

Implementación de un Servicio de Gestión de Tareas para Satélites Pequeños

Autor: Víctor Bóveda Prado

Tutor: Fernando Antonio Aguado Agelet

Curso: 2019-2020

1. Introducción

Como consecuencia del avance tecnológico y de la liberalización del acceso al espacio, se ha dado comienzo a una nueva etapa en la que el sector espacial, ya que no es exclusivo de gobiernos sino que se está abriendo una nueva oportunidad de negocio para distintas empresas en la creación de servicios globales basados en el uso de Small-Sats.

La diversidad de aplicaciones basadas en Small-Sats (satélites pequeños) es inmensa, desde comunicaciones punto a punto, M2M¹, estudio de la Tierra mediante observación remota, aplicaciones climatológicas, seguimiento de cultivos, etc. .

La forma de operar en el entorno espacial requiere realizar una serie de procedimientos muy exhaustivos y estandarizados de todas las fases de desarrollo del satélite. Desde un buen aislamiento térmico de los componentes del satélite, su estructura, su resistencia a vibraciones, su programación interna, etc. .

Para la elaboración tanto del software como del hardware del satélite LUME-1, el equipo de la Agrupación Aeroespacial de la Universidad de Vigo se basó en el estándar ECSS (European Cooperation for Space Standardization), que se define como una iniciativa establecida para desarrollar un conjunto coherente y único de estándares fáciles de usar para su empleo en todas las actividades espaciales europeas. En este estándar se recogen todos los puntos clave de la elaboración del satélite. La misión del satélite definirá los requisitos de usuario y técnicos que permitirán precisar la arquitectura del sistema del proyecto espacial.

El proyecto FireRS (acrónimo de wildFIRE Remote Sensing) tiene como objetivo proporcionar a las agencias de emergencia y/o centros de coordinación una herramienta innovadora para la detección y la gestión del incendio mediante nuevas tecnologías. La nueva plataforma creada en FireRS ofrecerá información casi en tiempo real, posicionado GPS, perímetro del fuego, imágenes infrarrojo, predicción de la propagación, protocolos de actuación, etc. El conjunto se compone de sensores de tierra, un picosatélite dedicado, una estación de tierra local y remota así como un centro de control de la misión. <http://www.fire-rs.com/es/>

¹M2M: Este concepto hace referencia al intercambio de información o comunicación entre dos máquinas.

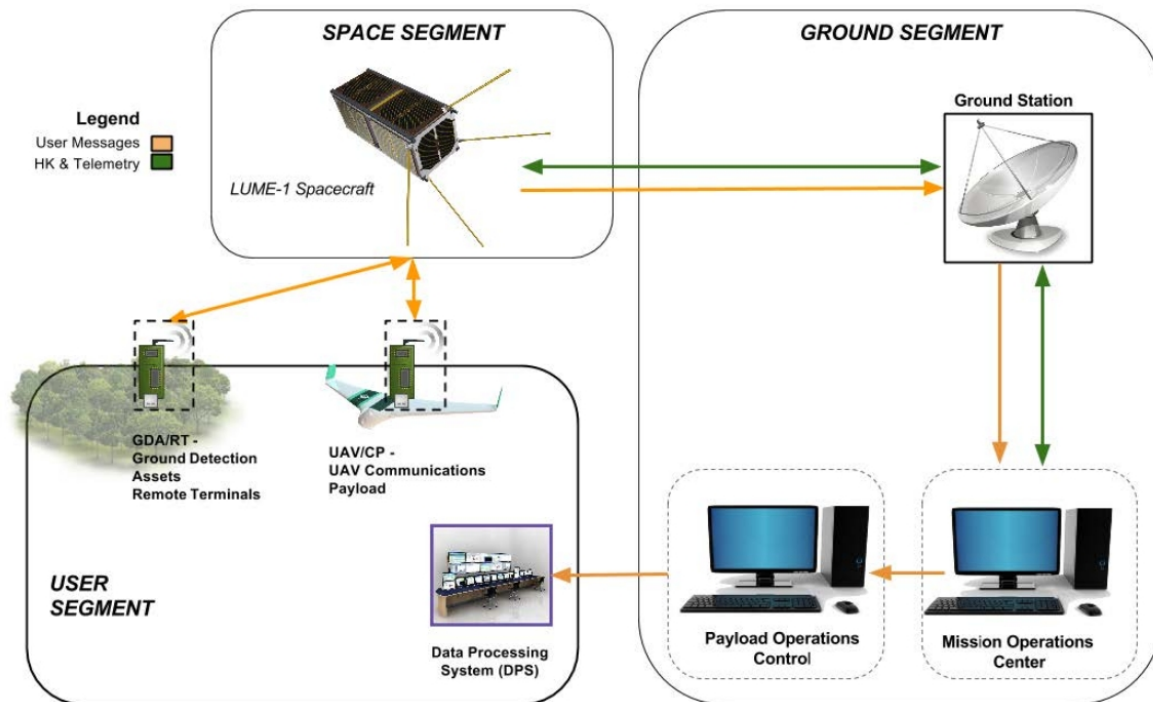


Figura 1: Arquitectura Sistema FireRS "<http://www.fire-rs.com/es/>".

Debido a las limitaciones físicas de tener un solo satélite en una órbita LEO², la ventana de comunicación se reduce a cuatro o cinco contactos diarios con el mismo. Además, en el caso de LUME-1 sólo se dispone de una estación terrena para establecer comunicación con el satélite por lo que la implementación de un programador de tareas para aprovechar el tiempo en el cual no se dispone contacto directo con él es de vital importancia.

²Low Earth Orbit: órbita baja terrestre.

2. Objetivos

El objetivo de este proyecto es la creación de una herramienta para la gestión de tareas automatizadas de un satélite. Los requisitos principales de este proyecto son:

1. Desarrollo Front-End de la aplicación.
2. Desarrollo Back-End de la aplicación.



Figura 2: Requisitos principales.

La idea principal de esta aplicación consiste en la creación de una librería de instrucciones y de un entorno gráfico para su gestión. Siguiendo para ello, con la línea establecida en el resto de aplicaciones desarrolladas por la Agrupación Aeroespacial de la Universidad de Vigo. La finalidad de este proyecto consiste en alcanzar la semi-automatización de procesos recurrentes en el plan de operaciones del satélite, ayudando de esta forma a agilizar las tareas de los operadores de operaciones y de los operadores de carga útil.

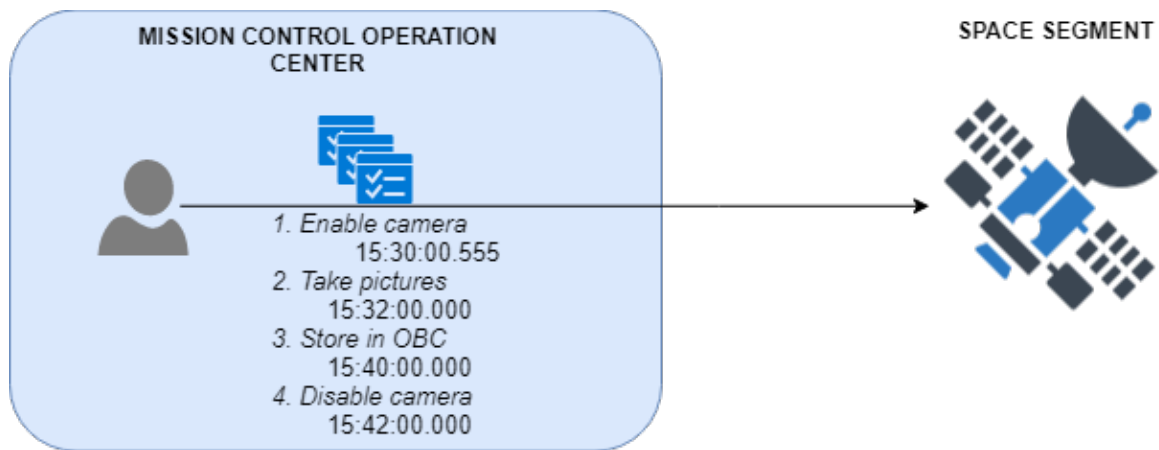


Figura 3: Ejemplo de uso.

En la imagen previa se ve un ejemplo del objetivo de la aplicación, en el cual el operario envía un conjunto de instrucciones con sus respectivas horas de ejecución. Si la operación a realizar tiene carácter periódico será necesario para su reutilización agrupar las instrucciones que desarrollan una operación bajo librerías. De esta forma, una vez configuradas las librerías es posible emplearlas repetidamente desde la aplicación con solo modificar el tiempo en el que se desea que se ejecuten.

2.1. Necesidad de un servicio de programación de tareas

La necesidad de un servicio de programación de tareas, surge de las limitaciones referentes a establecer una comunicación directa con el satélite. Esto es debido a la órbita que describe el satélite y a la distribución geográfica de las estaciones base con las que se opera.

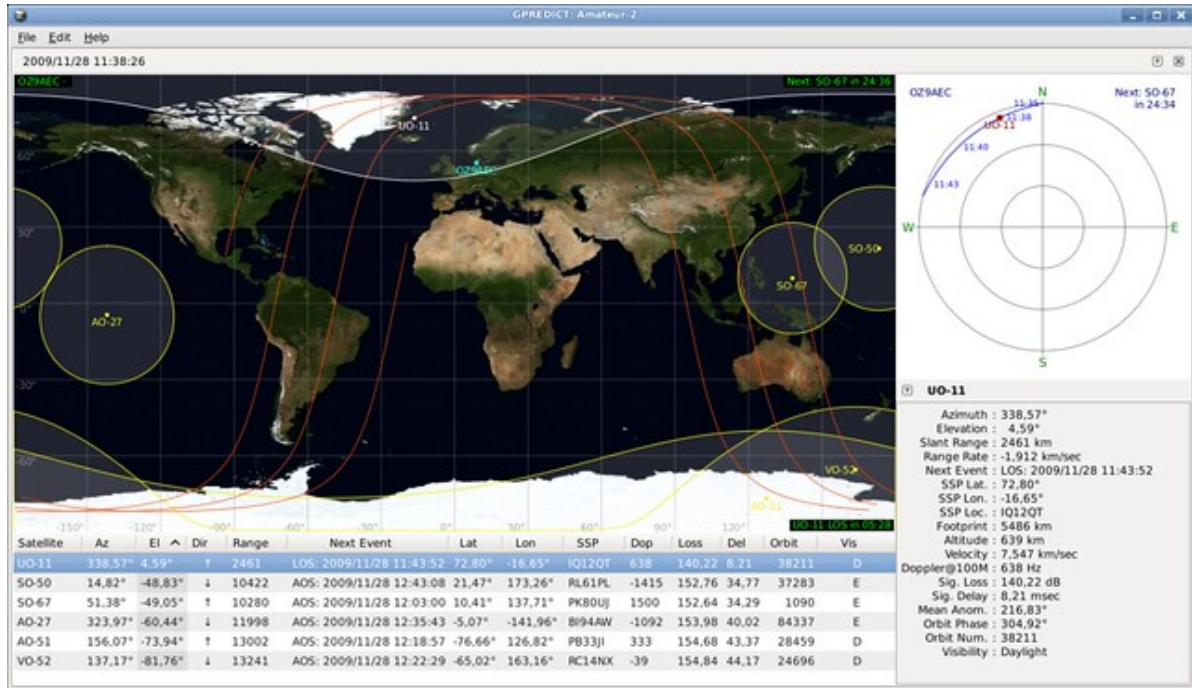


Figura 4: Ejemplo de pase visionado en la herramienta de predicción GPredict.

Para la creación del servicio de programación de tareas, la Agrupación Aeroespacial de la Universidad de Vigo, se basó en el estándar ECSS, el cual entre otras cosas, define la lógica del programador de tareas que está integrado en el OBSW (On Board Software).

Dentro de este estándar se define el estándar PUS (Packet Utilisation Standard), el cual fija la estructura y la funcionalidad de los telecomandos y telemetría, necesarios para abarcar los servicios imprescindibles que debe realizar el satélite, entre ellas el servicio de programación de tareas “Scheduler”. En el Anexo F se muestra el funcionamiento de este servicio, figura 17.

El gestor de tareas es el servicio complementario al programador de tareas que está integrado en el OBSW del satélite. Este servicio está implementado en la capa de software correspondiente al segmento de tierra. Esta capa de software se denomina Ground Station Software (GSSW).

Del GSSW cabe destacar que está formado por distintas capas de software que dan soporte a todos los sistemas necesarios para establecer una comunicación fiable con el segmento espacial: un terminal CSP, un GS proxy, comunicación por WebSockets, API Web, etc. . Para automatizar el despliegue de estos elementos se ha empleado el sistema de contenedores de Docker, el cual es un sistema que automatiza el despliegue de aplicaciones. Dentro de uno de estos contenedores se encuentra la aplicación GS-Web, el cual proporciona una interfaz web para el control del satélite basada en Django, que es un framework de desarrollo de aplicaciones web basado en el lenguaje Python. Para más información véase el Anexo H.

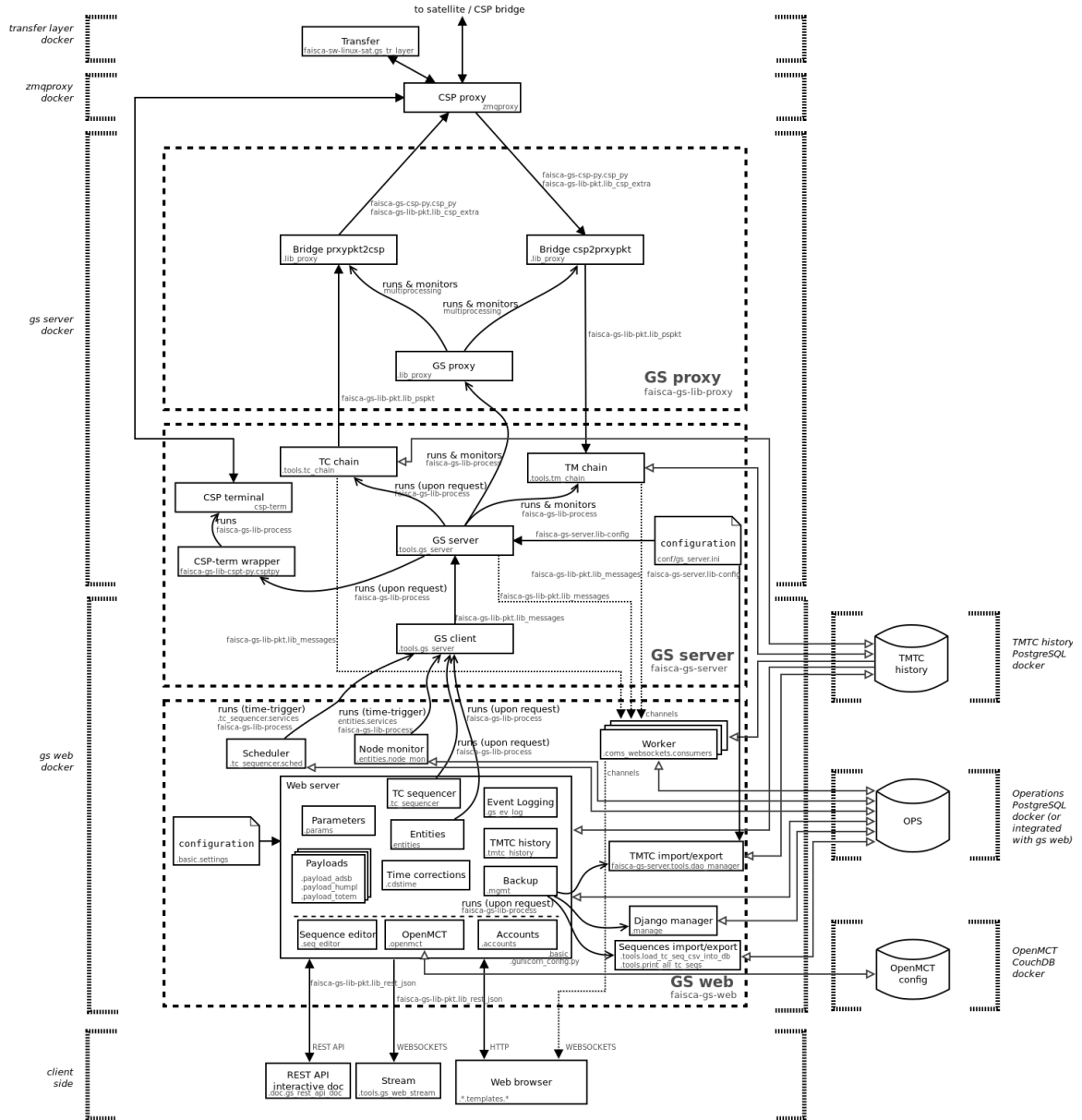


Figura 5: Arquitectura global del GSSW.

2.2. Funcionamiento inicial del servicio de gestión de tareas.

Para explicar este servicio es necesario definir los siguientes términos:

- **Packet Utilization Standard (PUS):** esta norma definida en el ECSS aborda la utilización de paquetes de telecomando y paquetes de telemetría para fines de monitoreo y control remoto de subsistemas y cargas útiles.
- **Scheduler:** se denomina al servicio PUS integrado en el OBSW del satélite, encargado de ejecutar las tareas programadas cuando corresponda.
- **Request/Verification:** es un servicio PUS que se encarga de informar al GSSW sobre la evolución del estado del procesamiento de un telecomando en tiempo real, si así se requiere.
- **Verificación:** conjunto de normas que han de cumplir los distintos telecomandos para que su interpretación por el OBSW sea correcta. Es decir, verificación de campos requeridos según el telecomando, verificación de integridad ante los campos spacecraft (identificador del satélite), APID³(identificador del subsistema al que va dirigido), type (servicio al que va dirigido dentro del subsistema), sub-type (subsistema dentro del servicio citado previamente).
- **Telecomando/Instrucción:** space paket (PUS) que encapsula la información que se envía al satélite.
- **Secuencia:** conjunto agrupado de telecomandos que se envían al satélite.

En este servicio del GS-Web es posible operar de dos formas distintas:

1. La primera consiste en subir un fichero a la aplicación web en formato *CSV*.
2. La segunda consiste en la edición *in situ* de una programación de secuencias mediante un formulario.

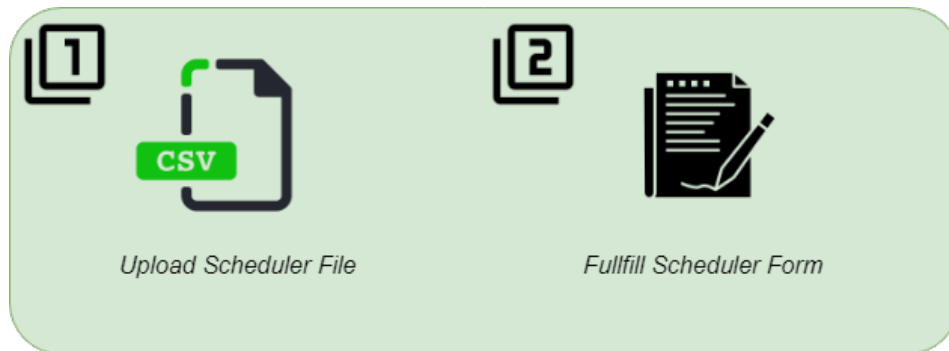


Figura 6: Formas de operar en la sección Upload Schedule.

En ambos casos, irán los campos de la instrucción “TC[11,4] insert activities into the time-based schedule”, la cual se encarga de insertar la siguiente lista de instrucciones en el “Scheduler” del satélite y al APID al que va dirigida la instrucción. A continuación, aparecerán una lista con la hora de ejecución de cada instrucción, seguida de su nombre, cada instrucción deberá estar cargada previamente en la base de datos de nuestro GSSW.

³APID: identificador del subsistema al que va dirigido

Cada instrucción va seguida de un campo groupId, el cuál es un campo que sirve para agrupar las distintas instrucciones que se cargan al satélite por grupos, lo cual habilita una funcionalidad extra que sirve para el filtrado y eliminación de instrucciones ya enviadas al “Scheduler” del satélite. Además de esto, siempre que se crea una programación de instrucciones, se añade por defecto la instrucción “TC[11,17] summary-report all time-based scheduled activities”, la cual se encarga de que el satélite envíe un informe de las instrucciones cargadas en su servicio “Scheduler”.

Ambos modos de operación han de pasar por un proceso de verificación de integridad, cuyo resultado se observa en el campo “Results” de la imagen 18 del Anexo G.

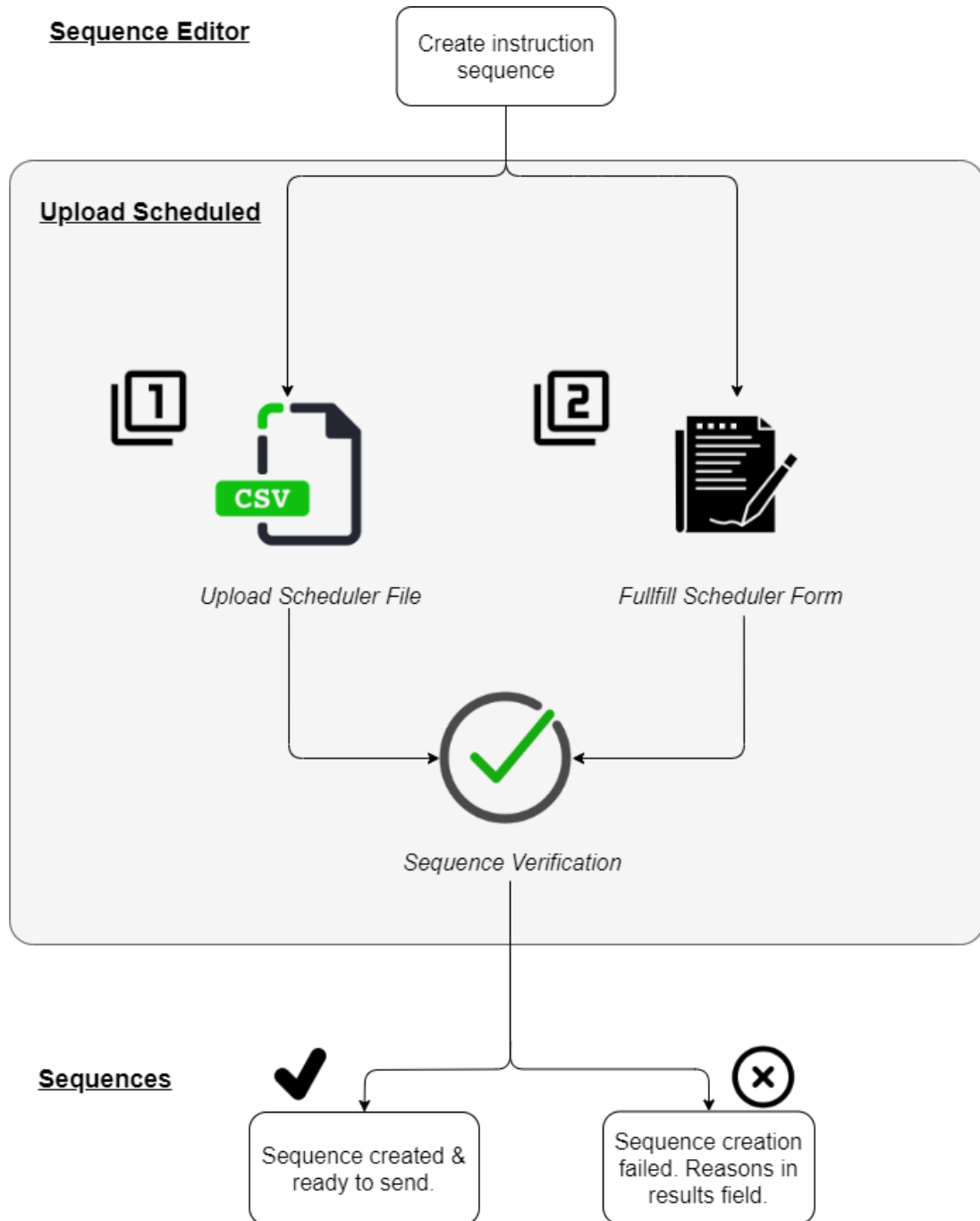


Figura 7: Diagrama funcionamiento sección Upload Schedule.

Como se puede observar, el servicio de gestión de tareas no permite al operador reutilizar de una manera cómoda secuencias que ya se han empleado. Ya que para ello, por ejemplo, se debería volver a cargar el mismo fichero *CSV* modificando la fecha y hora de ejecución de cada instrucción, lo cual es bastante ineficiente, cuando la cantidad de instrucciones programadas para cargar al “Scheduler” es grande. Otro punto a destacar es que sería necesario mantener un mantenimiento periódico de la sección “Sequences”, ya que estaríamos duplicando secuencias de los mismos telecomandos donde lo único que cambiaría sería el momento de ejecución de los mismos. Ante la necesidad de aumentar la eficiencia de este servicio, se ha planteado la solución descrita en el siguiente apartado.

2.3. Funcionamiento de la mejora para el servicio de gestión de tareas.

El funcionamiento de la mejora del servicio, que es sujeto de esta memoria, se basa en una reestructuración en la manera de conformar el servicio de gestión de tareas. Para ello, se ha implementado una nueva funcionalidad dentro de la aplicación GS-Web. Por consiguiente, es necesario entender la nueva nomenclatura utilizada en cada sección y entender la relación que hay entre estas secciones.

- **Actividades (Activities):** en esta sección se guarda registro de las distintas instrucciones que se envían al “Scheduler” del satélite siguiendo el orden en el que está programada su ejecución.
- **Conjunto de operaciones (Batch):** como su nombre indica en este nivel se agrupan las distintas operaciones programadas.
- **Operaciones (Operations):** cada operación hace referencia a una librería en concreto, además cada operación posee una hora de ejecución asociada.
- **Librería de operaciones (Operation library):** es un conjunto de instrucciones/telecomandos. En el modelo, está definido por “OpType”.
- **Instrucciones/Telecomandos:** space paket (PUS) que encapsula la información que se envía al satélite. En el modelo, está definido por “OpInstruction”.

Ahora que se conoce la terminología con la que se conforma esta funcionalidad, es necesario conocer la relación lógica que hay detrás. Para más información sobre los modelos y relaciones véase el Anexo D.

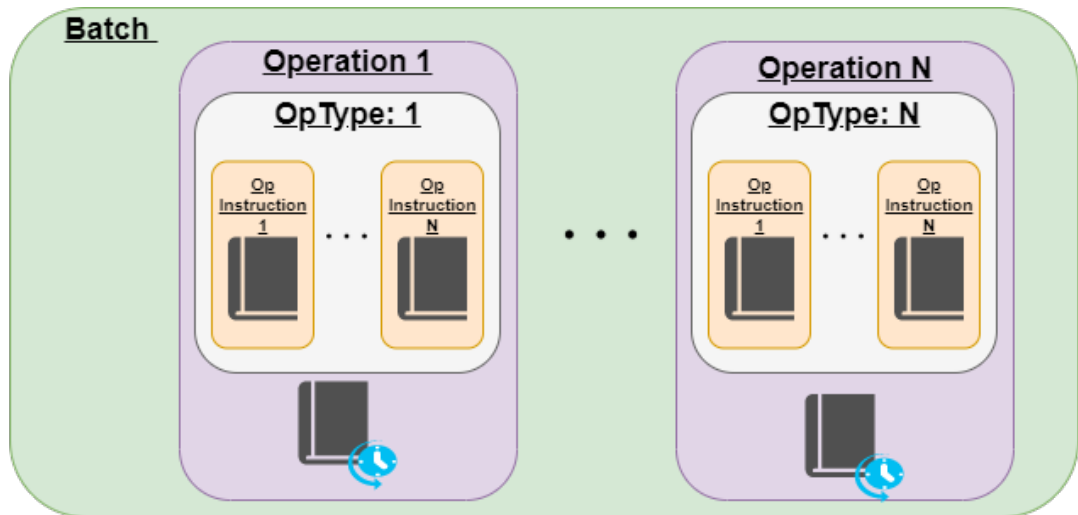


Figura 8: Diagrama de las relaciones de los modelos de un conjunto de operaciones “Batch”.

Como se puede apreciar por la imagen previa, las siguientes relaciones permiten sacar mucho más partido a la reutilización de datos para tareas de programación periódicas o puntuales del satélite. Observando el diagrama, se puede apreciar la necesidad de que se necesitarán como mínimo cinco ventanas de visualización para las operaciones de CRUD de los distintos modelos establecidos en la aplicación (para más información véase el Anexo B). El contenido de la información referente a cada sección está organizado en un sistema de tablas, el cual se detallará en mayor medida en el próximo punto. El funcionamiento de la aplicación se basa en:

1. Creación de las librerías de operaciones, las cuales tendrán asociadas un conjunto de instrucciones desde la sección “Operation Library”.
2. Creación de los conjuntos de operaciones (Batch), que inicialmente estarán vacíos hasta que dentro de cada uno se creen una o más operaciones, las cuales estarán relacionadas directamente con una librería de operación.
3. Desde la sección de conjunto de operaciones “Batch Editor”, y previamente a su envío, si así se desea, se deberá pasar cada conjunto de operaciones por un proceso de verificación, el cual es idéntico al ya comentado en el punto 2.1. Aquí pueden suceder dos situaciones:
 - a) En caso de pasar este proceso se podrá enviar este conjunto de operaciones, para lo que se habilitará un botón de envío que inicialmente estará deshabilitado.
 - b) En caso de no pasarlo, la interfaz web mostrará un modal, con la información referente a la razón por la que no ha pasado la verificación.
4. Si se produce el envío, el servicio registrará el contenido de los telecomandos enviados desde la aplicación, en la sección “Activities Register”. Para poder hacer un seguimiento más acotado de las instrucciones enviadas desde esta aplicación. Esto tiene sentido, ya que en otra aplicación del GS-Web hay un visor del histórico de todos los telecomandos enviados al satélite.
5. Si se produce el envío, a su vez se enviará la instrucción “TC[11,17] summary-report all time-based scheduled activities”. Para comprobar lo antes posible si el satélite ha registrado las instrucciones que se acaban de enviar en su “Scheduler”.

A continuación, se muestra un diagrama de uso de la aplicación resumiendo los casos previamente citados.

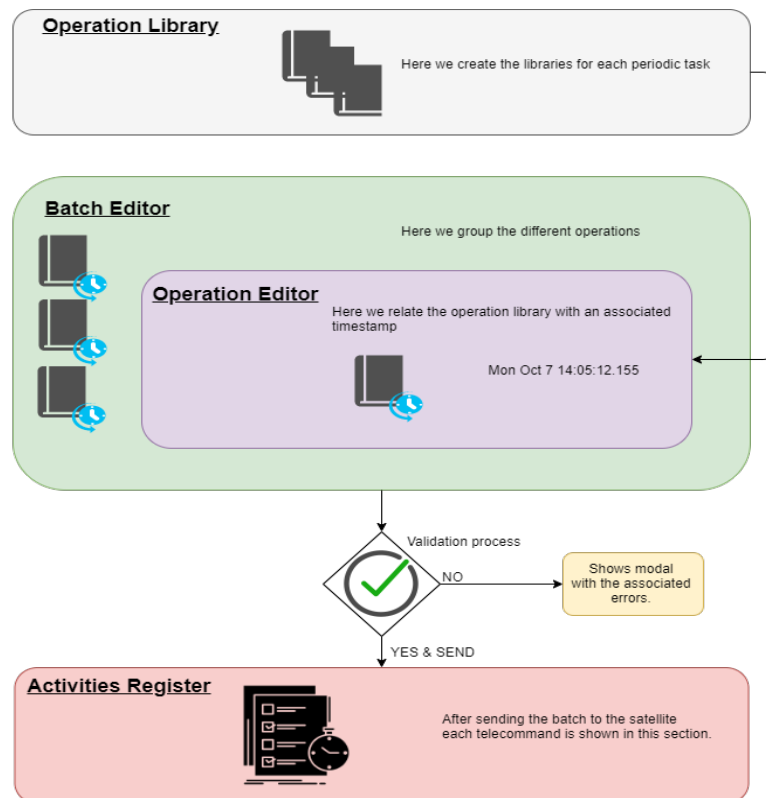


Figura 9: Diagrama de funcionamiento.

Esta aplicación tiene la particularidad de que permite reutilizar los conjuntos de operaciones directamente desde dos secciones. Pero dependiendo de la sección sus efectos sobre el resultado final varían:

- Desde la sección de Batch Editor: si se modifica la hora a la que se quiere programar su ejecución, esta modificará la hora de la primera operación contenida dentro de este conjunto de operaciones, pero con la peculiaridad de que las sucesivas operaciones mantendrán la misma diferencia temporal entre si.
- Desde la sección de Operation: si se modifica la hora de una operación en particular, sólo se modificará la hora de esa operación. Por lo que el único cambio visual en la tabla, a parte del cambio de hora será una reestructuración en el orden del visionado de las filas de la tabla, ya que estas se ordenan por defecto, en orden cronológico. Cabe destacar que para acceder a esta sección siempre es necesario pasar por la sección de Batch Editor.

Otra característica del servicio implementado es que cuando se añaden instrucciones a cualquiera librería de operaciones se podrá fijar un “time gap”, el cual, nos permite aplicar un intervalo temporal vacío entre instrucciones. Esto surge de la necesidad, de que dependiendo del tipo instrucción que se envíe, se podría llegar a necesitar el resultado de una instrucción previa, por lo que esto garantiza de una manera rápida y cómoda el poder cumplir este requerimiento. También, dentro de la sección de la librería de operaciones se puede observar el intervalo de tiempo total, este campo muestra la duración total que hay entre la primera y la última instrucción contenidas dentro de cada librería.

Entre las distintas secciones donde se conforman las instrucciones que se pretenden enviar (Operation Library, Batch Editor y Operation) hay posibilidad de verificar individualmente o conjuntamente, el conjunto de telecomandos que en ellas se hayan. Este proceso tendrá su reflejo en cada una de las otras secciones, gracias al uso de WebSockets. Para más información véase el Anexo H.

A continuación se muestran dos diagramas, en el primero se observa la interrelación que hay entre las distintas ventanas en función de donde se decida validar. En este caso observamos que tanto en la sección de operación como en la de librería de operación la verificación de una implica la de la otra y viceversa, al contrario que en la verificación desde la sección “Batch”. Esto sucede así debido a que la sección “Batch” lleva asociada una instrucción “TC[11,4] insert activities into the time-based schedule”, que es la que se encarga de introducir los telecomandos, bajo el conjunto de operaciones, en el “Scheduler” del satélite. Como ciertos campos de esta instrucción se pueden reconfigurar desde la sección “Batch” es lógico que si un usuario se encuentra en un nivel inferior esta no se valide.

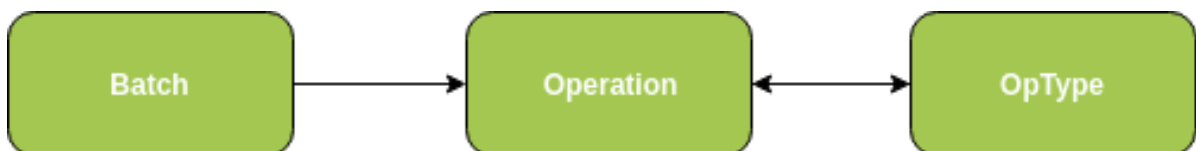


Figura 10: Interrelación de secciones al validar.

En el segundo diagrama se puede apreciar la interrelación que hay a la hora de modificar los datos de las tres secciones previamente citadas, ya que en la implementación realizada cualquier cambio en el contenido de cualquier entrada implica que esta y alguna otra sección, se invaliden por motivos de seguridad.

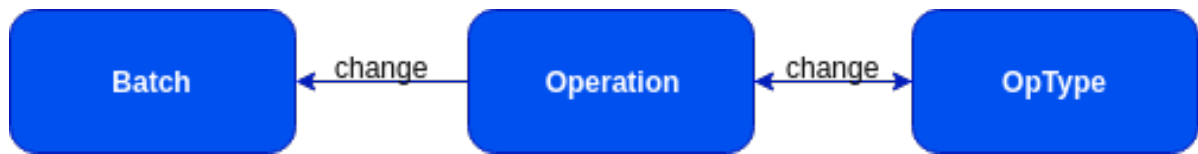


Figura 11: Interrelación en la verificación al modificar secciones.

De esta interrelación a la hora de validar es interesante explicar lo que realiza la sección “Batch”, ya que esta es la que engloba las operaciones y librerías de operaciones y la que se encarga de conformar las tareas que se envíen al satélite. Como se ha explicado previamente, la sección “Batch” contiene internamente la instrucción “TC[11,4]”. Para ello y de ahí su independencia de verificación con el resto de secciones, esta sección se puede reconfigurar si así se precisa, concretamente los distintos flags del servicio de “Request/Verification” del telecomando anterior.

Ahora que ya se comprende el funcionamiento interno de las secciones que nos ayudan a conformar una serie de tareas programadas, falta explicar en detenimiento que sucede cuando estas, finalmente se envían.

Como se ha explicado anteriormente, una vez se envían las instrucciones contenidas dentro del conjunto de operaciones, estas son almacenadas dentro de una sección denominada “Activities Register”.

En esta sección, se guarda un histórico de los telecomandos enviados desde nuestra aplicación de gestión de tareas, conjuntamente, con otros metadatos interesantes para el operador, a esto lo denominamos Actividad. Cada una de estas actividades consta de tres campos relevantes:

- `gs_db_id`: este campo es el identificador, del telecomando enviado, almacenado en la base de datos del GSSW.
- `rv_Completion`: este campo se encarga de informarnos si el satélite ya completo la ejecución de esta actividad. Forma parte del servicio de “Request Verification”.
- `packet_name`: este campo es un identificador único del nombre del space packet generado, esto ayuda a relacionar los paquetes enviados y recibidos de la estación de tierra y el satélite.

Desde esta sección y gracias al campo `gs_db_id` es posible acceder a un visor del paquete enviado aprovechando una de las funcionalidades ya habilitadas en el histórico de telecomandos y telemetría, esta funcionalidad se denomina “Packet Viewer”. Véase figura 19 del Anexo G.

Otra de las funcionalidades desarrolladas, ha sido la de añadir parámetros de configuración directamente desde la sección de “Operation” los cuales también son configurables desde esta sección. Esta funcionalidad fue solicitada inicialmente cuando se planteó el proyecto pero para esta versión no se ha decidido definirlas en mayor profundidad. De todas formas, el desarrollo de la GUI⁴ y de las operaciones de CRUD relacionadas con este campo están operativas a expensas de querer seguir desarrollando esta funcionalidad.

Cabe destacar que la funcionalidad principal de esta aplicación, una vez ya está creado el conjunto de operaciones que se desean insertar dentro del “Scheduler”, es en la elaboración automática de la instrucción “TC[11,4] insert activities into the time-based schedule”, la cual posee tres campos bastante tediosos de rellenar en caso de que el operario los deseara realizar a mano, ya que son tres arrays que poseen la misma longitud y que están relacionados por su índice. Estos son:

⁴Graphical User Interface: sirve para aportar al usuario un conjunto de imágenes y objetos gráficos para representar la información y acciones disponibles en la interfaz

1. releaseDay: día de ejecución desde un EPOCH⁵ dado.
2. releaseMsDay: milisegundo de ejecución desde un día dado.
3. instruction: nombre de la instrucción a ejectutar, de ahí que el campo “name” en todas las aplicaciones del GSSW sea un campo único. Derivado de la rigurosidad de rellenar estos tres arrays, ha surgido la necesidad de realizar aplicaciones que hagan que esto sea transparente para el operador.

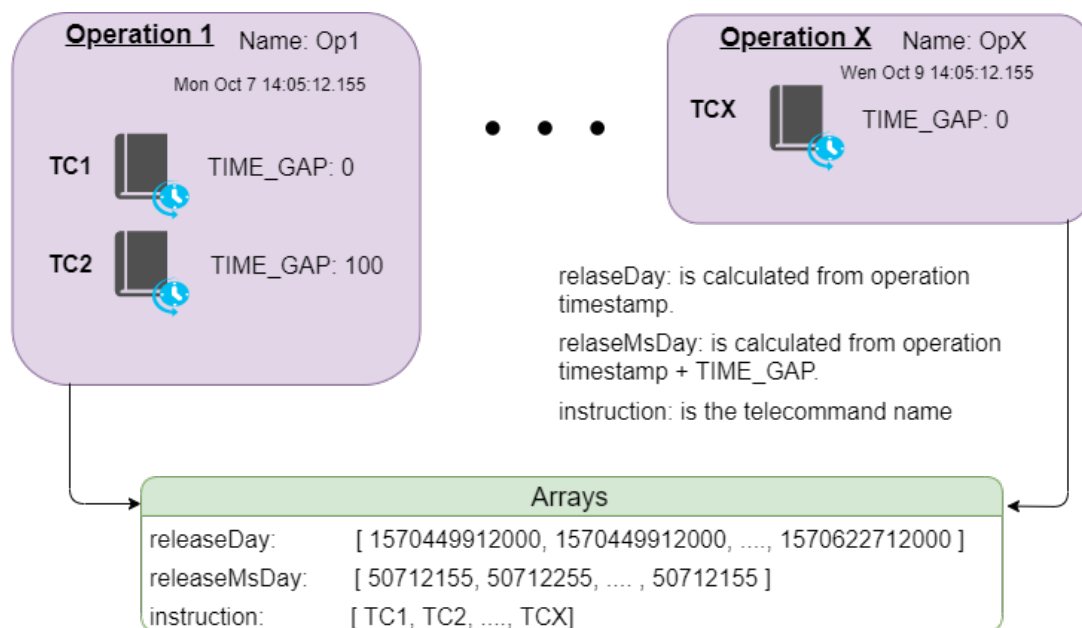


Figura 12: Creación de los arrays.

En el caso de esta mejora, la conformación de los tres campos viene dada por el orden cronológico de las instrucciones albergadas en cada librería contenida dentro de cada operación, la cual posee este campo temporal. Aún así dentro de cada instrucción recuerden que existe un campo “time_gap” el cual se tendrá en cuenta a la hora de rellenar el campo “releaseMsDay”. De esta manera, cada una de las secciones albergadas en esta mejora del servicio de gestión de tareas, permite que todo esto sea realizado de una manera totalmente transparente y en un formato, en el que el usuario esté más familiarizado.

2.4. Diseño de la mejora para el servicio de gestión de tareas.

Para el diseño de la mejora en el servicio se han utilizado distintas herramientas, las cuales se detallan a continuación:

- Para la visualización de los campos requeridos y para realizar las operaciones de CRUD en cada una de las secciones de la aplicación, se ha empleado una herramienta llamada JsGrid, la cual es un plugin ligero basado en jQuery que permite la gestión de datos en tablas desde el lado del cliente. Para más información véase el Anexo H. Con este plugin se han implementado las peticiones de create/read/update/delete del lado del cliente además de las opciones de filtrado, clasificación y paginado de las columnas y filas de cada tabla.

⁵EPOCH: es una fecha y hora desde la cual un computador mide la hora del sistema.

Después de lanzar la petición de cualquiera de las operaciones, anteriormente citadas, estas son enviadas como un formulario, empleando en función del tipo de operación, los métodos GET o POST, hacia una URL previamente definida, la cual invoca a un método que está implementado en el directorio de la aplicación, en el archivo “views.py”.

De esta forma en cada una de estas funciones se fijan las normas necesarias para realizar las operaciones de CRUD, de filtrado, clasificación y paginado necesarias para gestionar la entrada y salida de datos desde el back del framework hacia el front y viceversa. Todas estas operaciones se realizan de una manera sencilla para el programador gracias a la utilización de una aplicación basada en Django llamada, Django Simple REST.

La clasificación se ha realizado gracias a los métodos nativos de la ORM de Django, tales como “order_by(sortField)”. El filtrado también se ha realizado mediante estos métodos nativos, tales como “filter(field_name__typeofsimilarity = value)”. El paginado se ha implementado gracias a una funcionalidad que se encuentra en el “core” de Django llamada “Paginator”. Para más información véanse los Anexos [B](#) y [H](#).

- Para la visualización del campo de verificación entre cada una de las secciones se ha empleado la utilización de una herramienta llamada WebSockHop. Para más información véase el Anexo [H](#). WebSockHop es una biblioteca conveniente para clientes de Web-Socket que proporciona reconexión automática, ping periódicos e interacciones de solicitud/respuesta.
- Para la configuración del campo “Scheduled” en las secciones de “Operation” y “Batch”. Se ha empleado un plugin llamado Air Datepicker. Para más información véase el Anexo [H](#). Este es un plugin ligero basado en jQuery, construido en ES5 y con css Flexbox. Posee soporte para cualquier navegador moderno y es fácilmente configurable.
De hecho el empleado en esta aplicación ha sido configurado, ya que solo disponía de precisión temporal hasta el nivel de segundos, por lo que se tuvo que adaptar para poder operar a nivel de milisegundos, lo cual era necesario para operar plenamente con el “Scheduler”, ya que este posee esta precisión.
Cabe destacar que como formato temporal en todos los servicios de las aplicaciones del GS-Web, se utiliza un formato que varía de la norma ISO 8601, expresando la fecha y hora siempre correspondiente a GMT +0. Ej: 2019-06-11 18:30:08.526 .
- Para la edición de las distintas entradas de las tablas, se ha recurrido al uso de modales pertenecientes a Bootstrap, para más información véase el Anexo [H](#). Para poder así hacer una edición más cómoda de todos los campos necesarios para cada telecomando o para cada elemento, sin necesidad de alterar el formato de la tabla. Teniendo así una visualización más práctica de los contenidos requeridos en cada sección.
- Para la visualización de las distintas secciones se ha utilizado un fichero css llamado “custom_styles.css” mediante el cual se consigue que la aplicación sea “responsive” y esté operativa para resoluciones mayores o iguales a 480px.

3. Resultados

3.1. Cronología del desarrollo

A la hora de desglosar el desarrollo de esta aplicación se definieron cuatro fases con su duración aproximada. Para hacer una estimación del tiempo y esfuerzo que puede llegar a requerir realizar una aplicación de estas características, en un software que lleva en continuo desarrollo desde hace doce años.

<u>Fases</u>	<u>Cronología</u>	<u>Descripción</u>
Adaptación	09-07-18 hasta 01-09-18	<ul style="list-style-type: none">■ Adaptación al entorno de trabajo y sus herramientas.■ Planificación inicial del proyecto.
Desarrollo I	02-09-08 hasta 01-11-18	<ul style="list-style-type: none">■ Comienzo del desarrollo de la GUI de las primeras secciones hasta la sección “Activity Register”.■ Planificación de los modelos de datos.■ Revisión de la planificación inicial para adaptar a las nuevas necesidades derivadas del desarrollo.
Desarrollo II	01-11-18 hasta 01-03-19	<ul style="list-style-type: none">■ Desarrollo del proceso de verificación de cada sección.■ Interrelación de la GUI mediante WebSockets.■ Conclusión de los modales correspondientes a cada sección.■ Conclusión de la ventana “Activity Register” al enlazarla con los campos definidos en la sección 2.3.
Corrección y validación	01-03-19 hasta 24-04-19	<ul style="list-style-type: none">■ Depuración del código.■ Realización de tests sobre la GUI.

3.2. Comprobaciones y Testing

En esta sección se describen los trabajos que se realizaron para llevar a cabo las distintas comprobaciones y tests de la aplicación desarrollada. En primer lugar, debido a la extensión y a los distintos módulos alcanzados desde esta aplicación no se vio necesidad de realizar ningún test unitario de la aplicación, ya que las secciones referenciadas al “back” de la aplicación, las más sensibles como puede ser el proceso de verificación ya poseen varios tests que verifican la integridad de esta sección. Las distintas comprobaciones fueron realizadas durante el propio desarrollo como puede ser la verificación de los distintos elementos de la GUI, el cual no se puede testear a no ser que sea visualmente.

Las operaciones de comprobación de la aplicación desarrollada frente a un satélite fueron testeadas mediante el empleo de una imagen del OBSW montada en un contenedor de Docker. Por lo que todas las verificaciones del servicio de Request/Verification como los resúmenes de instrucciones albergadas en el “Scheduler” del satélite han sido probadas y testeadas en un entorno de simulación que asemeja a la realidad, comprobando de esta manera todos los posibles casos que puede haber dentro de un uso real de esta funcionalidad.

Coverage report: 33%						
Module ↓	statements	missing	excluded	branches	partial	coverage
/data/gssw/fg_csp/csp_py.py	98	40	0	2	0	58%
/data/gssw/fg_csptpy/const_cspt_py.py	39	0	0	0	0	100%
/data/gssw/fg_csptpy/lib_cspt_py.py	144	118	0	74	2	13%
/data/gssw/fg_csptpy/tools/csptpy.py	311	287	0	122	2	6%
/data/gssw/fg_db/lib_db_pg.py	529	287	0	222	18	40%
/data/gssw/fg_gs/const_config.py	148	0	0	0	0	100%
/data/gssw/fg_gs/const_dao_manager.py	11	0	0	0	0	100%
/data/gssw/fg_gs/const_gs.py	35	0	0	0	0	100%
/data/gssw/fg_gs/const_tc_chain.py	44	0	0	0	0	100%
/data/gssw/fg_gs/const_tm_chain.py	16	0	0	0	0	100%
/data/gssw/fg_gs/lib_config.py	653	95	0	120	48	80%
/data/gssw/fg_gs/lib_messages.py	314	150	0	38	7	49%
/data/gssw/fg_gs/lib_params.py	853	528	0	372	19	35%
/data/gssw/fg_gs/lib_srv_rv.py	10	5	0	4	0	36%
/data/gssw/fg_gs/lib_tc_handler.py	371	338	0	204	3	8%
/data/gssw/fg_gs/lib_tm_depacketizer.py	13	7	0	0	0	46%
/data/gssw/fg_gs/lib_tm_handler.py	23	15	0	10	0	24%
/data/gssw/fg_gs/tools/gs_server.py	1399	825	0	541	100	37%
/data/gssw/fg_gs/tools/tc_chain.py	556	517	0	238	3	5%
/data/gssw/fg_gs/tools/tm_chain.py	277	248	0	88	2	8%
/data/gssw/fg_log/lib_logging.py	95	19	0	18	5	75%
/data/gssw/fg_pkt/tools/tmtc_db_manager.py	303	274	0	74	2	8%
/data/gssw/fg_process/lib_process.py	194	57	0	76	23	67%
/data/gssw/fg_process/lib_python_process.py	202	129	0	69	10	31%
/data/gssw/fg_proxy/const_proxy.py	107	0	0	0	0	100%
/data/gssw/fg_proxy/lib_db_zmqspy.py	70	40	0	8	0	38%
/data/gssw/fg_proxy/lib_proxy.py	606	569	0	232	1	5%
Total	7421	4548	0	2512	245	33%

Figura 13: Tests de las herramientas del GSSW.

3.3. Resultados obtenidos

Los resultados obtenidos al concluir la fase comprobación y verificación han coincidido perfectamente con las necesidades expuestas en un principio, acerca del alcance y funcionalidades que se pretendían abarcar. El resultado ha sido probado por el equipo encargado de operar el satélite LUME-1 y han valorado positivamente la utilidad de la aplicación desarrollada en sus tareas cotidianas, tanto para la planificación de tareas diarias como para la reutilización de tareas periódicas. La aplicación ha cumplido también las expectativas de diseño, las cuales pretendían también ampliar su uso para ser empleada en dispositivos portátiles, tales como tablets.

4. Conclusiones

Ante las necesidades expuestas en el punto 2.1, y las limitaciones derivadas de la primera implementación de este servicio expuestas en el punto 2.2, se pone en valor que los resultados alcanzados mediante la nueva implementación de esta funcionalidad son bastante satisfactorios.

Ahora gracias a esta herramienta será posible la reutilización de una manera mucho más cómoda y visual, de las distintas operaciones periódicas que se programan en el satélite cuando este no tiene comunicación directa con la estación terrena. Pudiendo de esta forma crear distintas librerías de instrucciones para cada caso y pudiéndolas reconfigurar de una manera mucho más rápida y legible.

No cabe duda, de que será una herramienta utilizada en el día a día de todos aquellos operadores de misión que utilicen el software desarrollado por el equipo de la Agrupación Aeroespacial de la Universidad de Vigo.

5. Bibliografía

Referencias

- [1] “Docker documentation.” [Online]. Available: <https://docs.docker.com/>
- [2] “Django documentation.” [Online]. Available: <https://docs.djangoproject.com/es/2.2/>
- [3] “JsGrid documentation.” [Online]. Available: <http://js-grid.com/docs/>
- [4] “Django simple rest documentation.” [Online]. Available: <https://github.com/croach/django-simple-rest>
- [5] “WebSockHop documentation.” [Online]. Available: <https://github.com/fanout/websockhop>
- [6] “Airdatepicker documentation.” [Online]. Available: <http://t1m0n.name/air-datepicker/docs/>
- [7] “Require.js documentation.” [Online]. Available: <https://requirejs.org/>
- [8] S. Valera, “Telemetry and telecommand packet utilization.” [Online]. Available: https://indico.esa.int/event/85/contributions/3351/attachments/2713/3137/1225-ADCSS_2015_-_ECSS-E-ST-70-41C_status_-_S._Valera_-_Issue_1.pdf
- [9] “European cooperation for space standardization.” [Online]. Available: <https://ecss.nl/>
- [10] “About CubeSat.” [Online]. Available: <http://www.cubesat.org/about>

Anexos

A. Estado del Arte.

Como introducción a los que son los satélites pequeños y su programación caben destacar dos nombres los cuales fueron los impulsores de la creación del estándar para el desarrollo de los CubeSat, ellos son el profesor Jordi Puig-Suari de la California Polytechnic State University y el profesor Robert 'Bob' Twiggs de Stanford. Estas especificaciones buscan la simplificación de la estructura del satélite con el fin de abaratar costes y de acabar con las comunes trabas burocráticas que solía haber entre el lanzador y el equipo encargado en desarrollar el satélite. Los CubeSat destacan por su tamaño reducido y por ser accesibles para aquellas universidades que pretendan realizar proyectos de ciencia espacial.

Todo satélite precisa de un servicio de gestión de tareas, ya que este forma parte de los servicios obligatorios definidos en cualquier estándar (p.ej ECSS, CubeSat, etc.). Cada empresa u organización desarrolla su propio software con el fin de adecuarlo al objetivo de su misión, y mejorándolo ante cada nueva necesidad o fallo que encuentren en el transcurso de las misiones realizadas. Por ello la mayoría de este software es de carácter privado y hasta la fecha no se ha realizado un estudio exhaustivo sobre la historia del desarrollo de estas aplicaciones.

Lo que si se puede comentar en mayor medida es la evolución de las aplicaciones desarrolladas para las distintas misiones que ha promovido y desarrollado la Universidad de Vigo. Hasta el momento han sido cuatro misiones:

1. Xatcobeo
2. Humsat
3. Serpens
4. Lume

En las tres primeras misiones, se utilizaba una aplicación de escritorio para la gestión del satélite basada en JAVA, en la cual también había un servicio de gestión de tareas similar al explicado en el punto 2.2, en el cual se podía enviar un fichero o redactar una programación manual. Después por razones de escalabilidad y disponibilidad se decidió cambiar a una aplicación web como medio para gestionar las operaciones del satélite.

Esto ya fue realidad para el caso del LUME-1. A mayores y como novedad, se añadió el servicio de Request/Verification, el cual es un servicio que asiste al operador para saber las fases por las que pasa una instrucción desde que se acepta por parte del OBSW hasta que se completa. Esto causa que el satélite mande esta retroalimentación en formato de telemetría a la estación de tierra, así el operario es conocedor mediante la visualización de la instrucción enviada del estado actualizado del proceso.

B. Documentación API.

12/6/2019

Swagger Editor

Operation Management API ^{1.0.0}

[Base URL: localhost:8080/]

This is a description of the API methods contained in the service to manage the operations of a small satellite. CINAÉ <https://www.cinae.org/en/home/> or on <https://alen.space/es/inicio/>.

[Terms of service](#)

[Contact the developer](#)

[Apache 2.0](#)

[Find out more about Swagger](#)

Schemes

HTTP

Batch Collection of time-based operations



GET /batch_table/api Get the operation batch data

Parameters

Try it out

Name	Description
pageIndex * required string (query)	Pagination, current page index value.
pageSize * required string (query)	Pagination, current page size value.
sortName string (query)	Sorting, current column sorted.
sortOrder string (query)	Sorting, current column sorted order.

http://localhost:69/

1/25

Name	Description
name string (query)	Filtering, by batch name.
description string (query)	Filtering, by batch description.
SC string (query)	Filtering, by sc name.
apid string (query)	Filtering, by apid name.
scheduled string (query)	Filtering, by the first operation (ordered by scheduled) scheduled field inside the batch.

Responses

Response content type

Code	Description
------	-------------

Code	Description
200	<i>Successful operation</i>
<div>Example Value Model</div> <pre>[{ "id": 0, "name": "string", "description": "string", "apid": [{}], "pairs": "string", "date_created": "2019-06-12T08:11:01.253Z", "validated": true, "awaiting_validation": true, "validated_at": "2019-06-12T08:11:01.253Z", "awaiting_send": true, "sent_time": "2019-06-12T08:11:01.253Z", "sent": true }]</pre>	
500	<i>Invalid query value</i>

POST **/batch_table/api** Create operation batch

Parameters

[Try it out](#)

Name	Description
name * required string (formData)	Creates batch name.
description * required string (formData)	Creates batch description.
apid * required integer(\$int64) (formData)	Apid ID

Name	Description
pairs * required string (formData)	Creates batch instruction pairs ACKs. Default value : { "ackFlagsAcceptance": "0", "ackFlagsProgress": "0", "ackFlagsStart": "0", "appProcessId": 1000}

Responses	Response content type	application/json
-----------	-----------------------	------------------

Code	Description				
201	<i>An object with the batch ID.</i> <table><tr><td>Example Value</td><td>Model</td></tr><tr><td><pre>{ "id": 0 }</pre></td><td></td></tr></table>	Example Value	Model	<pre>{ "id": 0 }</pre>	
Example Value	Model				
<pre>{ "id": 0 }</pre>					
500	<i>Error in form data</i>				

PUT	/batch_table/api/{batch_id}	Update an existing batch
-----	-----------------------------	--------------------------

Parameters	Try it out
------------	------------

Name	Description
batch_id * required integer(\$int64) (path)	Batch ID.
name string (formData)	Updates batch name.
description string (formData)	Updates batch description.

Name	Description
scheduled	Updates the first operation ordered by scheduled field and updates the rest of the operation scheduled fields keeping the time gap between them.
string	
(formData)	
pairs	Updates batch instruction pairs ACKs.
string	Default value : { "ackFlagsAcceptance": "0", "ackFlagsProgress": "0", "ackFlagsStart": "0", "appProcessId": 1000}
(formData)	

Responses

Response content type

application/json

Code	Description
200	Successful operation
403	Forbidden access
500	Error / Batch not found

DELETE /batch_table/api/{batch_id} Deletes a batch

Parameters

Try it out

Name	Description
batch_id * required	Batch id to delete
integer(\$int64)	
(path)	

Operation Library Collection of related instructions

GET /batch_table/optype/api Get the operation library data

Parameters

Try it out

Name	Description
------	-------------

pageIndex string (query)	Pagination, current page index value.
--------------------------------	---------------------------------------

pageSize string (query)	Pagination, current page size value.
-------------------------------	--------------------------------------

sortName string (query)	Sorting, current column sorted.
-------------------------------	---------------------------------

sortOrder string (query)	Sorting, current column sorted order.
--------------------------------	---------------------------------------

name string (query)	Filtering, by operation library name. Required pageIndex and pageSize in order to work.
---------------------------	---

description string (query)	Filtering, by operation library description. Required pageIndex and pageSize in order to work.
----------------------------------	--

Responses

Response content type

Code	Description
------	-------------

Code	Description
------	-------------

200	<i>Successful operation</i>
-----	-----------------------------

Example Value	Model
---------------	-------

```
[
  {
    "id": 0,
    "name": "string",
    "description": "string",
    "date_created": "2019-06-12T08:11:01.264Z",
    "validated": true,
    "awaiting_validation": true,
    "awaiting_send": true,
    "validation_info": "string",
    "validated_at": "2019-06-12T08:11:01.264Z",
    "allow_gen_binary": true,
    "bin": "string",
    "sent_time": "2019-06-12T08:11:01.264Z",
    "sent": true
  }
]
```

500	<i>Error</i>
-----	--------------

POST /batch_table/optype/api Create operation library

Parameters

Try it out

Name	Description
name string (formData)	Creates operation library name.
description string (formData)	Creates operation library description.

Responses

Response content type

application/json

Code	Description
201	<i>An object with the Operation Library ID.</i>
<div>Example Value Model</div> <pre>{ "id": 0 }</pre>	
500	<i>Error in form data</i>

PUT **/batch_table/optype/api/{optype_id}** Update an existing operation library

Parameters

[Try it out](#)

Name	Description
optype_id * required integer(\$int64) (path)	Operation library ID.
name string (formData)	Updates operation library name.
description string (formData)	Updates operation library description.

Responses

Response content type

[application/json](#)

Code	Description
200	<i>Successful operation</i>
403	<i>Forbidden access</i>

Code	Description
500	<i>Batch not found</i>

DELETE `/batch_table/optype/api/{optype_id}` Deletes an operation library and its instructions

Parameters Try it out

Name	Description
optype_id * required integer(\$int64) (path)	Operation library id to delete

Responses Response content type **application/json**

Code	Description
200	<i>Successful operation</i>
500	<i>Error / Batch not found</i>

Operation Time-based operation library



GET `/batch_table/{batch_id}/api` Get the operations data

Parameters Try it out

Name	Description
batch_id * required integer(\$int64) (path)	Batch ID.
pageIndex * required string (query)	Pagination, current page index value.
pageSize * required string (query)	Pagination, current page size value.
sortName string (query)	Sorting, current column sorted.
sortOrder string (query)	Sorting, current column sorted order.
name string (query)	Filtering, by operation name.
description string (query)	Filtering, by operation description.
parameters string (query)	Filtering, by operation parameters.
op_type integer (query)	Filtering, by operation library.

Responses

Response content type

application/json

Code	Description
------	-------------

Code	Description
200	<i>Successful operation</i>
<div><div>Example Value</div><div>Model</div><pre>[{ "id": 0, "scheduled": "2019-06-12T08:11:01.271Z", "name": "string", "description": "string", "paramaters": ["string"], "batch": [{ "id": 0, "name": "string", "description": "string", "apid": [{}], "pairs": "string", "date_created": "2019-06-12T08:11:01.271Z", "validated": true, "awaiting_validation": true, "validated_at": "2019-06-12T08:11:01.271Z", "awaiting_send": true, </pre></div>	

Code	Description
------	-------------

201	<i>Successfull operation returns firstItem by schedule field</i>
-----	--

Example Value	Model
---------------	-------

```
{
  "id": 0,
  "scheduled": "2019-06-12T08:11:01.272Z",
  "name": "string",
  "description": "string",
  "paramaters": [
    "string"
  ],
  "batch": [
    {
      "id": 0,
      "name": "string",
      "description": "string",
      "apid": [
        {}
      ],
      "pairs": "string",
      "date_created": "2019-06-12T08:11:01.272Z",
      "validated": true,
      "awaiting_validation": true,
      "validated_at": "2019-06-12T08:11:01.272Z",
      "awaiting_send": true,
      "sent_time": "2019-06-12T08:11:01.272Z",
    }
  ]
}
```

500	<i>Error</i>
-----	--------------

POST /batch_table/{batch_id}/api Create operation

Parameters

Try it out

Name	Description
batch_id * required integer(\$int64) (path)	Relates with batch ID.
op_type * required integer(\$int64) (formData)	Relates with operation library.

Name	Description
name string (formData)	Creates operation name.
description string (formData)	Creates operation description.
scheduled * required string(\$date-time) (formData)	Creates time of execution of the operation.
parameters string (formData)	Creates operation parameters.

ResponsesResponse content typeapplication/json

Code	Description				
201	<i>An object with the operation ID.</i> <table><thead><tr><th>Example Value</th><th>Model</th></tr></thead><tbody><tr><td><pre>{ "id": 0 }</pre></td><td></td></tr></tbody></table>	Example Value	Model	<pre>{ "id": 0 }</pre>	
Example Value	Model				
<pre>{ "id": 0 }</pre>					
500	<i>Error in form data / Operation library not found</i>				

Name	Description
batch_id * required integer(\$int64) (path)	Batch ID.
operation_id * required integer(\$int64) (path)	Operation ID.
name string (formData)	Updates operation name.
description string (formData)	Updates operation description.
scheduled string(\$date-time) (formData)	Updates time of execution of the operation.
parameters string (formData)	Updates operation parameters.
op_type integer(\$int64) (formData)	Updates operation library. This is a 1 to 1 relationship

Responses

Response content type **application/json**

Code	Description
200	<i>Successful operation</i>
403	<i>Forbidden access</i>
500	<i>Error / Operation not found /Operation library not found</i>

DELETE `/batch_table/{batch_id}/api/{operation_id}` Deletes an operation

Parameters Try it out

Name	Description
batch_id * required integer(\$int64) (path)	Batch ID.
operation_id * required integer(\$int64) (path)	Operation ID.

Responses Response content type application/json

Code	Description
200	<i>Successful operation</i>
500	<i>Error / Batch not found</i>

Instruction

Instruction TC for the APID



GET `/batch_table/op_instructions/{op_type}/api` Get the instructions data

Parameters Try it out

Name	Description
op_type * required integer (path)	Operation library ID

Name	Description
pageIndex * required string (query)	Pagination, current page index value.
pageSize * required string (query)	Pagination, current page size value.
sortName string (query)	Sorting, current column sorted.
sortOrder string (query)	Sorting, current column sorted order.
apid string (query)	Filtering, by apid.
telecommand string (query)	Filtering, by telecommand name.
name string (query)	Filtering, by instruction name.
description string (query)	Filtering, by instruction description.
extra_checks string (query)	Filtering, by instruction extra_checks.
extra_info string (query)	Filtering, by instruction extra_info.

Responses

Response content type

application/json

Code **Description**

200 *Successful operation*

Example Value **Model**

```
[
  {
    "id": 0,
    "name": "string",
    "description": "string",
    "apid": [
      {}
    ],
    "telecommand": [
      {}
    ],
    "pairs": "string",
    "extra_checks": "string",
    "extra_info": "string",
    "op_type": [
      {
        "id": 0,
        "name": "string",
        "description": "string",
        "date_created": "2019-06-12T08:11:01.282Z",
        "validated": true,
        "awaiting_validation": true,
        "awaiting_send": true,

```

500 *Error*

POST **/batch_table/op_instructions/{op_type}/api** Create instruction

Parameters[Try it out](#)

Name	Description
op_type * required integer (path)	Operation library ID
apid * required integer(\$int64) (formData)	Apid ID

Name	Description
telecommand * required	Telecommand ID
integer(\$int64) (formData)	Default value : 86
name string (formData)	Creates instruction name.
description string (formData)	Creates instruction description.
pairs * required string (formData)	Creates instruction pairs ACKs. Default value : { "ackFlagsAcceptance": "0", "ackFlagsCompletion": "0", "ackFlagsProgress": "0", "ackFlagsStart": "0", "appProcessId": "" }
extra_checks string (formData)	Creates instruction extra_checks.
extra_info string (formData)	Creates instruction extra_info.

Responses

Response content type **application/json**

Code	Description
201	<i>An object with the instruction ID.</i>
	Example Value Model
	<pre>{ "id": 0 }</pre>
500	<i>Error in form data / Invalid operation library</i>

PUT **/batch_table/op_instructions/{op_type}/{instruction}/api** Get the activities data

Parameters

[Try it out](#)

Name	Description
op_type * required integer (path)	Operation library ID
instruction * required integer(\$int64) (path)	Instruction ID.
apid * required integer(\$int64) (formData)	Apid ID
telecommand * required integer(\$int64) (formData)	Telecommand ID Default value : 86
name string (formData)	Updates instruction name.
description string (formData)	Updates instruction description.
pairs * required string (formData)	Updates instruction pairs ACKs. Default value : { "ackFlagsAcceptance": "0", "ackFlagsCompletion": "0", "ackFlagsProgress": "0", "ackFlagsStart": "0", "appProcessId": "" }
extra_checks string (formData)	Updates instruction extra_checks.
extra_info string (formData)	Updates instruction extra_info.

Responses

Response content type

[application/json](#)

Code **Description**

200	<i>Successful operation</i>
403	<i>Forbidden access</i>
500	<i>Error / Instruction not found /Operation library not found!</i>

DELETE **/batch_table/op_instructions/{op_type}/{instruction}/api** Deletes an instruction**Parameters**[Try it out](#)

Name	Description
op_type * required integer (path)	Operation library ID
instruction * required integer(\$int64) (path)	Instruction ID.

Responses

Response content type

[application/json](#)**Code** **Description**

200	<i>Successful operation</i>
500	<i>Error / Instruction not found /Operation library not found!</i>

Activity**GET** **/act_register/api** Get the activities registered in the gs-server**Parameters**[Try it out](#)

Name	Description
pageIndex * required string (query)	Pagination, current page index value.
pageSize * required string (query)	Pagination, current page size value.
sortName string (query)	Sorting, current column sorted.
sortOrder string (query)	Sorting, current column sorted order.
id string (query)	Filtering, by activity ID.
op_type string (query)	Filtering, by operation library name.
batch string (query)	Filtering, by batch name.
apid string (query)	Filtering, by apid name.
telecommand string (query)	Filtering, by telecommand name.
scheduled string (query)	Filtering, by scheduled field.
Responses	Response content type <input type="text" value="application/json"/>

Code	Description
------	-------------

200	<i>Successful operation</i>
-----	-----------------------------

Example Value	Model
---------------	-------

```
[
  {
    "id": 0,
    "sc": [
      {}
    ],
    "apid": [
      {}
    ],
    "telecommand": [
      {}
    ],
    "batch": "string",
    "scheduled": "2019-06-12T08:11:01.297Z",
    "op_type": "string",
    "date_created": "2019-06-12T08:11:01.297Z",
    "timestamp_last_updated": "2019-06-12T08:11:01.297Z",
    "gs_db_id": "string",
    "packet_name": "string",
    "rv_Completion": 0
  }
]
```

403	<i>Forbidden access</i>
-----	-------------------------

500	<i>Error / Instruction not found /Operation library not found!</i>
-----	--

DELETE **/act_register/{activity}/api** Deletes an activity

Parameters

Try it out

Name	Description
activity * required	Activity ID
integer	
(path)	

Responses

Response content type

application/json

Code	Description
200	<i>Successful operation</i>
500	<i>Error / Activity not found</i>

Models

```
Batch {
  id                integer($int64)
  name              string
                  Name of the batch
  description       string
                  Description of the batch
  apid              [...]
  pairs             string
                  Acks Info
  date_created      string($date-time)
                  Creation time of the batch
  validated          boolean
                  Sees if the instructions of the operation contained in the
                  batch are validated by the gs_server

  awaiting_validation boolean
                  Flag for validation request
  validated_at       string($date-time)
                  Validation time response
  awaiting_send      boolean
                  Flag for send request
  sent_time          string($date-time)
                  Sent time response
  sent               boolean
                  Sees if the instructions of the operation contained in the
                  batch can generate a sequence if True -> generates
                  Activities
}
```

```
OpType {
  id          integer($int64)
  name        string
              Name of the operation library
  description  string
              Description of the operation library
  date_created string($date-time)
              Creation time of the operation library
  validated    boolean
              Sees if the instructions contained in the library are
              validated by the gs_server

  awaiting_validation boolean
              Flag for validation request
  awaiting_send        boolean
              Flag for send request
  validation_info      string
              Information related with validation request
  validated_at         string($date-time)
              Validation time response
  allow_gen_binary     boolean
  bin                  string($binary)
  sent_time            string($date-time)
              Sent time response
  sent                 boolean
              Sees if the instructions contained in the library are send
              to the gs_server
}
```

```
Operation {
  id          integer($int64)
  scheduled   string($date-time)
              Programmable schedule for the operation
  name        string
              Name of the operation
  description  string
              Description of the operation
  paramaters  [...]
  batch       [...]
  op_type     [...]
  date_created string($date-time)
              Creation time of the operation
}
```

```
OpInstruction {
  id                integer($int64)
  name              string
                  Name of the instruction
  description       string
                  Description of the instruction
  apid              [...]
  telecommand       [...]
  pairs             string
                  Acks Info
  extra_checks      string
  extra_info        string
  op_type           [...]
  gap_time          string($date-time)
                  This is a delay that sets execution time of the
                  instruction adding this time to the operation scheduled
                  field, corresponds to Django's TimeField
}
```

```
Activity {
  id                integer($int64)
  sc                [...]
  apid              [...]
  telecommand       [...]
  batch             string
                  Batch name
  scheduled         string($date-time)
                  Programmable schedule for the activity
  op_type           string
                  Batch name
  date_created      string($date-time)
                  Creation time of the activity
  timestamp_last_updated string($date-time)
                  Update of activity
  gs_db_id          string
                  This the GS database id for the activity
  packet_name       string
                  This the activity packetname
  rv_Completion     integer
                  This the ack completion flag
}
```

C. Documentación funciones Python.

[Docs](#) » [Views](#)

Views

@file views.py

Views.

Notes:

Here are defined the different views and simple API rest calls of the app.

@author Victor Bóveda Prado @copyright 2017 - CINA E

```
class op_management.views.Activities(**kwargs)
```

```
    delete(request, activity)
```

This is for the deletion of the batch data

Parameters: activity (*integer.*) – The pk of the batch of operations.

```
    get(request)
```

```
    to_json(objects, totalSize)
```

This is for serializing the activity data

Parameters:

- objects (*QuerySet.*) – The QuerySet containing the activities paged.
- totalSize (*integer.*) – The total size of the elements in the DB. (This is needed by JsGrid in order to paged)

```
class op_management.views.Batches(**kwargs)
```

```
    delete(request, batch_id)
```

This is for the deletion of the batch data

Parameters: batch_id (*integer.*) – The pk of the batch of operations.

```
    get(request)
```

```
    post(request)
```

put(*request, batch_id*)

This is for the update of the batch data

Parameters: **batch_id** (*integer.*) – The pk of the batch of operations.

to_json(*objects, totalSize*)

This is for serializing the batch data

Parameters:

- **objects** (*QuerySet.*) – The QuerySet containing the batches paged.
- **totalSize** (*integer.*) – The total size of the elements in the DB. (This is needed by JsGrid in order to paged)

class `op_management.views.OpInstructions`(***kwargs*)

delete(*request, op_type, instruction*)

This is for deleting the op-instruction.

Parameters:

- **op_type** (*integer.*) – The pk of the operation library.
- **instruction** (*integer.*) – The pk of the op-instruction we want to delete.

get(*request, op_type*)

post(*request, op_type*)

put(*request, op_type, instruction*)

This is for updating the op-instruction data.

Parameters:

- **op_type** (*integer.*) – The pk of the operation library.
- **instruction** (*integer.*) – The pk of the op-instruction we want to update.

to_json(*request, objects, totalSize*)

This is for serializing the op-instructions data.

Parameters:

- **objects** (*QuerySet.*) – The QuerySet containing the op-instructions paged.
- **totalSize** (*integer.*) – The total size of the elements in the DB. (This is needed by JsGrid in order to paged)

class `op_management.views.OpTypes`(***kwargs*)

delete(*request*, *optype_id*)

This is for deleting the operation library.

Parameters: *optype_id* (*integer.*) – The pk of the operation library we want to delete.

get(*request*)

post(*request*)

put(*request*, *optype_id*)

This is for updating the operation library data.

Parameters: *optype_id* (*integer.*) – The pk of the operation library we want to update.

to_json(*objects*, *totalSize*)

This is for serializing the operation libraries data.

Parameters:

- **objects** (*QuerySet.*) – The QuerySet containing the operation libraries paged.
- **totalSize** (*integer.*) – The total size of the elements in the DB. (This is needed by JsGrid in order to paged)

class `op_management.views.Operations`(*****kwargs***)

delete(*request*, *batch_id*, *operation_id*)

This is for the deletion of the operation data

Parameters:

- **batch_id** (*integer.*) – The pk of the batch of operations.
- **operation_id** (*integer.*) – The pk of the operation we want to delete.

get(*request*, *batch_id*)

post(*request*, *batch_id*)

put(*request*, *batch_id*, *operation_id*)

This is for the update of the operation data

Parameters:

- **batch_id** (*integer.*) – The pk of the batch of operations.
- **operation_id** (*integer.*) – The pk of the operation we want to update.

to_json(*objects*, *totalSize*)

This is for serializing the operations data

- Parameters:
- **objects** (*QuerySet.*) – The QuerySet containing the operations paged.
 - **totalSize** (*integer.*) – The total size of the elements in the DB. (This is needed by JsGrid in order to paged)

op_management.views.keepTimeDifference(*new_scheduled*, *batch*)

This is for keeping the time difference between the operations inside the batch . When updating the scheduled field of the batch View (really it's the first_op_scheduled from the current batch)

- Parameters:
- **new_scheduled** (*string.*) – This is the scheduled field of the update called.
 - **batch** – The updated batch.

op_management.views.send_batch(*request*, *batch_id*)

This is for the “Generate” link.

- Parameters:
- **batch_id** (*integer.*) – The pk of the batch of operations.

op_management.views.send_op(*request*, *op_type*)

This is for the “Send” link.

- Parameters:
- **op_type** (*integer.*) – The pk of the operation library.

op_management.views.show_act(*request*)

Method for displaying the activities that are ready to send to the satellite

op_management.views.show_batch(*request*)

Method for displaying the operation batches of the app.

op_management.views.show_inst(*request*, *batch_id*, *operation_id*)

Method for displaying the instructions inside an operation-library (op_type) from the Batch Editor path.

op_management.views.show_lib(*request*)

Method for displaying the operation-libraries (from the Operation Library path) Services: Validate and Send!

`op_management.views.show_lib_inst(request, op_type)`

Method for displaying the instructions inside an operation-library (op_type) (from the Operation Library path)

`op_management.views.show_operation(request, batch_id)`

Method for displaying the operations inside a batch.

`op_management.views.validate_batch(request, batch_id)`

This is for the "Validate" link.

Parameters: `batch_id (integer.)` – The pk of the batch of operations.

`op_management.views.validate_op(request, op_type)`

This is for the "Validate" link.

Parameters: `op_type (integer.)` – The pk of the operation library.

`op_management.views.validate_op_all(request, batch_id)`

This is for the "Validate All" link.

Parameters: `batch_id (integer.)` – The pk of the batch i.

Models

@file models.py

Models for GS entities.

@author AJ Vazquez Alvarez @copyright 2018 - CINAE

```
class gs_web.entities.models.ApplicationProcess(id, sc, apid, name, description, tc_policy)
```

```
    exception DoesNotExist
```

```
    exception MultipleObjectsReturned
```

```
class gs_web.entities.models.ApplicationProcessAdmin(model, admin_site)
```

```
    form
```

```
    alias of ApplicationProcessAdminForm
```

```
class gs_web.entities.models.ApplicationProcessAdminForm(data=None, files=None,
auto_id='id_%s', prefix=None, initial=None, error_class=<class 'django.forms.utils.ErrorList'>,
label_suffix=None, empty_permitted=False, instance=None, use_required_attribute=None)
```

```
class gs_web.entities.models.Node(id, node_type, node_id, name, description, poll_period_s,
is_up, last_update, last_request, parent_node)
```

```
    exception DoesNotExist
```

```
    exception MultipleObjectsReturned
```

```
class gs_web.entities.models.NodeAdmin(model, admin_site)
```

```
    form
```

```
    alias of ApplicationProcessAdminForm
```

```
class gs_web.entities.models.NodeAdminForm(data=None, files=None, auto_id='id_%s',
prefix=None, initial=None, error_class=<class 'django.forms.utils.ErrorList'>, label_suffix=None,
empty_permitted=False, instance=None, use_required_attribute=None)
```

```
class gs_web.entities.models.Spacecraft(id, sc_id, name, description)
```

```
    exception DoesNotExist
```

```
    exception MultipleObjectsReturned
```

```
class gs_web.entities.models.SpacecraftAdmin(model, admin_site)
```

```
    form
```

```
        alias of SpacecraftAdminForm
```

```
class gs_web.entities.models.SpacecraftAdminForm(data=None, files=None,
auto_id='id_%s', prefix=None, initial=None, error_class=<class 'django.forms.utils.ErrorList'>,
label_suffix=None, empty_permitted=False, instance=None, use_required_attribute=None)
```

@file models.py

Models for batch/operations_type/operations/op_instructions.

Example:

see ../op_management/.py

TODO:

Here are defined the models used in the op_management app! Testing, including:
fNothing yet!

Notes:

Docstring following sphinx-numpydoc format.

@author Victor Bóveda Prado @copyright 2017 - CINAÉ

```
class gs_web.op_management.models.Activity(id, instruction, date_created, batch, scheduled)
```

```
    exception DoesNotExist
```

```
    exception MultipleObjectsReturned
```

```
class gs_web.op_management.models.Batch(id, name, description, date_created, validated,
validated_at, awaiting_validation, sent_time, sent, awaiting_send)
```

```
    exception DoesNotExist
```

```
    exception MultipleObjectsReturned
```

```
class gs_web.op_management.models.OpInstruction(id, apid, telecommand, name,
description, pairs, extra_checks, extra_info, op_type, gap_time)
```

exception **DoesNotExist**

exception **MultipleObjectsReturned**

save(*args, **kwargs)

Save the current instance. Override this in a subclass if you want to control the saving process.

The 'force_insert' and 'force_update' parameters can be used to insist that the "save" must be an SQL insert or update (or equivalent for non-SQL backends), respectively. Normally, they should not be set.

```
class gs_web.op_management.models.OpType(id, name, description, date_created, validated,
awaiting_validation, awaiting_send, validation_info, validated_at, allow_gen_binary, bin, sent_time, sent)
```

exception **DoesNotExist**

exception **MultipleObjectsReturned**

save(update=True, *args, **kwargs)

Save the current instance. Override this in a subclass if you want to control the saving process.

The 'force_insert' and 'force_update' parameters can be used to insist that the "save" must be an SQL insert or update (or equivalent for non-SQL backends), respectively. Normally, they should not be set.

```
class gs_web.op_management.models.Operation(id, scheduled, name, description, parameters,
date_created, batch, op_type)
```

exception **DoesNotExist**

exception **MultipleObjectsReturned**

save(update=True, *args, **kwargs)

Save the current instance. Override this in a subclass if you want to control the saving process.

The 'force_insert' and 'force_update' parameters can be used to insist that the "save" must be an SQL insert or update (or equivalent for non-SQL backends), respectively. Normally, they should not be set.

Services

@file services.py

Layer above models to provide extra features.

@author AJ Vazquez Alvarez @copyright 2017 - CINAÉ

op_management.services.check_process_batch_sent_response(*check_ok, ref_id, msg*)

Process sent response for batch sequence.

check_ok :bool

True if validated successfully, False otherwise.

ref_id :int

Identifier of the request.

msg :str

Details of the validation (if check_ok is False).

op_management.services.check_process_batch_validation_response(*check_ok, ref_id, msg*)

Process validation response for batch sequence.

check_ok :bool

True if validated successfully, False otherwise.

ref_id :int

Identifier of the request.

msg :str

Details of the validation (if check_ok is False).

op_management.services.check_process_op_sent_response(*check_ok, ref_id, msg*)

Process sent response for OPS sequence.

check_ok :bool

True if validated successfully, False otherwise.

ref_id :int

Identifier of the request.

msg :str

Details of the validation (if check_ok is False).

op_management.services.check_process_op_validation_response(*check_ok*, *ref_id*, *msg*)

Process validation response for OPS sequence.

check_ok :bool

True if validated successfully, False otherwise.

ref_id :int

Identifier of the request.

msg :str

Details of the validation (if check_ok is False).

op_management.services.empty_op_lib(*ref_id*)

Process validation error for empty operation library. Input --

ref_id :int

Identifier of the request.

op_management.services.get_e2e_inst_by_id(*inst_id*)

Returns a lib_sp_e2e.Instruction object given a db instruction id, or None if the instruction does not exist.

op_management.services.get_jsonized_str(*csv_name_v*, *lines_csv_v*)

Get an ordered json from a list of names of files and their associated lists of lines.

csv_name_v :list of str

List of names of each of the files.

lines_csv_v :list of list of str

List of lists of lines associated to each file.

jsonized_str:str

Serialized string with information for all the files to be sent.

op_management.services.send_seq_obj(*sequence_e2e*, *test_mode=False*, *ref_id=None*, *gen_bin=False*, *ref_type='op_validation'*)

Send a lib_sp_e2e.Sequence object directly to the GS.

seq_obj: lib_sp_e2e.Sequence

Sequence object.

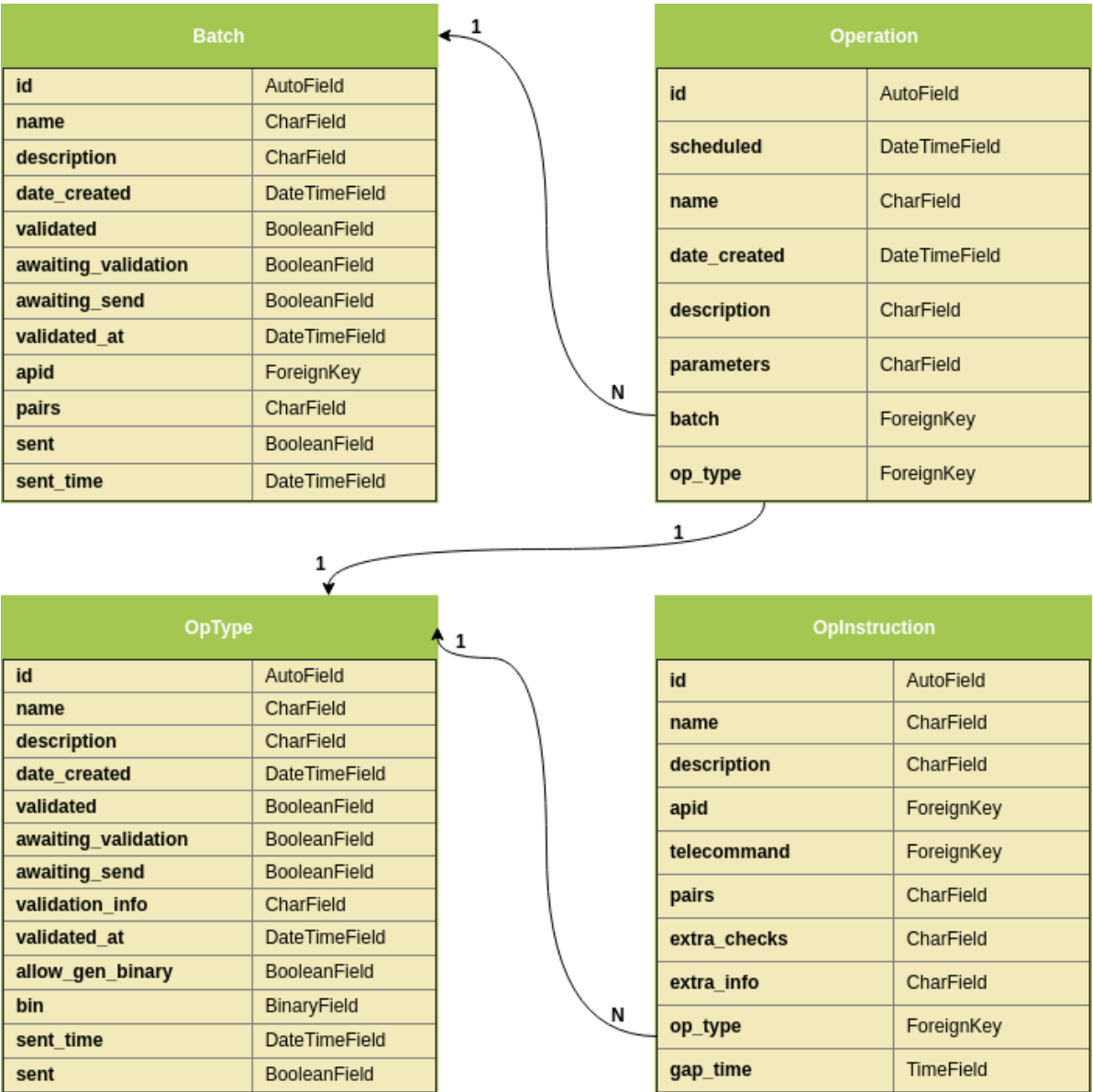
test_mode: boolean

Validation request if 1, Send request if 0.

error: {None,int}

None if no error, int otherwise.

D. Modelos y relaciones utilizados en Django.

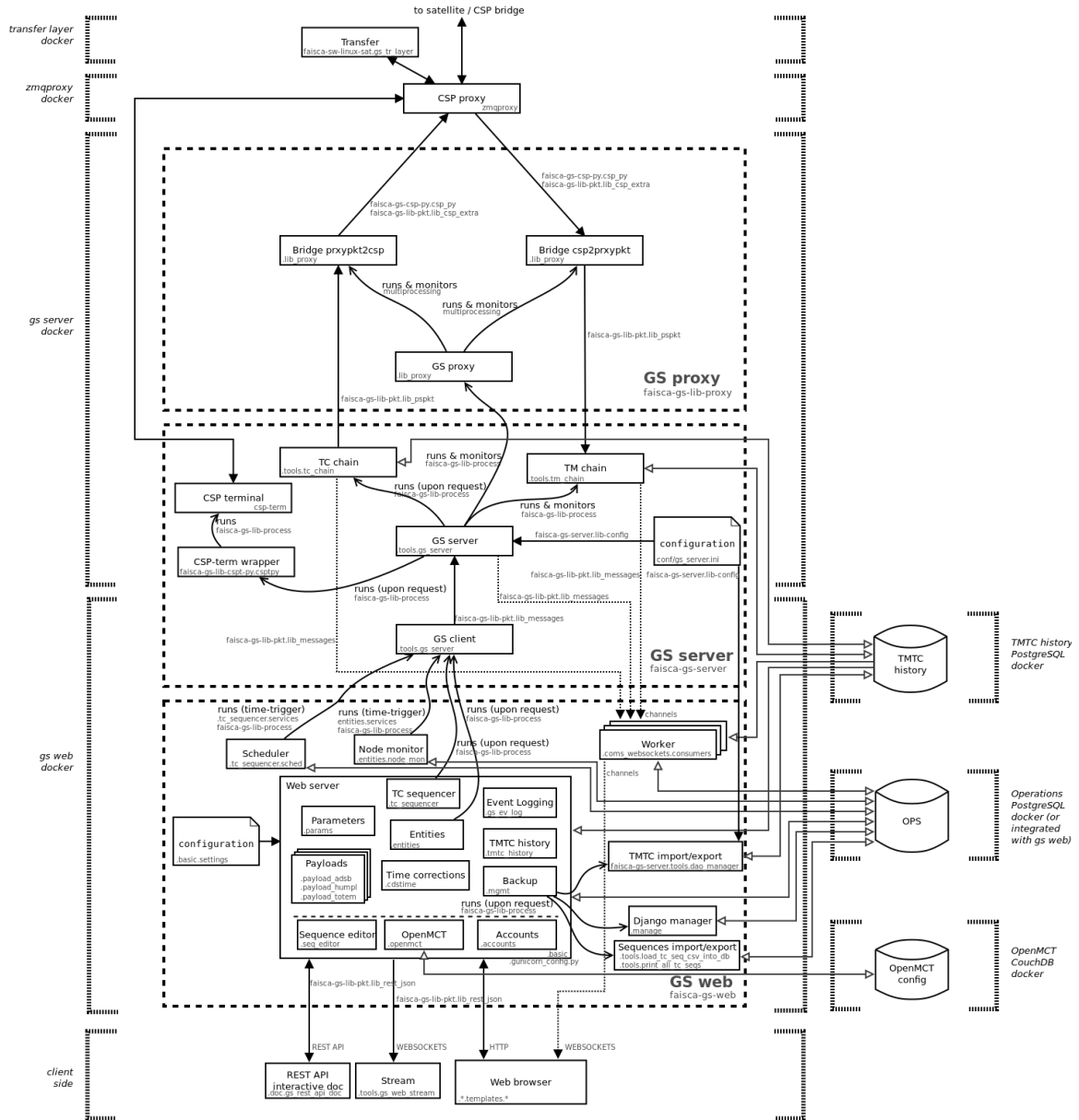


Activity	
id	AutoField
sc	CharField
apid	CharField
telecommand	CharField
batch	ForeignKey
scheduled	DateTimeField
date_created	DateTimeField
timestamp_last_updated	DateTimeField
gs_db_id	CharField
packet_name	CharField
rv_Completion	IntegerField

Figura 14: Modelos y relaciones utilizados en Django.

E. Arquitectura del GSSW.

La arquitectura global del GSSW se muestra en la siguiente imagen.



Como se puede apreciar, la arquitectura del servicio se agrupa en cinco secciones:

- **Capa de transferencia:** es la capa encargada de gestionar los telecomandos salientes o la telemetría recibida y de enviarla al satélite o al proxy.
- **Capa de proxy:** para esta capa se utiliza la herramienta `zmq_proxy`, la cual sirve para establecer comunicaciones simétricas basadas en sockets entre el front y el back. En este caso se utiliza para traspasar la información recibida hacia la terminal CSP o para enviar información a la capa de transferencia.
- **Capa de GS-Server:** se encarga de gestionar todas las operaciones relacionadas con la verificación, consulta, escritura y actualización de telecomandos y telemetría. El GS-

Server se basa en una serie de hilos de procesos que se encargan de gestionar cada uno de estos servicios.

- Capa de GS-Web: esta es la capa desarrollada en Django donde se gestionan las operaciones lanzadas por el cliente, y donde se gestionan los eventos procedentes del GS-Server.
- Capa de cliente: en esta capa se hayan los elementos que hacen posible que el cliente acceda al servicio.

F. Servicio Scheduler.

TMTC services architecture

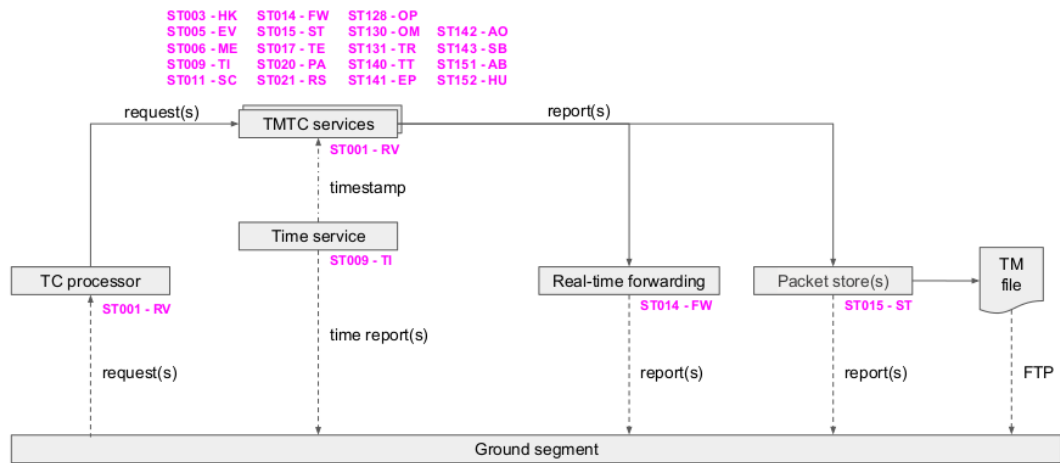


Figura 15: TMTC services in APID 1.

TMTC services architecture

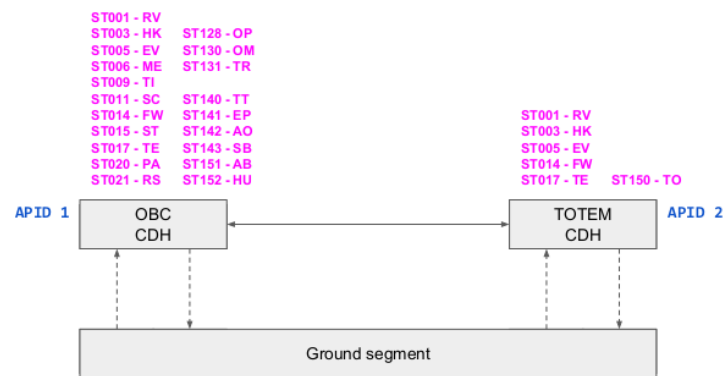
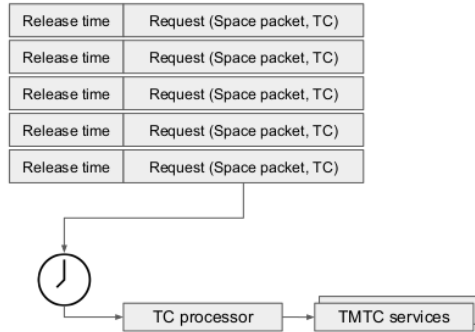


Figura 16: TMTC services in APID 1 and 2.

ST011 - Time-based scheduling service



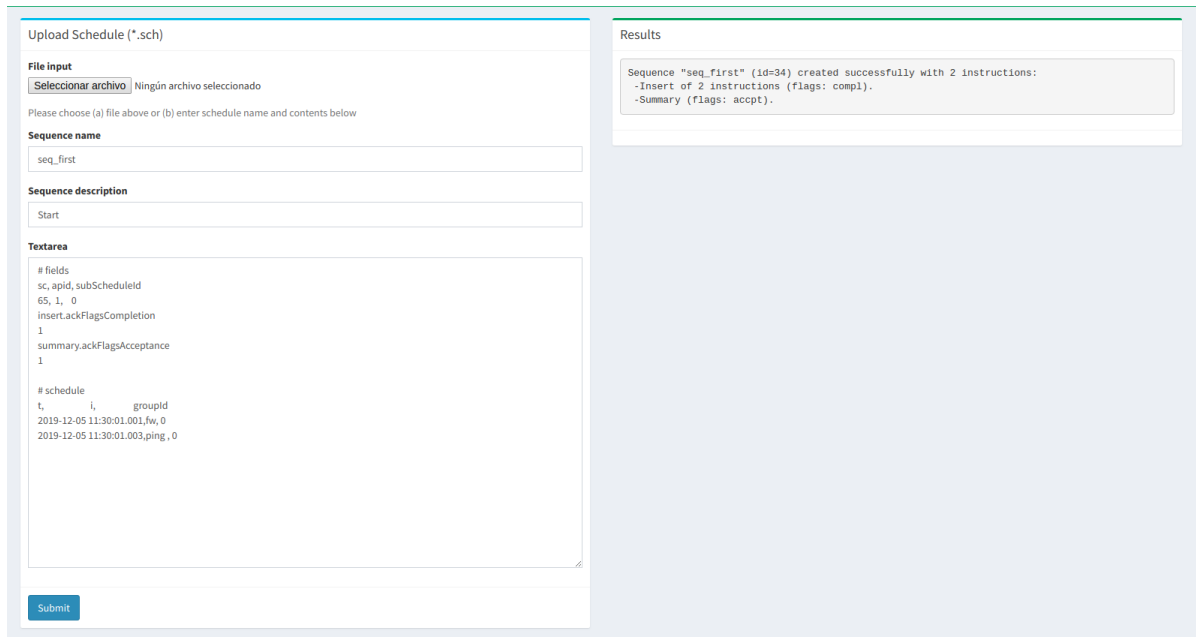
Sub-schedules and groups are not implemented for AISTECHSAT-2.

- TC[11,1] enable the time-based schedule execution function
- TC[11,2] disable the time-based schedule execution function
- TC[11,3] reset the time-based schedule
- TC[11,4] insert activities into the time-based schedule
- TC[11,15] time-shift all scheduled activities
- TC[11,17] summary-report all time-based scheduled activities
 - TM[11,13] time-based schedule summary report (*)
- TC[11,16] detail-report all time-based scheduled activities
 - TM[11,10] time-based schedule detail report (*)
- TC[11,6] delete the time-based scheduled activities identified by a filter
- TC[11,8] time-shift the scheduled activities identified by a filter
- TC[11,14] summary-report the time-based scheduled activities identified by a filter
- TC[11,11] detail-report the time-based scheduled activities identified by a filter

(*) TM[11,13] and TM[11,10] will have a maximum length of 2048 bytes. If the schedule report has a greater length it will be splitted in several TM[11,13] or TM[11,10] reports.

Figura 17: Time-based scheduling service.

G. Imágenes de aplicación.



Upload Schedule (*.sch)

File Input
 Seleccionar archivo Ningún archivo seleccionado

Please choose (a) file above or (b) enter schedule name and contents below

Sequence name
 seq_first

Sequence description
 Start

Textarea

```
# fields
sc, apid, subScheduleId
65, 1, 0
insertAckFlagsCompletion
1
summary.ackFlagsAcceptance
1

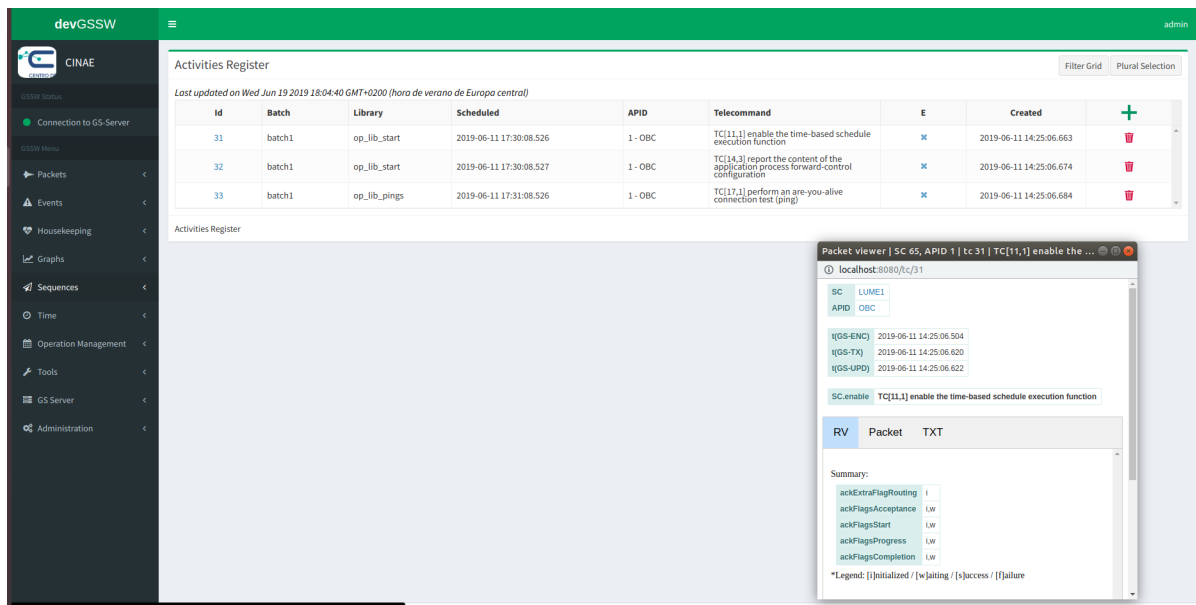
# schedule
t, i, groupId
2019-12-05 11:30:01.001,fw,0
2019-12-05 11:30:01.003,ping,0
```

Submit

Results

Sequence "seq_first" (id=34) created successfully with 2 instructions:
 -Insert of 2 instructions (flags: compl).
 -Summary (Flags: acpt).

Figura 18: Interfaz de usuario de la sección Upload Schedule.



devGSSW admin

Activities Register

Last updated on Wed Jun 19 2019 18:04:40 GMT+0200 (hora de verano de Europa central)

ID	Batch	Library	Scheduled	APIID	Telecommand	E	Created	
31	batch1	op_lib_start	2019-06-11 17:30:08.526	1 - OBC	TC[11.1] enable the time-based schedule execution function	✖	2019-06-11 14:25:06.663	+
32	batch1	op_lib_start	2019-06-11 17:30:08.527	1 - OBC	TC[4.3] report the content of the application process forward-control configuration	✖	2019-06-11 14:25:06.674	+
33	batch1	op_lib_pings	2019-06-11 17:31:08.526	1 - OBC	TC[17.1] perform an are-you-alive connection test (ping)	✖	2019-06-11 14:25:06.684	+

Packet viewer | SC 65, APIID 1 | tc 31 | TC[11.1] enable the ...

localhost:8080/8c/31

SC LUME1
APIID OBC

[OS-ENC] 2019-06-11 14:25:06.504
 [OS-TX] 2019-06-11 14:25:06.620
 [OS-UPD] 2019-06-11 14:25:06.622

SC.enable TC[11.1] enable the time-based schedule execution function

RV Packet TXT

Summary:

```
ackExtraFlagRouting I
ackFlagsAcceptance LW
ackFlagsStart LW
ackFlagsProgress LW
ackFlagsCompletion LW
```

*Legend: [i]initialized / [w]aiting / [s]uccess / [f]ailure

Figura 19: Vista de Activity Register con Packet Viewer.

H. Documentación framework, lenguajes y plugins.

Las distintas herramientas utilizadas en la elaboración del proyecto se comentan en esta sección:

- **Docker:** la idea de Docker es crear contenedores ligeros y portables para las aplicaciones software que puedan ejecutarse en cualquier máquina con Docker instalado, independientemente del sistema operativo que la máquina tenga por debajo, facilitando así también los despliegues. Los contenedores de Docker son significativamente más pequeños que las imágenes de máquina virtuales y además Docker es OpenSource. Debido a esta versatilidad y potencia Docker es una de las herramientas más utilizadas por desarrolladores de software, ingenieros de sistemas y miembros de la comunidad de desarrolladores de aplicaciones.
- **Django:** es un framework de aplicaciones web gratuito y de código abierto (open source) escrito en Python. La idea de este framework es adaptar de forma sencilla todas las necesidades recurrentes de la programación web con toda la potencia computacional que nos ofrece Python.
- **Airdatepicker:** es un plugin ligero basado en jQuery, que nos permite la elección de fecha y hora. Airdatepicker está desarrollado en ES5 y posee un estilo elegante y actual. La versión utilizada en esta aplicación es una adaptación de este plugin, a la cual se le ha aumentado su precisión hasta los milisegundos.
- **WebSockets:** es una tecnología avanzada que hace posible abrir una sesión de comunicación interactiva entre el navegador del usuario y un servidor. Con esta API, puede enviar mensajes a un servidor y recibir respuestas controladas por eventos sin tener que consultar al servidor para una respuesta.
- **Bootstrap:** facilita la maquetación de sitios web, además de ser compatible con preprocesadores como Less y SaaS, nos ofrece las herramientas para que nuestro sitio web se vea bien en toda clase de dispositivos, ahorrándose así el trabajo de tener que rediseñar un sitio web.
- **JsGrid:** es una librería ligera que permite trabajar con datos en formato de tabla desde el lado del cliente. Está basado en jQuery y admite operaciones de cuadrícula básicas como la inserción, el filtrado, la edición, la eliminación, la paginación y la clasificación. JsGrid es flexible y permite personalizar su apariencia y componentes.