devices in buildings integrated with the platform. Another example of REST API is the one provided by the Kafka software, which enables connecting the devices with the streaming components of the platform.

SATO Devices API

Syntactic and semantic interoperability were some of the main requirements identified for the SATO platform in Deliverable D1.3. Device Services and the Device Semantics are two critical components of the SATO platform that will respectively address these two requirements. They offer services that enable the interaction with heterogeneous devices and commercial platforms.

More specifically, the Device Services component provides a unifying REST interface to other components of the platform, abstracting the heterogeneity of all the integrated underlying platforms and smart systems. By doing so, the self-assessments and optimization services can more easily send actuations to the devices by only needing to be compatible with the interface of the Devices Services. The Device Semantics component stores and provides information about devices, platforms, and other soft sensors (e.g., the EPREL external data source) through ontologies, which persist data using semantic triples that form a graph with all its data representing the context.

When a request arrives at the Device services component, it is handled by the SATO API, identifying the corresponding SATO Integration module and forwards the request to it. Then, the SATO Integration module converts the request into specific requests according to the rules implemented and needed to exchange the request with the integrated platform.

The SATO Integration module (Figure 9) is a conceptual component associated with the Devices Services and the Data Catalog components in Figure 10 and follows a modular architecture that allows one to implement other mechanisms to interact with the integrated platform or device to deal with different platforms and vendors. Since some platforms or devices may use publish/subscribe mechanisms, this module can handle this type of communication. Additionally, and according to the platform, other communication mechanisms can be added to the module.

Since the SATO Integration module is responsible for dealing with the commercial IoT platform that will be integrated with the SATO platform, it is also responsible for mapping all incoming data. The SATO architecture defines a common data model (CDM) that is used inside the SATO platform. To address this requirement, the SATO Integration module needs to convert proprietary data formats into SATO data format, with the SATO CDM component being responsible for this conversion.

The SATO Device Services has an API that considers three main resources:

- **Users:** registered users that are authorized to interact with specific components in the platform.
- **Devices:** Uniquely identified devices that are connected to the platform for providing data or consuming actuations.
- **Services:** Represent the software components that will provide services to the devices and users.

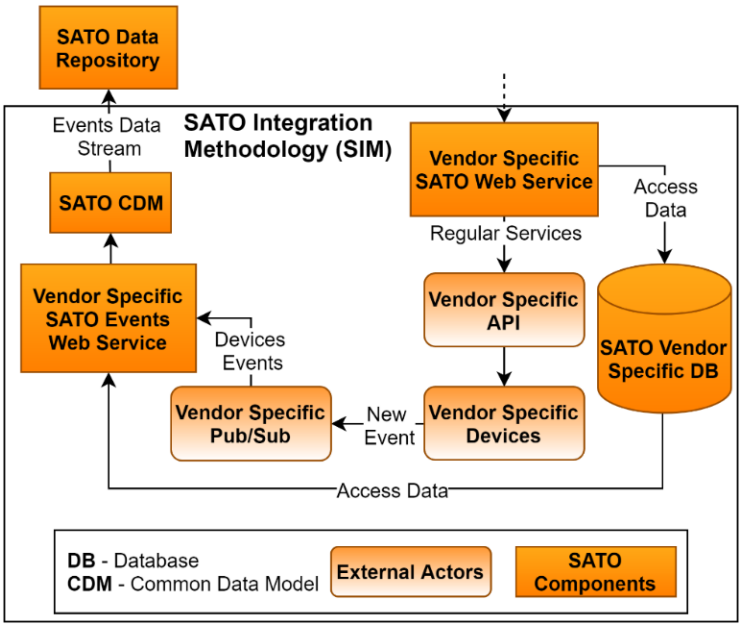

Figure 9: An overview of the SATO integration module

These three components are enough for structuring the REST interface for the SATO platform, which follows the logic of associating devices to users and services. Examples of provided operations include:

- **`POST /users`**: Creates a new SAT user and returns its SATO identifier, which will be used in the subsequent interactions.
- **`GET /users/fuser-idg/devices/fdevice-idg/services/fservice-idg`:** Get the status of the service for the specified device (which belongs to a given user).
- **`POST /users/fuser-idg/devices/fdevice-idg/services/fservice-idg`:** Modifies the service status to the indicated level.

Additionally, not all commercial IoT platforms can inject directly the users' data on the platform. For that reason, this type of platforms requires that there is a user record in the interface to register the access credentials to the proprietary interface of the platform. This limitation requires adding some service paths to allow the integration of this type of platforms. Examples of additional paths include:

- **`POST /users/fuser-idg/platforms`:** Register a new platform with the associated user's credentials.
- **`DELETE /users/fuser-idg/platforms/fplatform-idg`:** Remove a platform from the service.
- **`POST /users/fuser-idg/platforms/fplatform-idg/devices`:** Register a device for systems that do not have a platform for user registration.
- **`DELETE /users/fuser-idg/platforms/fplatform-idg/devices/fdevice-idg`:** Delete a specific device from the platform, which will stop receiving the events from this device.

Although this set of resources and interfaces satisfy the initial needs of the SATO platform, the device services component is extensible to satisfy additional requirements and services identified in the future. Regarding the implementation of the interfaces, existing frameworks were identified that follow the JAX-RS specification [10]. This specification facilitates the creation of REST web services through code annotation. Eclipse Jersey [11] was the REST framework selected for implementing the first prototypes of the device services API. Finally, Apache Tomcat [12] is the Web server to run the Web services created with the mentioned tools.

More details about the Device Services component can be seen in Deliverable D1.4 (*Description of the system architecture of the SATO platform*) from Task T1.3 (*Requirements and System Architecture for the SATO platform and for it to support SRI calculation*) and in the MSc thesis from the FC.ID student that developed it [13]. Its final implementation form will be described in the upcoming deliverable D2.5 (*SATO Platform*) from Task T2.4 (*Development and integration of the SATO Platform*).

2.2.2. Publisher/subscriber APIs

The streaming components adopted in the SATO Platform are Kafka instances that provide multiple interfaces for publish/subscribe methods. These methods allow other software components to register in the streaming components as publisher (i.e., to produce events) or subscriber (i.e., to consume events) in specific topics of interest.

Kafka provides the Kafka Connect, a declarative framework for connecting input and output data with Kafka. They require client components to provide a configuration with the connector class, URL, and some other parameters. Once a connector is running, it sends (resp. receives) events to (resp. from) the selected topics, making data flow from the devices in buildings to the processing toolbox (e.g., libraries, frameowrks) that will support the self-assessment framework (SAF) of the SATO Platform.

These Kafka Connectors are scalable and provide fault tolerance to avoid loosing events in the communication path. Additionally, they decouple the source and the sink destination of data. For instance, the MySQL connector from Debezium [14] enables monitoring changes in relational databases and forwarding these changes to Kafka as soon as they happen. Other prominent examples include connectors to ElasticSearch, neo4j, S3, and MQTT. The Confluent Connector Hub [15] provides hundreds of connectors to integrate Kafka with the many traditional tools used in streaming pipelines (including most of tools that SATO will use).

Finally, Kafka Connectors also provide simple (stateless) transformations with the SMTs – Single Message Transforms. If one needs stateful tranformations, then Kafka Streams are a more appropriate solution. SATO is currently exploring how to replace parts of the REST architecture described in Section 2.2.1 with Kafka Connectors and Streams to obtain better scalability and fault tolerance.

2.2.3. Filesystem Interfaces

Although IoT architectures heavily rely on stream communication and processing, storage components are imperative for persisting data and analyses results for the long run. The SATO platform is not an exception in this aspect since it contains at least four storage levels in its data pathway. These storage components are the Transient Storage, the Refined Storage, the Structured Storage, and the Results Storage.

The Transient Storage component is responsible for consuming and ingesting information from devices in buildings (and other sources) as fast as possible to reduce the chances of the SATO platform losing any control or data events. Information is kept in this component only for a timeframe window, while it clears expired data after the expected data lifespan. Notwithstanding, Apache Kafka provides sufficient mechanisms for storing transient data for a specific period or size limit. It uses lower-level filesystem interfaces to persist data on disks during this time.

The Refined Storage component is considered the trustworthy persistent storage of all control and data events coming from the devices and buildings or the external data sources. Data stored in this component will have passed through the Data Catalog component to be standardized using the Common Data Model (CDM) and, opportunistically, will have passed through the Data Quality and Data Enhancement components (when configured for doing so). This storage component also relies in lower-level filesystem interfaces from the operating system to persist data on disks while they have space. Components from upper layers in the platform (e.g., the SAF) will access data stored in this component when they need to process the events the SATO platform has received.

Other (usually higher-level) filesystem interfaces will be provided by the Structured Storage since this component will translate the original/enhanced events into other structured formats that are required for specialized processing. For instance, InfluxDB loads time series data into data models that can be queried with structured queries similar to the ones from relational databases.

Finally, the Results Storage is a component that will collect the outcomes from the self-assessments and will provide them to the upper layers of the architecture, namely self-optimization services and to the actors via the user interfaces. It may contain several storage configurations, tools, and drivers, where these different storage instances can provide the appropriate service levels and security protection according to specific data types and data utilization. For instance, the BIM models can be stored in this component before they are sent to the BIM cloud server (the storage component that is read when a model is loaded in the BIM editing or visualization tool).

Other remote filesystem interfaces can be leveraged at any of the mentioned storage components. There are multiple solutions available for different remote or distributed storage systems. Examples include the IPFS (InterPlanetary File System) with focus on interoperability, LustreFS for HPC (High-Performance Computing), s3fs for direct AWS S3 access, SSHFS for filesystem on top of SFTP channels, and even Vawlt.io (a spin-off from FC.ID) for resilient multi-cloud storage.

Most of the mentioned filesystem interfaces resort to a software interface from the operating system called FUSE (Filesystem in USErspace). This interface enables non-privileged users to create their own file systems without having to modify the OS kernel. Basically, this interface bridges user-level functionalities to the appropriate kernel interfaces.

2.3. Other interfaces

There are other types of interfaces that were not discussed in this document either because they are not going to be used by the SATO project or they are simple interfaces that do not require a detailed description on how they work.

SMS (Short Message Service) or e-mails are two examples of these additional interfaces. They add up to the SATO Web App interfaces because they can be used to emit alerts or send recommendations to the appropriate stakeholders with a direct link to the suggested service or action, reducing the effort to make changes happen.

3. Interfaces in the components of the SATO Platform

All the previously described types of interfaces have been considered in the design and the development of the components of the SATO Platform. Some components provide a single interface for stakeholders, while others may provide multiple ones for the diverse types of interactions they will support. Figure 10 depicts exactly the type of interfaces that are provided by each of these components.

In this Figure 10, the interfaces are presented as one of the following types:

- **G**: Graphic User Interfaces (Mobile and Web-based), as presented in Section 2.1.1.
- **R**: REST APIs, as presented in Section 2.2.1.
- **P**: Publisher/subscriber is a specific type of API from the streaming components, as explained in Section 2.2.2.
- **F**: Filesystem APIs provides interaction with the storage components, as presented in Section 2.2.3.
- **O**: Other interfaces encompass the remaining interfaces that do not fit in other categories, such as the ones mentioned in Section 2.3.

The detailed description of each component of the SATO platform is available in Deliverable D1.4 (*Description of the system architecture of the SATO platform*).

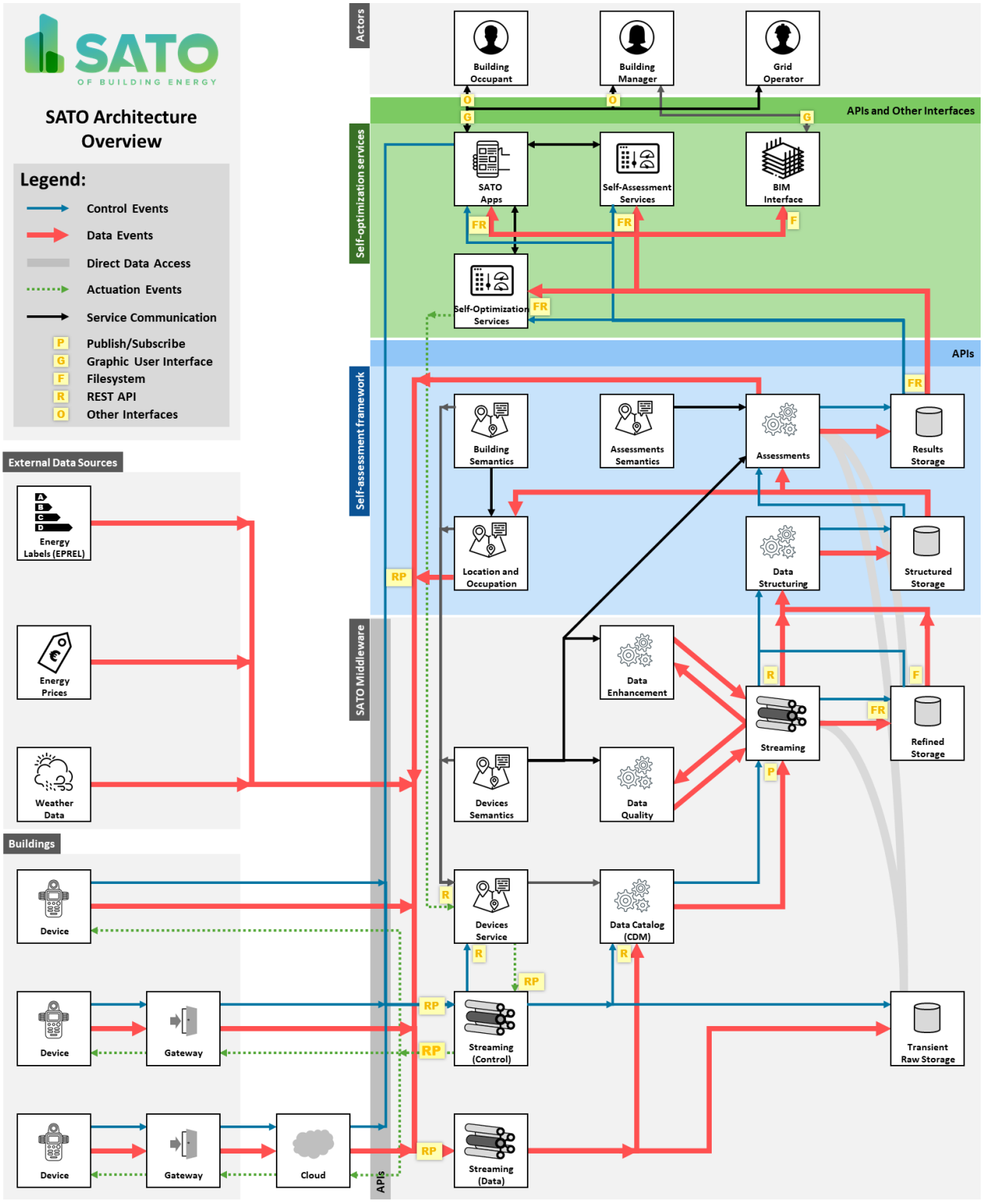

Figure 10: Overview of the SATO platform with the foreseen positioning of its interfaces

4. Final Remarks

This deliverable presented an overview of the many interfaces that will be present in the SATO platform to enable the main stakeholders of the project and the software components of the platform to interact with each other. It introduced the main six categories of interfaces of interest to the project: Graphical User Interfaces (GUI), Command-Line Interfaces (CLI), Web Application Programming Interfaces (Web API), Publisher/Subscriber Interfaces, Filesystem Interfaces, and other (e.g., email and SMS).

In Figure 10, these interfaces are positioned in the corresponding building blocks of the SATO platform, which identifies the status of the platform prototype. In the near future, other interfaces may be included (or some may be replaced) according to the maturity evolution of the SATO platform with the next development iterations.

Bibliography

[1] CYPE Ingenieros, «CYPE Software for Architecture, Engineering, and Construction,» 2021. [Online]. Available: http://www.cype.com/en/.

[2] Elasticsearch B.V., «Kibana,» 2021. [Online]. Available: https://www.elastic.co/kibana/.

[3] Elasticsearch B.V., «Logstash,» 2021. [Online]. Available: https://www.elastic.co/logstash/.

[4] Grafana Labs, «Grafana,» 2021. [Online]. Available: https://grafana.com/.

[5] Meteor, «Meteor Software: A Platform to Build, Host, Deploy and Scale Full-Stack Javascript Applications,» [Online]. Available: https://www.meteor.com/.

[6] Docker Inc., «Use the Docker command line.,» 2021. [Online]. Available: https://docs.docker.com/engine/reference/commandline/cli/.

[7] Confluent, Inc., «Apache Kafka® CLI : Command Example,» 2021. [Online]. Available: https://docs.confluent.io/platform/current/tutorials/examples/clients/docs/kafka-commands.html.

[8] Apache Software Foundation, «Hadoop Commands Guide,» 2021. [Online]. Available: https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/CommandsManual.html.

[9] R. T. Fielding, «Architectural Styles and the Design of Network-based Software Architectures,» PhD thesis. University of California, Irvine, USA, 2000.

[10] I. Grimstad e S. M. Stark, «Jakarta RESTful Web Services,» 2021. [Online]. Available: https://projects.eclipse.org/projects/ee4j.jaxrs.

[11] Eclipse Foundation, «Eclipse Jersey,» 2021. [Online]. Available: https://eclipse-ee4j.github.io/jersey/.

[12] The Apache Software Foundation, «Apache Tomcat,» 2021. [Online]. Available: http://tomcat.apache.org/.

[13] V. B. Ferreira, «Middleware de integração de diferentes plataformas e sistemas inteligentes IoT,» MSc thesis. Faculty of Sciences of the University of Lisbon, Lisbon, Portugal, 2021.

[14] Debezium Community, «Debezium: Stream changes from your database,» 2021. [Online]. Available: https://debezium.io/.

[15] Confluent, Inc., «Confluent Hub: Discover Kafka connectors and more,» 2021. [Online]. Available: https://www.confluent.io/hub/.