SpaceOps-2023, ID # 497

# PintaOnWeb - The Front End of GSOC's Next Generation Mission Planning Systems

**Armin Wiebigke[a]*, Jonas Krenss[a], Jens Hartung[a], Sebastian Wiesner[a], Rainer Nibler[a], Anna Fürbacher[a]**

[a] *Mission Technology Department, German Space Operations Center (GSOC), Deutsches Zentrum für Luft- und Raumfahrt e.V. (DLR), , Münchener Str. 20, 82234 Weßling, Germany*
*firstname.lastname@dlr.de*
* Corresponding Author

**Abstract**

PintaOnWeb is the next-generation user interface of the generic mission planning tool suite at GSOC. It is a web-based application, which is mostly connected to a dedicated planning system based on GSOC's Reactive Planning framework, where PintaOnWeb takes its part as an "inspection window" into the planning system. Moreover, PintaOnWeb allows for modifications of the planning model, so that the automated planning can be complemented by manual modifications if needed. As a consequence, PintaOnWeb can be used for fully automated planning as well as fully manual planning, or anything in-between. Since it runs independently of the core planning system, we can easily deploy multiple instances with different configurations or to different locations, such as to a cloud environment that allows it to scale based on the demands of the users. The approach of developing a web-application required several design and architectural decisions and is performed in an agile development process based on evolving requirements. Both topics are described in the paper at hand, besides giving a detailed insight into the new tool, outlining the chosen technologies and available features.

PintaOnWeb is already part of the fully automated planning system for the Earth observation mission EnMAP which launched in April 2022. Here, PintaOnWeb both helps the planners to get visual feedback for any implemented change, and serves as a display of the planning system to operators of other subsystems. Furthermore, it forms the basis of GSOC's novel "Integrated Terminal and Antenna Scheduling" application. One of the possible future applications is the planning and scheduling of satellite constellations, such as for the next generation of the global navigation satellite system Galileo.

**Keywords:** Mission Planning, Sequence of Events, Scheduling Software, Web Application, User Interface

**Acronyms/Abbreviations**

Application Programming Interface (API)
Consultative Committee for Space Data Systems (CCSDS)
Continuous Integration/Continuous Delivery (CI/CD)
Environmental Mapping and Analysis Program (EnMAP)
Global Navigation Satellite System (GNSS)
German Space Operation Center (GSOC)
Integrated Development Environment (IDE)
Integrated Terminal and Antenna Scheduling (InTAS)
JavaScript Object Notation (JSON)
Launch and Early Operations Phase (LEOP)
Mission Planning System (MPS)
PLAnning TOol (PLATO)
Program for INteractive Timeline Analysis (PINTA)
Sequence of Events (SoE)
User Interface (UI)

## 1. Introduction

Mission Planning at GSOC has a long tradition. Over the last three decades, several fully- and semi-automated planning systems have been created to support the generation of mission timelines and SoEs for different kinds of spacecraft operated at GSOC and also for external partners. In addition, not only spacecraft-related planning issues were covered with the developed planning software, like the scheduling of ground station passes or the generation and maintenance of on-call shift plans for the operations teams[1].

The most used graphical application is PINTA[2], the development of which started about two decades ago. To support the presentation of the timelines generated with PINTA to a wider audience, a web-based viewer called TimOnWeb [2] was developed. Both tools were and are used for several missions like GRACE, TET-1[3], BIROS[4], GRACE-FO, EuCROPIS, PIXL-1, V3C[5] and, in the SoE editor customization[6], to support the LEOP or spacecraft maintenance activities for missions like TerraSAR-X/TanDEM-X, BIROS, HAG-1 (also known as H36W-1), PAZ, GRACE-FO, EuCROPIS, EDRS-C and EnMAP[7]. Furthermore, PINTA was extended to be used for the Long- and Mid-Term planning of the Galileo Constellation[8]. Mainly with the last project some bottlenecks like maintainability for large spacecraft constellations, the support of multiple users (working on the same timeline at the same time), and cross-plattform support (PINTA is bound to Windows systems) were discovered, while in parallel the capabilities of web-based software development were increased and became more attractive, which led to the decision to develop a successor called PintaOnWeb. In addition, a graphical viewer with editing possibilities for GSOC's Reactive Planning Framework[9–11] was needed. As this is using the newer planning library Plains (Planning in Scala) while TimOnWeb was built on the predecessor PLATO, setting up a novel web-based application had more advantages than reusing the already existing one.

In this paper, we provide a detailed insight into PintaOnWeb's functionalities and design, as well as its development process, which follows agile practices since the beginning. Furthermore, we explain the deployment process for releasing a new version within minutes, so changes can be deployed to the productive environment with minimal delay. As PintaOnWeb is already part of the fully automated planning system for the Earth observation mission EnMAP which launched in April 2022, we demonstrate the usage of PintaOnWeb in an operational context, where the planners immediately get visual feedback for any implemented changes and operators of other subsystems get an always up-to-date display of the planning system. Additionally, we give some insights into future applications as the planning and scheduling of satellite constellations, such as for the next generation of the global navigation satellite system Galileo, or the scheduling of the shifts for on-call support for multiple missions. At the end, we summarize the already implemented key features and processes and provide an outlook of the planned functionality extensions.

## 2. Overview

PintaOnWeb is a web-based interactive planning and visualization tool, accessible at any time and from any device, enabled by a well-designed new service- and web-oriented application architecture. Compared to its predecessor PINTA, which is a locally running Windows application, PintaOnWeb is easily accessible without any needs for installing or updating. Since computation-heavy calculations are done on the server, it can be used from any device without affecting the productivity of the user. Through PintaOnWeb, multiple users have access to the same planning system at the same time. This allows for easier collaboration, since everyone can view the latest state. Compared to locally stored states, that may be shared by exporting and importing files, this allows for fast feedback from all involved parties. By incorporating user authentication and mission-specific access rights, users may be allowed to only access parts of the planning model based on their rights. This allows for different levels of access, such as read-only mode or full write access, based on the needs and permissions of the user's role.

A core feature of PintaOnWeb is the representation of planning information in a graphical, easy to understand way. PintaOnWeb can show timeline entry plots, that display planned activities, and resource plots that display the state of a resource over time. Customizable plot configurations can be stored as a workspace, which can also be shared

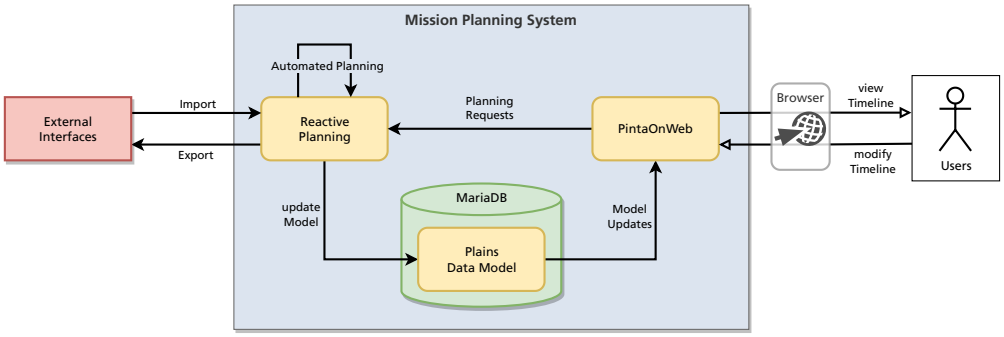


**Figure 1 A mission planning system based on PintaOnWeb, Reactive Planning and Plains**

easily across multiple users, allowing different views into the planning system based on the uses cases. The old web interface TimOnWeb only provides a relatively simple view of the timeline. In comparison, PintaOnWeb not only provides much more detailed information about the planning model, it also allows modification of the planning state, such as rescheduling a planned activity. PintaOnWeb is not a complete planning system itself, instead it builds on top of a Reactive Planning system and a Plains planning model, see Figure 1. All data is stored in a Plains model, which provides basic functionality like conflict checking. The Reactive Planning system is responsible for automated import and export of planning information, as well as all automated planning. Any changes made by a user via the PintaOnWeb web interface are also processed in the Reactive Planning system, ensuring a consistent planning state at all times. Multiple instances of PintaOnWeb may access the same Reactive Planning system with different configurations, such as a publicly available instance and one for the internal network. Simultaneously, a single PintaOnWeb instance can connect to multiple planning systems for different missions, thus allowing configurations based on the individual use cases. Together, PintaOnWeb, Plains and Reactive Planning provide a generic tool suite for creating planning systems. Their aim is to provide high reusability of generic functionality to minimize the effort to set up a new planning system. At the same time, custom components can be integrated to implement mission-specific requirements. This tool set replaces the old one based on PINTA, PLATO and TimOnWeb. While these have been successfully used in many missions, features often needed to be implemented in inflexible, mission-specific components, which increased the development and maintenance effort in the long run. For PintaOnWeb, the goal is to implement almost all features in a generic way as part of the core PintaOnWeb software, and only have mission-specific extensions for unique use cases.

One important aspect of the system design are the interfaces between the mission planning system and other subsystems. In general, the goal is to rely on general, standardized interface descriptions, primarily these defined by the CCSDS[a]. The Reactive Planning Framework provides generic implementations for data import through file ingestion, for which only configuration is necessary. Additionally, a message based interface is supported, which allows for easy integration of the system in a service-oriented architecture, both for collecting information from other services and exporting planning information on demand. One advantage of this approach is that importer and exporter implementations can be reused across multiple missions, with only minor adjustments necessary.

### *2.1 Model Connection*

PintaOnWeb displays a Plains model, which can come from different sources. To connect a Plains model to a PintaOnWeb instance, it uses a so-called *model connector*. Currently, two kinds of model connectors are implemented:

- patch files: The model connector reads Plains patch files from a specified directory. Each patch file describes the changes from one model version to the next one. Every time a new patch file is added, the model connector sends the updated model to PintaOnWeb.
- Reactive Planning: The model connector reads the model from a database to which a Reactive Planning system writes model updates. Every time an update is written to the database, the model connector sends it to PintaOnWeb.

For a full planning system, the nominal operation mode is expected to be a connection to a database managed by a Reactive Planning system, since this setup allows for full utilization of the planning features. However, there are use cases where a simple patch file based model connector may be more suitable. For example, a Reactive Planning system, running inside an internal network, can export all model updates as patch files. Then the patch files can be transported to a PintaOnWeb instance running in another network, using an existing file transport solution. This solution allows for a straight-forward setup of read-only PintaOnWeb instances in various network contexts, with no sophisticated implementations necessary.

Each model connector is responsible for exactly one Plains model. PintaOnWeb is able to use multiple model connectors at the same time, so a single PintaOnWeb instance can serve multiple missions. At any time, PintaOnWeb displays one model, and a user can switch between the available models with a simple mouse click.

### *2.2 Distributed and Redundant System*

The generic GSOC planning tools can be deployed as independent subsystems: The Plains planning model is stored persistently in a MariaDB database, the Reactive Planning system ingests input and updates the planning model, and PintaOnWeb displays the planning model by accessing the database. This system design makes the deployment very flexible and allows for greater resilience. For example, we can use the existing enterprise solutions for MariaDB to achieve a highly available, redundant storage of the data. Using multiple Reactive Planning and PintaOnWeb instances,

[a]https://public.ccsds.org

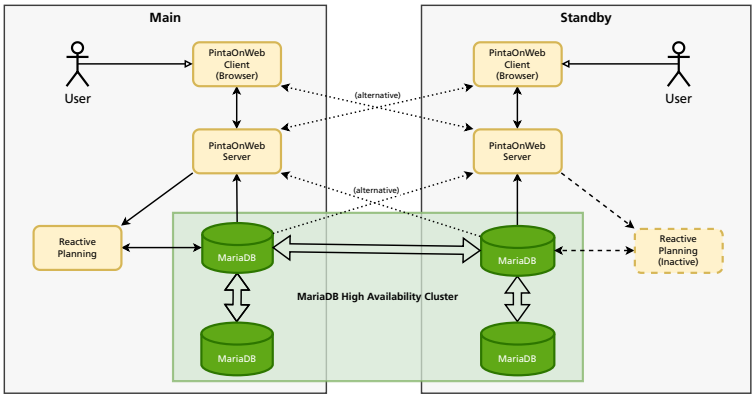


**Figure 2 A distributed and redundant planning system**
Multiple instances of the Reactive Planning system and PintaOnWeb run independently, synchronizing over the Plains model that is stored in a MariaDB, which can utilize existing enterprise solutions for data redundancy.

we can also deploy a distributed system, in which a standby system can take over the operation in case the main system fails or is unavailable for some other reason. See Figure 2 for an example how such a system could look like.

## 3. Automated Planning Process

When combined with a Reactive Planning system, PintaOnWeb is well suited as a part of a fully automatic planning system. In this configuration, the Reactive Planning system takes care of the entire planning process, such as ingesting inputs, planning a timeline and exporting commands, while ensuring that the planning model is always kept in a valid state. All data transfer from and to the system and between its internal components is automated per default, which means that a planner does not have to manually take care of transferring data. PintaOnWeb then just serves as a viewer of the planning model, without any manual modifications anticipated. Nonetheless, it provides detailed information about the state of the model and timeline, making it an invaluable tool for both mission planning engineers and operators.

On the other hand, PintaOnWeb's interactive features also allow manual modifications of the planning model. A planning engineer can trigger the import of files, reschedule activities, create new activities, etc. This way, the planning process could also be handled manually. Each change by a user is sent to the Reactive Planning system, where it is processed like any other input (with regard to a customizable message prioritization). If the change is successfully integrated into the model, the updated model is immediately sent to all connected clients. Resulting conflicts in the planning model, that is violations of the defined constraints, are directly visible to the user. This gives the users very fast feedback for all implemented changes, so they can work interactively with the planning model, without requiring time-consuming manual procedures to check for conflicts.

Combining the automated system with manual user interactions allows the realization of a semi-automated planning process, which uses automated imports and algorithms for many cases, but in certain cases may still require additional information or approval by manual interaction. PintaOnWeb's capabilities allow for an arbitrary mix of manual and automated planning, for instance by implementing a new use case as a manual planning procedure at first, and then later replacing it with an automated solution. The combination of PinatOnWeb and Reactive Planning is thus suitable for a wide range of possible missions.

Since PintaOnWeb is implemented as a web application, no installation is required and new versions can be rolled out and distributed to all users within minutes. Through its design as a mostly automatic system, with the option for manual web-based interaction, the number of manual procedures can be minimized, allowing the personnel to make efficient use of their time.

## 4. User Interface

PintaOnWeb is designed as a user-friendly and customizable graphical user interface, featuring visualization and interactive modification of a planning model. It is based on state-of-the-art web development technologies and frameworks, with a focus on usability, responsiveness and maintainability, and implemented as a single-page web application. In this section, we will explain the most important UI elements and their functionalities.

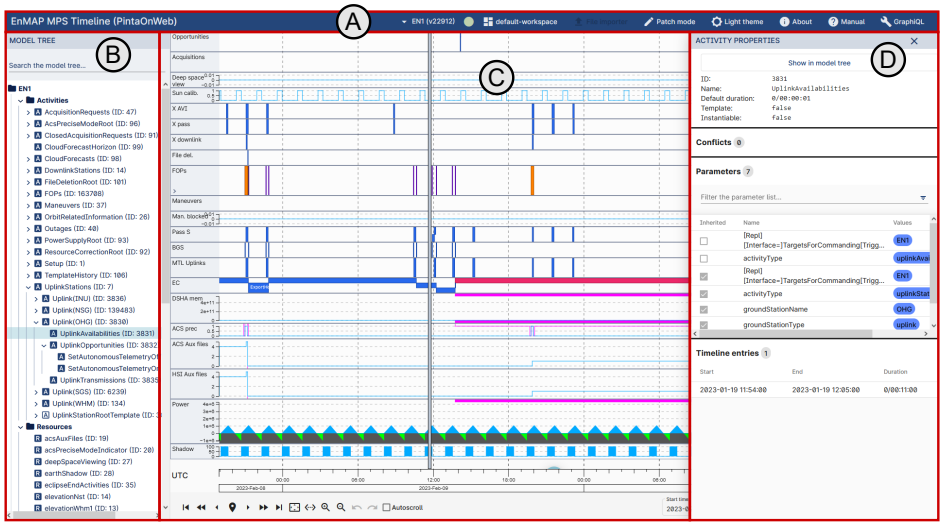


**Figure 3 PintaOnWeb graphical user interface**
The user interface with the main sections highlighted in red:
The toolbar (A), the model tree (B), the timeline view (C), and the properties view (D).

*4.1 Toolbar*

The toolbar at the top of the screen (see A in Figure 3) provides quick access to many important features. First, it let's the user switch between the available projects and workspaces, as well as turning the editing mode on or off. The toolbar also contains a theme switcher, to choose between the light and the dark theme. There are links for accessing the PintaOnWeb user manual and the GraphiQL web-interface (explained in Section5.1), which both open in an additional browser tab.

*4.2 Model Tree*

The model tree (see B in Figure 3) displays the core structure of the planning model in a compact and intuitive way, similarly to the directory/file structure of a file system. It has two sections for the most important types of entities, activities and resources. In a Plains planning model, each activity may have a parent relationship to one or more other activities, circular relationships are not allowed. This way, the activities form a tree-like structure, although an activity can appear multiple times in the tree if it has multiple parents. Resources do not have parent/child relations, so they are always shown as a flat list. The nodes of the model tree can be expanded and collapsed, thus offering the user a very fast and intuitive way to traverse and explore the entire project.

The model tree can become quite large and complex, depending on the underlying mission, so an important feature is to search for specific information. The model tree view contains a search input, that filters for nodes matching a given search expression, see Figure 4. Additionally, all parent nodes are displayed, so the correct structure of the model tree is always preserved. It is possible to search for activities and resources based on their name, their parameter values or their object reference, by filtering for values that match a given regular expression. Additional search filters are planned to be added in the future.

*4.3 Timeline View*

The central part of the user interface is the time line view (see C in Figure 3). Here, information for a selected time range is displayed using customizable plots of different types. There are currently two types of plots implemented. Further types of plots are planned to be added in the future, such as a ground track plot that visualizes the location of a satellite on a world map. A vertical line indicates the current time, either the real time or a configured simulation time.

*4.3.1 Timeline Entries Plot*

A Timeline Entries Plot displays the activities planned in the timeline, with each plot only showing activities that match a customizable filter. All colors of the plot, such as the timeline entry box color, the lower and upper bound,

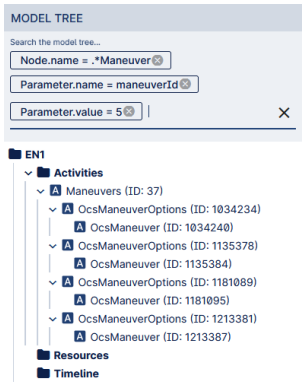


**Figure 4 Model tree search**
The resulting model tree of a search query.

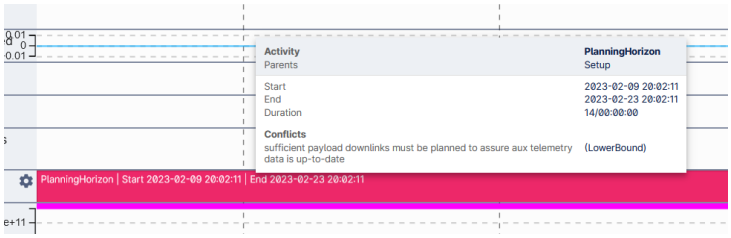


**Figure 5 A conflicting timeline entry with its tooltip**
The activity is marked red because it is part of a conflict. The tooltip provides a description for the cause of the conflict.

etc. can be modified. Activities with a conflict (some planning model constraint is not fulfilled) are highlighted in another color, by default in red. This makes it easy to recognize conflicts in the planning state. The constraints which are responsible for the conflict are displayed in a tooltip when hovering over the timeline entry, see Figure 5.

### *4.3.2 Resource Plot*

A Resource Plot displays the state of a specific resource, such as the charging state of a battery. This includes the current value of the resource, its upper and lower limits (from physical/logical limitations) and its upper and lower bounds (from constraints). The plots can be customized in various ways, including the minimum and maximum y-value.

### *4.3.3 Plot Hierarchy*

The plots are arranged in a tree-like structure: Each plot can contain other plots as subplots, each of which may also contain subplots, and so on. Subplots can either be visible or collapsed, which allows displaying specific planning information with the click of a button. If a Timeline Entries Plot has a subplot that is also a Timeline Entries Plot, then the subplot color configuration is applied to all activities that match the subplot's filter. This way, the activities in the parent plot are automatically displayed in the color inherited from the subplots. For example, consider a plot that contains all visibilities of a satellite with all available ground stations. That plot may contain one subplot for each ground station, where only visibilities with that ground station are displayed, see Figure 6. By giving each ground station subplot another color, it is immediately visible in the parent plot to which ground station each visibility belongs.

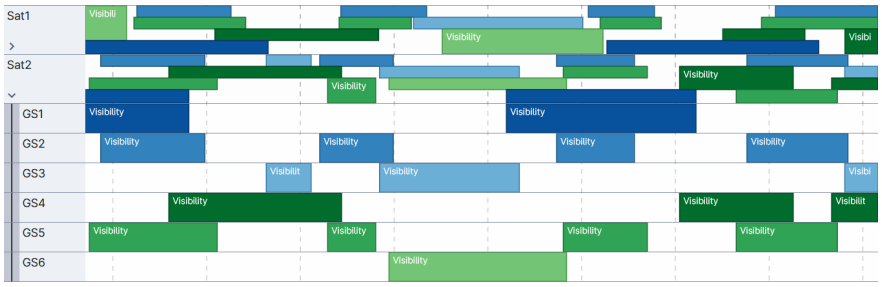


**Figure 6 Color propagation across subplot hierarchy**
A hierarchy of timeline entries plots displaying the visibilities between the satellites (Sat*) and ground stations (GS*). Shown are the subplots of the plot for satellite *Sat2*, one for each ground station. The colors from the subplots distinguish the different activities in the plot.

*4.3.4 Timeline Navigation*

Underneath the plots, a time scale displays the time horizon that is currently visible, with markings for specific time intervals (such as days) depending on the zoom level. The user can zoom and scroll the time horizon using either the mouse, the keyboard, or navigation buttons.

*4.3.5 Workspaces*

The color configuration and hierarchy of the plots is stored as a *workspace*. PintaOnWeb features a workspace storage system to organize and share workspaces. A workspace can be stored with a user-specified name on the server, which allows all other users to load this workspace with a single click. This allows a user to switch between specific views of the planning model easily, depending on the needs of different roles and use cases. In the future, it should be possible to store additional information in a workspace, such as the state of the model tree, to display all relevant planning information immediately. Optionally, workspaces can be marked as read-only, to prevent any changes by users.

*4.4 Properties View*

When a model object, such as an activity or resource, is selected either via the model tree or timeline view, a properties view for that object is displayed on the right side of the application (see D in Figure 3). This view contains all important information of the object, such as its name and parameters. Depending on the type of model object, additional properties are displayed, for example for an activity the list of timeline entries. This view, as the model tree view, can be resized freely in its width.

*4.5 Editing Mode*

PintaOnWeb runs in a read-only mode by default, which hides UI elements that are used for modification. When the editing mode is enabled by a click on the corresponding button in the toolbar, these modification UI elements are displayed and the model can be changed by the user. As an example, the properties view allows for modification of the displayed values, such as updating the parameter values or adding/deleting timeline entries. Entering editing mode is only allowed if the user has write access for the currently displayed mission. Additionally, PintaOnWeb can be set up in read-only mode, so that no user can modify anything using that PintaOnWeb instance.

## 5. Architecture

PintaOnWeb is based on state-of-the-art technical achievements in the fields of front-end and back-end programming. As a generic application, it will be used in a multitude of different missions, so it is expected to be extended with many more features in the future, depending on the needs of the missions. As a result, we expect the application to keep growing in scope and complexity, so it is of high importance to ensure that the software is maintainable and extendable at all times. Therefore we put our focus on picking scalable, maintainable technologies. This includes functional programming (Scala, Typescript) and a standardized, well defined interface between the frontend and backend (GraphQL). See Figure 7 for an overview of the main components of PintaOnWeb.

*5.1 GraphQL Interface*

The interface for transferring data between the frontend and backend is based on the GraphQL[b] query language. One main idea of GraphQL is that a client can specify exactly what information it needs, and then gets it without any unnecessary data. For a specific use case, for example the data to be displayed in a plot, all necessary data from the planning model can be requested using a single query. This keeps the transportation overhead to a minimum and allows for very fast responses. GraphQL queries are described as simple text files and responses are given in JSON format. The GraphQL specification is agnostic to the transport layer and programming languages of the endpoints, giving maximal flexibility for the implementation. We use WebSockets to transfer data between the PintaOnWeb frontend and backend, allowing for real-time updates of the displayed data through GraphQL subscriptions.

The GraphQL API can also be accessed separately over a GraphiQL[c] web-interface, a fully featured interactive GraphQL IDE which is bundled with PintaOnWeb. Here, GraphQL queries can be manually dispatched, and data is simply returned in text form. GraphQL also provides introspection features for intuitive exploration of the schema,

[b] https://graphql.org/
[c] https://github.com/graphql/graphiql

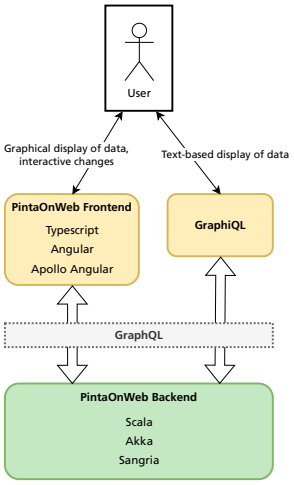


**Figure 7 PintaOnWeb architecture**
The frontend, running in a web browser, queries data using the GraphQL interface of the backend.

which makes it usable in a variety of other contexts, for example as part of Jupyter[d] notebooks for interactive data analysis.

### *5.2 Frontend*

The PintaOnWeb frontend is written in TypeScript[e] and based on the Angular framework[f]. It is designed as a single-page web application, with a focus on responsiveness and usability. Angular's opinionated approach for style and structure helps to keep a uniform implementation style. Angular provides most standard features needed for interactive web-applications, without reliance on third-party libraries. The Angular project also provides Angular Material, a kit of UI elements with a modern and unified look, based on Google's Material Design[g]. These UI components make it easy to create a nicely looking view quickly, but can also be be customized if needed. TypeScript builds on top of JavaScript but is strongly typed, which allows for comprehensive static code analysis and interface descriptions. Both Angular and TypeScript facilitate the development of separate components with clearly defined interfaces, with the goal of keeping changes in one part of the application from affecting other parts unless absolutely necessary.

The graphical display of the timeline as plots is implemented based on canvas drawing, which allows us to draw thousands of elements without noticeable negative impacts on responsiveness.

### *5.3 Backend*

The backend is written in Scala[h], with a focus on functional programming paradigms, with almost all information being stored as immutable data. We use Akka[i] libraries to implement the components of the system as loosely-coupled parts that use message-based communication. This approach allows for development of a scalable, resilient system, without having to worry about the low-level implementation details. Akka provides a set of libraries to address these issues, such as its actor system and data stream implementation.

## 6. Development and Deployment Processes

Since PintaOnWeb as a generic planning tool will be used in many different missions, we decided to follow an agile development approach from the beginning. This allows for fast reaction times to new requirements through its flexible development cycle, compared to a classical approach that tries to define all requirements and a development plan in the

[d] https://jupyter.org/
[e] https://www.typescriptlang.org/
[f] https://angular.io/
[g] https://m2.material.io/
[h] https://scala-lang.org/
[i] https://akka.io/

beginning. With the knowledge of the predecessor PINTA and ideas and suggestions for improvement, we created user stories, which we then transferred into requirements. Each requirement is tracked in a GitLab issue tracking system, and the issues get routinely reordered to keep the assigned priorities in line with the current goals. On this basis, we can incorporate new requirements, analyze their priority and then always work on the most urgent features within the current sprint cycle.

We organize our development process based on the Scrum approach, in sprints of four weeks. At the end of each sprint, we conduct a review meeting with the relevant stakeholders. At the start of each sprint, we hold a planning meeting to decide on our focus for the following weeks, and hold a retrospective meeting to gather feedback for improving our development process. The team of the project consists of four to six developers, a dedicated Scrum Master and two experts for the existing planning tools in an advisory role. The developers of our team are also active in other projects, so they cannot commit 100% of their time to PintaOnWeb. In the sprint planning, the development team estimates the total time available and sets the sprint goals accordingly. Because of the fluctuating developer time available, we decided against the use of a "traditional" daily meeting for exchanging progress reports and general discussion. Instead, we regularly adjust the number of meetings per week, depending on how much work can actually be done, which has been shifting between one and five meetings per week.

One core principle for agile development is that all developed features are intensively tested. This allows system defects to be detected and fixed early, reducing the maintenance effort. New features are implemented on feature branches, which get merged into the main branch after successful code review. A CI/CD pipeline running in a GitLab repository ensures that new features do not break any existing functionality. Additionally, our pipeline ensures that code style conventions are satisfied, checks for license compliance, and publishes artifacts to a binary repository. New versions are released by simply assigning a Git tag to a specific commit. The CI/CD pipeline then automatically publishes the state of that commit as the new version.

For deployment, PintaOnWeb is packed as a Docker[j] container. Using this container, a new version can be deployed by manually starting the newest version on a server. The preferred solution though is a GitLab runner that takes care of deployment, whenever a developer pushes changes to a dedicated deployment branch. The runner then pushes a new version to a given server where a Kubernetes[k] cluster is running, which updates the running images with the newly released ones. This process only takes a couple of minutes and removes almost every need for manual interaction to deploy a new version of PintaOnWeb.

## 7. Use in Missions and other Projects

### *7.1 EnMAP MPS*

The EnMAP mission uses a fully automated Reactive Planning system for its core mission planning system[12], and deploys a read-only version of PintaOnWeb to display the timeline to all relevant personel. During commissioning MPS also heavily relied on PintaOnWeb to validate the planned timeline and analyse the correctness of resource modelling. The following two examples illustrate how PintaOnWeb was used:

When planning datatakes the EnMAP planning system takes into account historical data about world-wide cloud coverage and recent forecasts for cloud coverage in the target area. The dynamic nature of cloud forecasts implies that the timeline changes frequently and on short-notice. To validate the correctness of these changes, MPS used the detailed planning data views in PintaOnWeb to inspect the success probability and forecast information of planned datatakes in order to verify that the expected datatakes got planned. With this information from the planning model at hand, MPS could trace the planning process and thus validate that the MPS system planned the correct datatakes with respect to predicted cloud coverage.

The planning system also models several on-board resources. The most notably of these is the amount of free space in mass memory of the data handling assembly of the spacecraft which puts a firm upper limit on the amount of successful datatakes between two data downlinks. Correct modeling of the mass memory consumed by datatakes is crucial for mission success, to neither lose data because of insufficient free space, nor plan less datatakes than would be possible. To validate this model, MPS used the the GraphQL API, to seamlessly integrate data from PintaOnWeb with internal and third-party tools for further analyis. In this case, MPS used this API to extract the modelled free mass memory and compare it against the actual free mass memory provided by the spacecrafts telemetry. The correlation of these two data sources allowed MPS to quickly validate that the modeled free mass memory corresponds to the actual

[j] https://www.docker.com/
[k] https://kubernetes.io

free memory.

*7.2 Galileo Future Planning Prototype*

The GNSS mission Galileo currently uses a constellation containing 26 satellites, with additional satellites scheduled to launch in the near future. For each satellite, a contact to one of the six available ground stations must be planned for routine housekeeping at least once per orbit, which is about every 14 hours. Each routine contact must also be accompanied by a backup contact, that is used in case the routine contact failed for some reason. To improve the existing planning process and prepare for the next phase of the mission, Galileo Second Generation (G2G), a mission planning system based on PintaOnWeb and Reactive Planning is envisioned and has been implemented as a prototype. The current planning system in operation for Galileo contains many manual processes for planning and validation, which often involves multiple tools and scripts in a single workflow. The new system consolidates all regular user interaction to happen via the PintaOnWeb web-interface, which helps to greatly simplify common workflows. Instead of having to handle multiple programs to plan, check for conflicts and export results, a planner only has a single point of interaction.

The Reactive Planning part of the system automatically imports all relevant data for the planning process from files. A constellation scheduling algorithm then calculates a valid timeline for a given planning horizon, by scheduling routine and backup contacts so that all relevant constraints are fulfilled. Via PintaOnWeb, the resulting timeline can be viewed using preconfigured workspaces, which display all contacts either grouped by ground station or satellite, see Figure 8 for an example. Using the editing features, contacts may be moved on the timeline or increased/reduced in duration. In the future, it would also be possible to semi-automatically schedule special contacts for non-routine activities.

The satellites which the planning system considers are stored as project parameters. Via the PintaOnWeb web-interface, a mission planning engineer can modify these parameters, for example to add a new satellite. This modification is then sent to the Reactive Planning system, which automatically updates the entire planning model to account for the new satellite. This includes the creation of related resources, such as the visibility for all registered ground stations, as well as activity templates for contacts, and planning constraints. After the planning model has been updated this way, the system is ready to ingest input specific to that satellite and will include the satellite in the next planning run. Since the Galileo mission will be ongoing for a long time and an extension of the constellation is already in preparation, this flexibility of the planning system to handle new satellites or ground stations is very important.

*7.3 InTAS*

Integrated Terminal and Antenna Scheduling (InTAS) is a new application developed by GSOC. One of the main goals is the coordination and scheduling of ground related activities, such as ground station contacts, or managing that the multi-mission control room personnel is not overwhelmed by parallel activities required by multiple missions at once.

As the basis for this tool, PintaOnWeb with Reactive Planning in the background has been chosen, since a mix of automated and manual planning is required. In the nominal case, the system should receive information about the supported satellite missions, such as visibilities and requested passes, via an automated interface. Based on

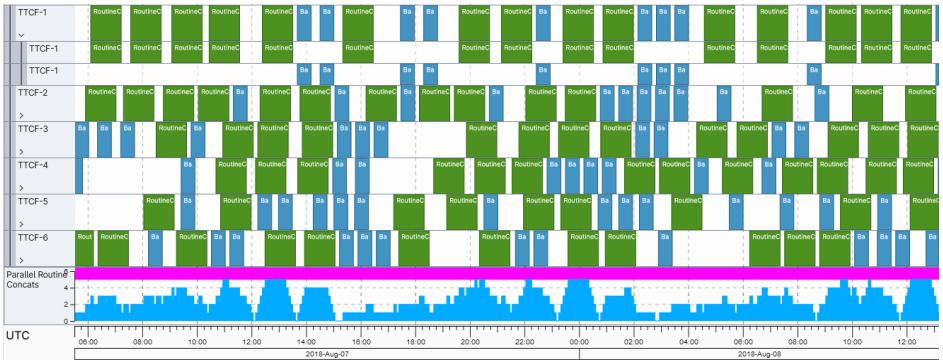


**Figure 8 Planned routine and backup contacts for the Galileo ground stations**
Shown are the routine contacts (green) and backup contacts (blue) for the ground stations (TTCF-*), as well as a resource modeling the number of parallel routine contacts.

that information, InTAS continuously generates a conflict-free plan considering all kinds of constraints, such as required contact frequency and types and minimum contact durations, groundstation preferences, communication (un-)availabilities and schedules the possible passes based on the priority of the mission. In the nominal case, PintaOnWeb will mainly serve as a "window" into the automatic planning black box where it can be checked if everything is planned as expected or as information source for operators when the next contact possibility exists. In off-nominal cases, PintaOnWeb can transform from a "simple timeline viewer" to a fully functional planning tool, where the user can modify the already existing plan to adapt to the current situation, for example schedule more off-nominal passes for a satellite to handle some kind of contingency. Since PintaOnWeb is web-based, a possible use case is that a flight director can schedule passes even from home, since InTAS would provide all information necessary for requesting such contacts and an easy to use interface to do so. This has the potential to remove the necessity for on-site presence, if the contingency is only minor and the present operators can solve it without the direct presence of the flight director.

### *7.4 On-call team planning*

In the near future it is also planned to replace the current on-call planning at GSOC, for which at the moment PINTA is used, with a PintaOnWeb solution. Currently, the plan is generated once every quarter by an algorithm and then manually modified based on new constraints of the affected personnel. This can be a tedious process at times since PINTA is a standalone application where only one user can modify the project at a given time. With the introduction of PintaOnWeb, true multi-user functionality will be supported and the on-call personnel can modify the plan themselves, such as changing shifts easily with the web application. The envisioned workflow would be something like one user wants to change his shift with someone else. This change is then requested via PintaOnWeb by changing the person to the given shift, but since another user will be affected by this change, this is only a requested change. The other user then would get a notification and then confirm or reject the proposed change and only if it is confirmed the overall plan is updated to show the change.

## 8. Conclusion and Outlook

We have presented the PintaOnWeb application, the essential interactive component of the next-generation generic planning tool suite at GSOC. It is in use in several projects already, and planned to be used for all future missions, replacing the old planning tool suite based on PINTA and TimOnWeb. It is well suited for a wide range of mission types, from mission planning systems that require fully automated planning to ones that include a lot of manual procedures. The software is designed to consist of a core of generic features, which can be complemented by mission-specific extensions to cover all possible use cases. Its development process is focused on agile practices, with constant priority readjustments based on mission needs.

While already usable in its current state, there are still additional functionalities in the queue to be implemented. For the near future, we aim for integrating user authentication based on the OpenID Connect standard, with granular user rights restrictions to support various user roles. Another important feature to enable real multi-user capabilities will be allowing users to work on a local copy of the planning model, and then commit all changes once they have finished their work. If such a change conflicts with other updates of the planning model, that have occurred in the meantime by parallel manual or automated modifications, we need to provide an interactive way to resolve these conflicts. These improvements will enable PintaOnWeb further to cover the whole range of future applications. In parallel, PintaOnWeb as well as Plains and Reactive Planning continuously profit from the day-to-day experience collected within the projects where they are already in use, with maintenance needs, user feedback and novel requirements leading to further enhancements. We are convinced that in the end, PintaOnWeb will become a powerful, robust, long-term workhorse for the missions operated at GSOC and beyond, and thus an adequate successor to our "old" flagship PINTA.

## Acknowledgments

We thank Vlad Filip and Jake Ashdown (Heavens Above) for their work in the PintaOnWeb development team, and Spencer Ziegler, Sabrina Moser and Ulrich Kling (DLR, Galileo Competence Center) for their collaboration.

## References

[1] Nibler, R., Hartung, J., Krenss, J., Fürbacher, A., Mrowka, F., and Brogl, S., *PINTA – one Tool to plan them all*, Springer International Publishing, Cham, 2022, pp. 345–379. doi:10.1007/978-3-030-94628-9_16, URL https://doi.org/10.1007/978-3-030-94628-9_16.

[2] Nibler, R., Mrowka, F., Wörle, M. T., Hartung, J., and Lenzen, C., "PINTA and TimOnWeb - (more than) generic user interfaces for various planning problems," *IWPSS 2017 - 10th International Workshop on Planning and Scheduling for Space*, 2017. URL https://elib.dlr.de/114095/.

[3] Spörl, A., Lenzen, C., Wörle, M. T., Hartung, J., Mrowka, F., Braun, A., and Wickler, M., "Mission Planning System for the TET-1 OnOrbitVerification Mission," *13th International Conference on Space Operations*, Pasadena, California, USA, 2014. doi:10.2514/6.2014-1758, URL https://doi.org/10.2514/6.2014-1758.

[4] Wörle, M. T., Spörl, A., Hartung, J., Lenzen, C., and Mrowka, F., "The Mission Planning System for the Firebird Spacecraft Constellation," *14th International Conference on Space Operations*, Daejeon, Republic of Korea, 2016. URL https://elib.dlr.de/105950/.

[5] Gärtner, S. A., Harder, N., Hartung, J., Hobsch, M., and Weigel, M., *A Mobile and Compact Control Center for Quick Decentral Satellite Access*, Springer International Publishing, Cham, 2022, pp. 419–446. doi:10.1007/978-3-030-94628-9_19, URL https://doi.org/10.1007/978-3-030-94628-9_19.

[6] Hartung, J., Nibler, R., Spörl, A., Lenzen, C., Wörle, M. T., and Peat, C., "GSOC SoE-Editor 2.0 - A Generic Sequence of Events Tool," *14th International Conference on Space Operations*, Daejeon, Republic of Korea, 2016. URL http://elib.dlr.de/105083/.

[7] Guanter, L., Kaufmann, H., Segl, K., Förster, S., Rogass, C., Chabrillat, S., Küster, T., Hollstein, A., Rossner, G., Chlebek, C., Straif, C., Fischer, S., Schrader, S., Storch, T., Heiden, U., Müller, A., Bachmann, M., Mühle, H., Müller, R., Habermeyer, M., Ohndorf, A., Hill, J., Buddenbaum, H., Hostert, P., van der Linden, S., Leitao, P. J., Rabe, A., Doerffer, R., Krasemann, H., Xi, H., Mauser, W., Hank, T., Locherer, M., Rast, M., Staenz, K., and Sang, B., "The EnMAP Spaceborne Imaging Spectroscopy Mission for Earth Observation," *Remote Sensing*, , No. 7, 2015, pp. 8830–8857. URL https://elib.dlr.de/97231/.

[8] Brogl, S., Manailescu, S., Gutierrez Vela, J., Ballweg, R., and Mrowka, F., "How Galileo Planning became automated," *16th International Conference on Space Operations*, 2021. URL https://elib.dlr.de/142237/.

[9] Wörle, M. T., Lenzen, C., Göttfert, T., Spörl, A., Grishechkin, B., Mrowka, F., and Wickler, M., "The Incremental Planning System – GSOC's Next Generation Mission Planning Framework," *13th International Conference on Space Operations*, Pasadena, California, USA, 2014. URL http://elib.dlr.de/89586/.

[10] Fruth, T., Lenzen, C., Gross, E., and Mrowka, F., "The EnMAP Mission Planning System," *15th International Conference on Space Operations*, American Institute of Aeronautics and Astronautics, Marseille, France, 2018. doi:10.2514/6.2018-2525, URL https://elib.dlr.de/120448/.

[11] Wörle, M. T., Lenzen, C., Wiesner, S., Prüfer, S., Petrak, A., and Müller, K., "Replacing the TDP-1 Mission Planning System – more than just another Technical Demonstration Project," *72th International Astronautical Congress 2021*, 2021. URL https://elib.dlr.de/145871/.

[12] Lenzen, C., Wörle, M. T., Prüfer, S., Krenss, J., Wiesner, S., and Fruth, T., "EnMAP MPS: Challenges, Enhancements and Evaluations of the Early Mission Phase," *17th International Conference on Space Operations*, Dubai, United Arab Emirates, 2023.