UNIVERSIDAD DE MENDOZA
FACULTAD DE INGENIERÍA

TESIS DE
MAESTRÍA EN TELEINFORMÁTICA

Plataforma SaaS de impresión 3D

Autor: César Omar Aranda

Director: Mg. Ing. Diego Navarro

Mendoza – Julio 2020

ÍNDICE

RESUMEN 5

I. INTRODUCCIÓN 6

CONTROL DE LA IMPRESIÓN 3D 6

OBJETIVO PRINCIPAL 10

OBJETIVOS SECUNDARIOS 11

BENEFICIOS ESPERADOS 11

ESTRUCTURA DEL TRABAJO 12

II. MARCO TEÓRICO Y ANTECEDENTES 13

PLATAFORMA 13

XaaS 14

API REST 16

LA IMPRESIÓN 3D 19

COMANDOS G-CODE 20

SISTEMA EMBEBIDO Y SBC 23

SERVIDOR WSGI 24

PATRONES DE DISEÑO WEB HABITUALES 27

AUTENTICACIÓN Y AUTORIZACIÓN MEDIANTE JWT 29

III. DESARROLLO Y RESULTADOS 32

ESCENARIO DE APLICACIÓN A 32

ESCENARIO DE APLICACIÓN B 33

ESCENARIO DE APLICACIÓN C 36

OBSERVACIONES 38
DEFINICIÓN DE REQUISITOS 39
SELECCIÓN DE LA IMPRESORA 3D 40
SELECCIÓN DEL SOFTWARE DE CONTROL 43
SELECCIÓN DE HARDWARE PARA EL HOST PRINCIPAL 45
SELECCIÓN DEL SOFTWARE DE SISTEMA 49
PROTOTIPO DE LA PLATAFORMA 54
DISEÑO DE LA API REST 56
IMPLEMENTACIÓN DE LA PLATAFORMA 65
COMUNICACIÓN ENTRE APLICACIONES DE SERVIDOR 69
DISEÑO FÍSICO DE LA PLATAFORMA 74
ESCALABILIDAD A DIFERENTES ESCENARIOS 78
DISEÑO DE LA APLICACIÓN WEB CLIENTE 79
IV. CONCLUSIONES 86
V. BIBLIOGRAFÍA 88
APÉNDICE 91
ANEXO A: MÓDULO RPC 92
ANEXO B: LANZAMIENTO DE LA PLATAFORMA 95

RESUMEN

Desde hace unos años se observa una marcada difusión y penetración de las impresoras 3D en la sociedad, tanto a nivel comercial como educativo o domiciliario.

Es una tecnología madura en algunos aspectos pero en crecimiento y evolución desde otros.

Las actuales impresoras 3D de escritorio poseen una funcionalidad y control de proceso limitados, no desde el punto de vista del tamaño o calidad de las piezas obtenidas, sino fundamentalmente desde su autonomía de trabajo y su conectividad.

Se define e implementa aquí una plataforma de impresión 3D, basada en una arquitectura orientada a servicios, como alternativa de solución a las limitaciones mencionadas, y se describen diferentes escenarios para su aplicación.

A lo largo del desarrollo se describen los requisitos a satisfacer por el sistema y las generalidades de los elementos seleccionados que conforman tanto la infraestructura del sistema como la arquitectura del software a usar, con las justificaciones de cada decisión tomada.

Se especifican las características del diseño de la API REST, se implementa y se documenta la misma.

Finalmente se muestran aspectos del diseño e implementación de las interfaces gráficas de usuario de un cliente prototipo, capaz de consumir los servicios fundamentales provistos por la plataforma.

I. INTRODUCCIÓN

En este capítulo se introduce el problema a investigar, se identifican los objetivos del trabajo y finalmente se reseña el contenido del documento.

CONTROL DE LA IMPRESIÓN 3D

Una impresora 3D es un tipo de dispositivo formado por elementos mecánicos, electrónicos e informáticos, destinado a producir piezas u objetos físicos tridimensionales.

En función de diferentes parámetros (tamaño de las piezas que puedan obtenerse, de la resolución, precisión, tiempo del proceso de impresión, materia prima, y otros) los fabricantes están ofreciendo máquinas en 3 categorías o gamas que pueden denominarse: doméstica o de escritorio (desktop), profesional e industrial.

La tecnología de impresión 3D, posee más de 30 años, sin embargo es en los últimos 10 años cuando se ha producido su crecimiento más significativo. En particular, en el mundo de las impresoras 3D de escritorio, a partir tanto de la fabricación y venta masiva de componentes electrónicos pre-armados (habitualmente denominados módulos), como de tendencias filosóficas y sociales (el movimiento maker, por ejemplo) que promueve que las mismas usen electromecánica accesible a todos y sean de código abierto.

El desarrollo dentro del sector ha sido importante en los últimos años y cada vez, con mayor velocidad, se obtienen resultados en nuevas áreas de aplicación.

Desde un punto de vista general, para administrar las operaciones que se llevan adelante durante el proceso de impresión, se presentan diversas soluciones:

A. Mantener una computadora Host, de escritorio o portátil, conectada local y permanentemente al controlador de la impresora,

B. Agregar módulos electrónicos complementarios, generalmente controladores con display y memoria RAM adicional (mediante el uso de SD o microSD) donde alojar el archivo del modelo a imprimir, de modo de independizarse parcialmente de la computadora Host. Podría identificarse con modelos autónomos (stand-alone).

C. Reemplazar la computadora por módulos basados en microprocesadores y/o microcontroladores con prestaciones intermedias entre las 2 opciones anteriores.

D. Reemplazar la computadora y el controlador por una única minicomputadora.

La mayoría de las soluciones disponibles actualmente corresponden al tipo A, habitualmente con módulos basados en microcontroladores del tipo Arduino o similares. En este caso, lo habitual es usar aplicaciones instaladas en una PC o notebook como Printrun Pronterface, Repetier Host, Cura 3D o Makerbot MakerWare, orientados a servir de interfaz de usuario para las tareas iniciales de configuración, control y monitoreo.

Las soluciones del tipo B, buscan independizarse de la PC o notebook durante el proceso de impresión. Generalmente, las interfaces usadas se encuentran adheridas físicamente a la impresora.

Entre las soluciones del tipo C, es habitual el uso de módulos basados en microprocesadores del tipo Rasberry Pi.

Las soluciones del tipo D se aplican en general al ámbito industrial o comercial.

Se suma, a las anteriores configuraciones, la incorporación de módulos con dispositivos de red que permiten obtener información de manera remota.

En las impresoras comerciales o en ciertas tecnologías propietarias y bajo dispositivos con IOS (Apple - Macintosh) existen soluciones que permiten, a partir del uso de Internet, gestionar la impresión mediante tecnologías cloud computing.

Actualmente, excepto para el caso de soluciones del tipo C, y específicamente para Rasberry Pi, existen algunas aplicaciones de control y monitoreo remoto, ya sea que se orienten a web o a dispositivos móviles. Son los casos de Simplify3D, AstroPrint, MatterControl Touch, OctoPrint y Whip/Appaloza.

Un exponente diferente y en evolución es 3D PrinterOS, un sistema operativo para la gestión y el control remoto impresoras 3D, capaz de correr sobre cualquier entorno.

Al trabajar con impresoras 3D, debe usarse también otros tipos de software.

En primer lugar, aquel orientado, en general, a resolver los problemas de diseño 3D y a obtener archivos donde se encuentra almacenado el modelo del objeto 3D deseado. Para esto, existe un amplio y variado conjunto de aplicaciones, entre las que pueden mencionarse a modo de ejemplo, Blender, Sketch Up, SolidWorks, Autocad, SolidEdge, Zbrush, Maya, 3DStudio Max, Open SCAD, FreeCAD, etc.

En segundo lugar, el software convertidor de formatos, generalmente disponible como pequeñas herramientas, capaces de obtener los archivos de base requeridos para que la impresora pueda realizar la fabricación, a partir de los archivos de diseño anteriores. Existen diferentes tipos de archivos soportados por las impresoras 3D (stl, stlb, collada, wrml, vrml, ply, 3ds, fbx, zpr). En este grupo se encuentran, por ejemplo, MeshLab, Google SketchUp, 123D, 3DVia, Online 3D model converter.

Cabe aclarar, que dadas las necesidades de obtener archivos en formato de impresoras 3D, las aplicaciones del primer grupo, van incorporando

opciones de trabajo con herramientas como las mencionadas en el segundo. Pero aún presentan ciertas incompatibilidades y no soportan todos los tipos de formatos existentes.

En tercer lugar, hace falta convertir los archivos de base en instrucciones comprensibles para la impresora 3D, denominadas según el caso gcode, s3g o x3g. En este grupo encontramos a Slic3r, Axon, Replicator G, Makerware, Marlin, Skeinforge entre otros.

Desde el punto de vista de las aplicaciones dedicadas a la configuración, control y monitoreo del proceso de impresión (habitualmente conocidas como Aplicaciones del Host de Impresión), existen diferentes alternativas, relacionadas con el tipo de solución tecnológica adoptado para interconectar el controlador de la impresora 3D con el equipo donde corre la aplicación host.

Al analizar desde el punto de vista del control y/o monitoreo remoto de una impresora 3D, propias del segmento de mercado de las impresoras de escritorio de fabricación personal, se observa que la mayoría de las soluciones propuestas consiste en aplicaciones que actuando como Host del proceso de impresión, operan también en modo servidor para los clientes remotos.

Dichas aplicaciones corren en general sobre computadoras personales y en mucho menor medida en microcomputadoras Raspberry. Existen mínimas referencias usando otras tecnologías (como BeagleBone o Arduino y derivados).

Algunos fabricantes de impresoras 3D, ofrecen aplicaciones de control basados en comunicaciones mediante redes.

Se observa una lenta migración de todas las aplicaciones mencionadas hacia arquitecturas orientadas a la web y particularmente a las más

modernas orientadas a servicios. En este sentido cabe destacar los avances realizados por OctoPrint.

Gartner, afirma que las nuevas plataformas tecnológicas y de servicios para IoT, serán un enfoque clave hasta 2020 (Panetta, 2016).

En la Figura 1 puede observarse la forma en que operan en general.

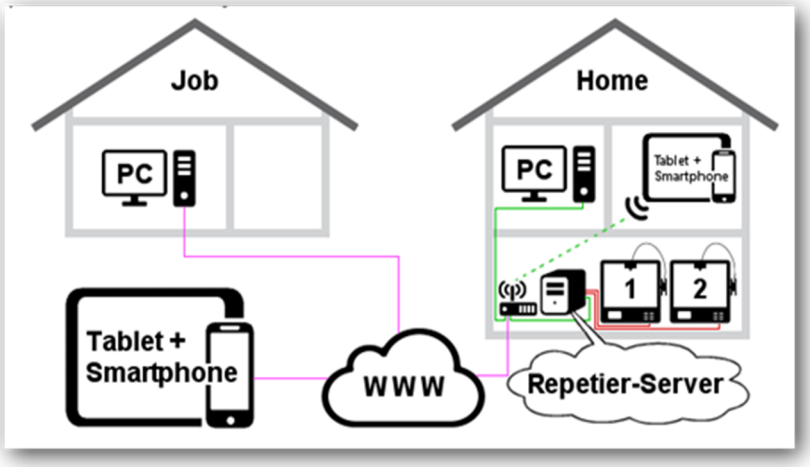

Figura 1: Conectividad de Repetier-Server con aplicación de celular.
Fuente: repetier-server.com/

En algunos casos, dentro del segmento elegido, se tiene la posibilidad de descargar e instalar en el dispositivo una aplicación de monitoreo o control. Generalmente, estas aplicaciones poseen funcionalidades reducidas, a veces sólo limitadas iniciar o parar la impresión, y a recibir mensajes del estado del proceso.

OBJETIVO PRINCIPAL

- Diseñar e implementar una plataforma SaaS para el monitoreo y control de impresiones 3D.

OBJETIVOS SECUNDARIOS

- Reemplazar la necesidad de una PC o notebook directamente vinculada a la impresora haciendo uso de sistemas embebidos en SBC
- Realizar la integración de una aplicación de control de impresoras 3D preexistente con otras aplicaciones de servidor.
- Implementar un cliente web (front-end) capaz de consumir y vincular diferentes servicios de la API REST de la plataforma.
- Evaluar la escalabilidad de la plataforma a diversos escenarios.

BENEFICIOS ESPERADOS

Como se describió anteriormente, una de las características operativas de la mayoría de las impresoras 3D no comerciales actuales, es que responden a los tipos A y B. Las mismas, además de ocupar una estación de cómputo (PC o notebook) de manera casi dedicada durante el tiempo que dure la impresión, requieren que el operador se encuentre cerca, no sólo para iniciar y parar la impresión, sino para actuar ante cualquier problema.

Lograr el control y monitoreo remoto bajo una solución como la propuesta en este trabajo provee varios beneficios, tanto para el usuario de la impresora como para la comunidad teleinformática en general. Entre ellos, pueden mencionarse:

- La posibilidad de operar el dispositivo sin tener que encontrarse en el lugar para configurar o controlar el proceso de impresión.
- La reducción de los costos del equipamiento, al no requerir de una computadora local que cumpla la función de host.
- La reducción del espacio ocupado, por la misma razón anterior.

- Capacidad para administrar 1 o más impresoras 3D, configuradas tanto para en un ámbito local como distribuido, para 1 o muchos usuarios.
- Un modelo de integración de software para controlar dispositivos, usando tecnologías orientadas a los servicios a través del uso de técnicas REST y sistemas embebidos.

Se ofrece, como beneficio adicional, dejar los resultados de este trabajo disponibles para la comunidad de usuarios y hobbistas de impresoras 3D, pudiendo servir de base para otros proyectos futuros.

ESTRUCTURA DEL TRABAJO

En este documento se recorren los conceptos generales del proceso de impresión 3D, se describe el significado dado al término plataforma de impresión y se realiza una breve introducción a las tecnologías de SaaS, API REST y JWT, usadas en la propuesta.

A continuación, y dado que los servicios a proveer dependen tanto del dispositivo específico como de los archivos utilizados, se presentan las generalidades del proceso de impresión 3D y de los conceptos de G-Code que deben usarse.

Finalmente, se presentan elementos especiales de diseño adoptados en la construcción de la plataforma, como son los sistemas embebidos en una SBC, los servidores WSGI y los patrones de diseño web.

Una vez mencionados los componentes principales involucrados se procede a describir las características de la plataforma de impresión 3D propuesta.

II. MARCO TEÓRICO Y ANTECEDENTES

PLATAFORMA

En informática, el concepto de plataforma toma varios significados.

En general, puede entenderse como plataforma a un sistema que sirve como base para hacer funcionar determinados módulos de hardware y/o de software con los que es compatible. Dicho sistema está definido por uno o más estándares alrededor del cual se determina una arquitectura particular de hardware y software.

Sin embargo, también se usa el término plataforma para referirse solamente al software, esto es a todo el conjunto de aplicaciones (incluidos los entornos) requeridas para prestar un cierto servicio.

En cualquiera de los 2 casos al definir la plataforma, se establecen los tipos de arquitectura, sistema operativo, lenguaje de programación e interfaz de usuario de modo que todo sea compatible y operativo.

Muchas veces el término plataforma se entiende conociendo el contexto de análisis. Por lo cual para precisar su interpretación, es conveniente usar combinaciones como plataforma tecnológica, plataforma de explotación, plataforma virtual, plataforma de comunicación, plataforma de videojuegos o plataforma IoT, entre otras.

En este trabajo, se usará el término según la segunda acepción, esto es como la plataforma de software capaz de proveer los servicios de impresión 3D. Es conveniente aclarar que, una plataforma de software no es una aplicación, sino un concepto más amplio que incluye a ésta, como se describe a continuación.

XaaS

XaaS significa todo como servicio. Un servicio en la nube es cualquier recurso que se provee a través de internet.

Como puede verse en la Figura 2, los recursos ofrecidos como servicios en la nube son el software como servicio, la plataforma como servicio y la infraestructura como servicio. Ellos son los tres pilares sobre los que se asienta el modelo de servicios en la nube. (Techtarget, 2014)

Sus acrónimos son SaaS, PaaS e IaaS, y pueden ser referenciados de manera genérica como XaaS.

Figura 2: Esquema general de componentes de un XaaS. Fuente: https://azure.microsoft.com/es-es/overview/what-is-paas/

La infraestructura como servicio hace referencia a un tercero que proporciona una infraestructura de TI altamente automatizada y escalable (almacenamiento, alojamiento, computación, redes) y sólo aplica cargos por lo que se usa de ella.

Esto implica que, en lugar de poseer activos como licencias de software o servidores, las empresas pueden alquilar recursos de forma flexible de acuerdo con sus necesidades.

Ejemplos de estas empresas son AWS, Microsoft, Google o IBM.

La plataforma como servicio es posiblemente el más complejo de los tres modelos de nube. Al igual que el anterior, PaaS incluye infraestructura (servidores, almacenamiento y redes), pero también incluye middleware, herramientas de desarrollo, servicios de inteligencia empresarial, sistemas de administración de bases de datos, y otros. PaaS está diseñado para sustentar el ciclo de vida completo de las aplicaciones web: compilación, pruebas, implementación, administración y actualización.

Dado que esto podría ser middleware, bibliotecas, gestores de bases de datos, un sistema operativo y/o cualquier tipo de servidor, resulta evidente que hay sabores diferentes en lo que se ofrece. (Chavis, B. y Jones, T., 2015)

La idea detrás de una solución completa de PaaS, es que el proveedor proporciona servicios de software integrados basados en la nube que incluyen bases de datos preinstaladas y configuradas y software de middleware (como aplicaciones y servidores web), todo proporcionado con una base de suscripción rentable, optimizada y segura. En ella, los usuarios cliente puedan centrarse en instalar o crear y ejecutar aplicaciones, en lugar de construir y mantener la infraestructura y los servicios subyacentes. (Miller, 2017)

Entre los proveedores de PaaS, se incluyen Google App Engine, Oracle Cloud Platform, Pivotal’s Cloud Foundry y Heroku (de Salesforce).

El software como servicio corresponde a la provisión parcial o total del software necesario para satisfacer un requerimiento. Este software puede orientarse desde usos genéricos (ofimática) a específicos (diseño gráfico, conversión de formatos de archivo, y otros). Este software está alojado y/o administrado por un tercero y se puede acceder a través de la web, normalmente solo iniciando sesión. Generalmente se cobra por suscripción o por usuario.

SaaS se vuelve más es relevante en aplicaciones específicas. Ejemplos de ello son: el correo electrónico (Outlook de MS), la gestión de relaciones con el cliente (CRM de Salesforce), la gestión de documentos (Google Docs), el almacenamiento de archivos en la nube (Dropbox).

Todo XaaS funciona sobre la base de definir una API con la cual puedan interactuar cada uno de los actores del sistema.

En este caso se define una API siguiendo las recomendaciones REST.

API REST

REST es el acrónimo de “REpresentational State Transfer”. Se dice que “*REST es un estilo arquitectónico. Es un conjunto de restricciones, a respetar cuando se diseña la arquitectura de un servicio web*” (Amodeo, 2013).

Esta variante de arquitectura de software se apoya en el protocolo HTTP para funcionar. Lo cual significa que los servicios web implementados usan métodos HTTP para manipular recursos.

Las restricciones (también llamadas principios) propuestas en REST son las siguientes (Richardson, L. y otros, 2013):

- REST está orientado a recursos lógicos. No se representan acciones, sino entidades de negocio.
- Cada recurso expuesto posee un identificador único universal (UUID o GUID) bajo el cual es posible referenciarlo de manera unívoca, expresado como URI en formato de directorios.
- La implementación, y la forma de representar internamente un recurso es privada y no accesible desde el exterior.

- Cada recurso tiene una interfaz, o conjunto de operaciones que admite, seleccionadas de entre los métodos HTTP (GET, POST, PUT, DELETE, PATCH, HEAD. OPTIONS, CONNECT, etc.) de manera explícita, siguiendo el protocolo definido por los RFCs 2616 y 5789.
- La interfaz es homogénea para todos los recursos. Para cada recurso se deben escoger las operaciones que soporta.
- Las operaciones se realizan mediante la transferencia del estado de un recurso entre cliente y servidor.
- Las operaciones son stateless. Es decir, no se mantiene el estado por lo que el resultado de una operación es independiente de la conversación que haya existido anteriormente entre el cliente y el servidor.
- La información se transfiere como XML, JavaScript Object Notation (JSON), o ambos.
- Los recursos pueden ser multimedia (principio HATEOAS o Hypermedia As The Engine Of Application State - Hipermedia Como Motor del Estado de la Aplicación), por lo cual la interfaz debe tener la capacidad de proporcionar al cliente los datos o enlaces adecuados para ejecutar acciones concretas sobre dichos recursos.

Un principio de diseño básico en REST, es que identificando con claridad los recursos (end-points) a usar en una aplicación web, puede lograrse que el servicio resulte más útil y más fácil de desarrollar.

En general, cuando se define una API REST, lo que se desea exponer es una capa de lógica de aplicación. Esta API REST describe un conjunto de recursos y un conjunto de métodos con los que se puede acceder a dichos recursos.

Es habitual realizar el diseño con un enfoque orientado a datos, donde los servicios se definen en relación a las operaciones CRUD. Siendo CRUD el acrónimo de “Create Read Update Delete”, nombres de las operaciones habitualmente usadas en mantenimiento de datos, sobre tablas de un gestor relacional de base de datos. En este tipo de enfoque existen 2 tipos de recursos: las entidades y las colecciones.

En general, las equivalencias funcionales de los 5 métodos más usados en una petición HTTP (respetando las recomendaciones REST) son:

- GET: se considera equivalente de una función para recuperar un elemento del sistema de archivos y/o de una sentencia SQL de selección de datos.
- POST: se considera equivalente de una función de creación de un elemento en el servidor y/o de una sentencia SQL de inserción de datos.
- PUT: se considera equivalente de una función para cambiar el estado completo de un recurso, modificar un elemento del sistema de archivos y/o de una sentencia SQL de actualización de datos.
- PATCH: similar al anterior, pero opera de manera parcial sobre un recurso, por ejemplo modificar el estado de algunos atributos del mismo. Usado especialmente para proveer cada orden específica de control del hardware
- DELETE: se considera equivalente de una función para eliminar un elemento del sistema de archivos y/o de una sentencia SQL de borrado de datos.

Cuando los principios REST se usan para diseñar e implementar una aplicación web (o simplemente una API) orientada a proveer servicios, se dice que la misma es RESTful.

Un elemento importante en la implementación es la definición de los identificadores que se usen para localizar cada recurso, más precisamente de los URIs del servicio web.

Un URI (Uniform Resource Identifier) está dado por una cadena alfanumérica que permitir identificar de forma única cada recurso expuesto en la red.

Las URL, Uniform Resource Locator, son un tipo de URI, que además de identificarlo permite localizarlo, ya sea para poder acceder a él o compartir su ubicación.

En general, una URL se estructura de la siguiente forma:

{ protocolo }://{ dominio o hostname }[:puerto]/{ ruta del recurso }

La plataforma propuesta se enfoca la provisión de servicios de impresión 3D, por lo cual los end-point o recursos se definirán en este sentido.

LA IMPRESIÓN 3D

Como se dejó entrever en la introducción, la impresión 3D es un proceso complejo que involucra diversas fases, las que pueden observarse en la Figura 3.

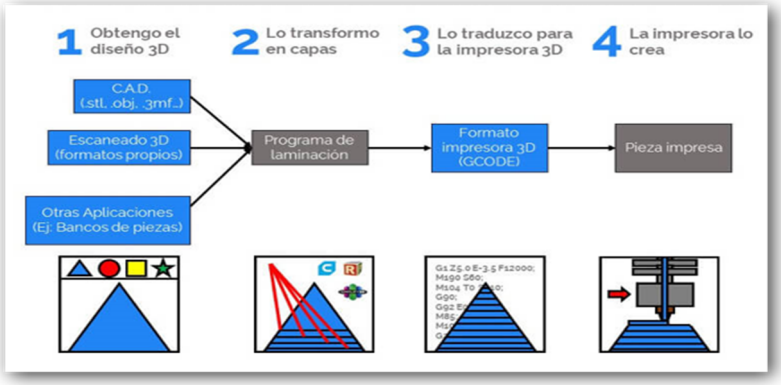

Figura 3: Esquema del proceso de impresión 3D.
Fuente: https://of3lia.com

Inicia en una computadora, particularmente en un archivo que contiene un modelo digital del objeto 3D que se desea fabricar. Este modelo digital puede obtenerse mediante un diseño CAD-3D, desde un archivo STL descargado desde la web o desde un escáner 3D.

Una vez obtenido el modelo digital, se deberá traducir a un formato particular llamado STL y luego verificar que el archivo se encuentre en condiciones correctas para su fabricación. Este formato traduce los modelos matemáticos provenientes de cualquier sistema CAD en una malla de polígonos que encierran un volumen.

Este archivo debe ser procesado por una herramienta software de laminado, con la finalidad de dividir al objeto 3D digital en un conjunto de capas. Luego y, en base a ciertos parámetros particulares, genera las instrucciones necesarias para que el equipo de impresión 3D pueda construir la pieza final, capa sobre capa.

Las “instrucciones” que genera este software se conocen como G-codes. El conjunto de instrucciones resultante puede encontrarse en un archivo o en memoria.

Finalmente se produce la impresión propiamente dicha y las tareas de acabado final.

COMANDOS G-CODE

La etapa final del trabajo realizado por el software corresponde a la secuencia de órdenes que producen los mini movimientos que el dispositivo final debe hacer para imprimir la pieza.

Esta tarea es realizada por el firmware a partir de conocer los comandos (o funciones preparatorias) específicos. Estas funciones son importantes, ya que controlan el modo en que la máquina va a realizar un trazado, o el

modo en que va a desplazarse sobre la superficie de la pieza que está trabajando.

Los comandos pueden seguir diferentes formatos. En el caso de las impresoras del proyecto RepRap siguen las reglas sintácticas de G-codes (códigos G):

Lnnn espacio **Lnnn** espacio **…….** espacio **Lnnn**

Es decir una función o instrucción G-code RepRap está dada por una lista de campos, separados por espacios en blanco.

Cada campo consiste de una letra seguida de un número o de una letra única (flag). Un campo puede interpretarse como un comando, un parámetro u otra información de propósito especial.

La letra indica la acción a realizar. En la Figura 4, se listan las acciones posibles más habituales.

Letra	Significado
Gnnn	Comando GCode estándar, como moverse hasta un punto
Mnnn	Comando definido por RepRap, como encender un ventilador
Tnnn	Seleccionar la herramienta nnn. En RepRap, las herramientas son extrusores
Snnn	Parámetro de comando, como la tensión enviada a un motor
Pnnn	Parámetro de comando, como el tiempo en milisegundos
Xnnn	Una coordenada X, normalmente para moverse a ella. Puede ser un número entero o racional.
Ynnn	Una coordenada Y, normalmente para moverse a ella. Puede ser un número entero o racional.
Znnn	Una coordenada Z, normalmente para moverse a ella. Puede ser un número entero o racional.
Innn	Parámetro - Actualmente no utilizado
Jnnn	Parámetro - Actualmente no utilizado
Fnnn	Feedrate en mm por minuto. (Velocidad de movimiento del cabezal de impresión)
Rnnn	Parámetro - usado para temperaturas

Figura 4: Fragmento de la tabla de códigos g-code/Reprap genéricos.
Fuente: https://reprap.org/wiki/G-code/es

El parámetro adicional, dependiendo del contexto, cambia su significado y también su tipo, ya que pueden ser números enteros (368) o reales (12.42).

Como ejemplo, se listan algunos comandos con su significado:

- M115: Obtener información sobre el firmware
- M114: Leer la posición actual
- G28: Hacer un 'homing'
- G1 X50: Ir a posición (50,0,0)
- G1 X50 F600: Ir a posición (50,0,0) con velocidad de 600mm/min (10mm/sec)

Estos comandos se envían a través del puerto serie de la computadora como texto ASCII, generalmente a la velocidad de 115200 baudios. Las unidades de desplazamiento están en milímetros (por defecto), y las de velocidad en mm/min (por defecto).

Los G-code permiten controlar la impresora desde programas alojados en otro equipo o dispositivo HMI conectado al puerto serie correspondiente y enviando códigos ingresados por teclado directamente.

Los principios para comunicar los códigos a la impresora son similares a los de comunicación de otros lenguajes. Los pasos son:

- Iniciar una comunicación serie a través de puerto que corresponda a una velocidad dada (por ejemplo, 115200 baudios).
- Recibir la respuesta "start" entregada por el firmware para indicar que está listo.
- Enviar los comandos que se desea sean ejecutados por la impresora.
- Cerrar la conexión

En las impresoras 3D de tipo RepRap (aunque no es una característica exclusiva) el firmware se encarga de:

- Interpretar los G-codes enviados y convertirlos a las acciones adecuadas en la parte electrónica de la impresora.

- Realizar interpolaciones entre posiciones
- Gestionar las temperaturas del extrusor y de la cama caliente si corresponde, mientras se construye la pieza.
- Manipular una pantalla LCD, desplegar un menú con las opciones principales y reaccionar a los eventos si los hubiera.
- Leer comandos G-code desde una tarjeta de memoria SD.

Estas secuencias de comandos en escenarios favorables y desfavorables de comportamiento, deben ser tenidas en cuenta al momento de diseñar los servicios de control que serán provistos por la plataforma.

SISTEMA EMBEBIDO Y SBC

Una SBC (Single Board Computer) es una PC de placa única. Esto quiere decir que a diferencia de las computadoras tradicionales, las SBC son placas electrónicas que contienen todos (o la mayor parte de los componentes) de una computadora.

Las SBC (Single Board Computer) son PCs de placa única. Esto quiere decir que a diferencia de las computadoras tradicionales, las SBC son placas electrónicas que contienen todos (o la mayor parte de los componentes) de una computadora.

Una SBC se destaca por su tamaño reducido (actualmente de 8% a 20% de un pc mini-ITX), por su bajo precio (promedio menor a 100U$S en el mercado internacional) comparado con una PC (promedio mayor a 500 U$S en el mismo mercado) y por ofrecer poca potencia de cómputo (aunque esto es relativo en el ámbito de la ofimática y de IoT).

Una de las particularidades del trabajo con placas SBC es que requieren manipular sistemas operativos y otras aplicaciones, generalmente embebidos en ellas.

SERVIDOR WSGI

Si bien se justifica su uso más adelante, el lenguaje elegido para el desarrollo de la aplicación web es Python.

La Python Software Foundation ha definido un estándar, denominado WSGI (Web Server Gateway Interface) que describe cómo un servidor web se comunica con una aplicación web.

WSGI es similar a la especificación Java Servlet o ASP/ASP.NET, basadas en el estándar CGI, aunque más simple y con un enfoque pitónico para hacerla reentrante, persistente, etc.

WSGI es una interface simple y universal entre los servidores web y las aplicaciones web o frameworks implementados en Python y definida en Python Enhancement Proposal 333 (Eby, 2003).

Describe cómo un servidor web se comunica con las aplicaciones web realizadas en Python, o que se ejecutan dentro de un servidor haciendo uso de un intérprete Python. Describe cómo manejar una solicitud, y cómo responder a la solicitud.

Desde un punto de vista operativo, para el caso de interactuar con un servidor web no Python, puede verse como aquel protocolo que permite comunicar 2 servidores.

La API establece que, del lado de la aplicación, se especifica un punto de entrada (objeto, método, función), con dos parámetros: la variable de entorno (environ) y la función para iniciar la respuesta start_response. Esta última encargada a su vez, de transferir el estado y los encabezados, y devolver un iterable con los datos a ser enviados al cliente.

Del lado del servidor, se invoca la aplicación por cada pedido que recibe del cliente HTTP, con las variables de entorno establecidas (estilo CGI)

Como se observa en la Figura 5, un servidor web compatible con WSGI sólo recibe una solicitud y la pasa a un objeto de la aplicación. Finalmente, envía la respuesta del objeto al navegador.

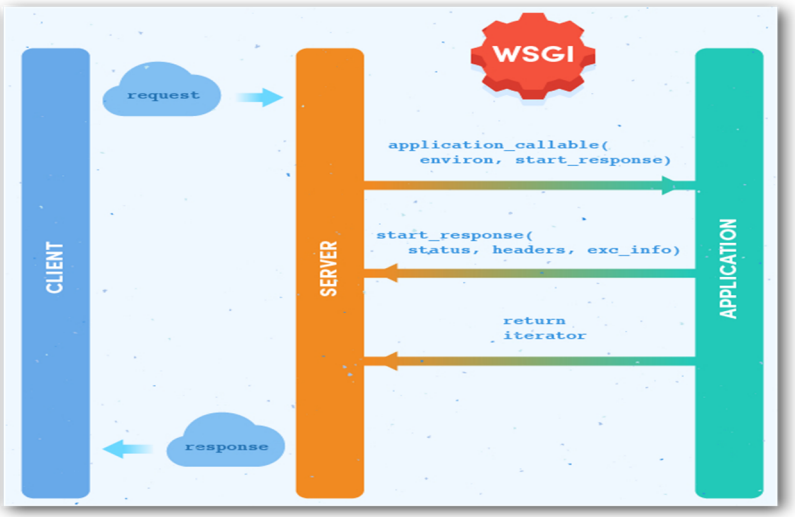

Figura 5: Esquema de mensajes usando protocolo WSGI.
Fuente: uploads.toptal.io/blog/

Desde la óptica del desarrollador, WSGI es muy simple. Posee 4 características básicas:

- Las aplicaciones WSGI son objetos python invocables (funciones o clases) con un método __call__ donde se transmiten dos argumentos: un entorno WSGI como primer argumento y una función que inicia la respuesta.
- La aplicación debe iniciar una respuesta utilizando la función proporcionada y devolver un iterable donde cada elemento producido debe poder ser escrito y liberado.
- El entorno WSGI es como un entorno CGI sólo con algunas características adicionales que son proporcionadas por el servidor o un middleware.

- Es posible agregar funcionalidad a una aplicación existente de una manera general envolviendo la misma en otra aplicación compatible con WSGI.

Si bien la opción sugerida para desarrollos en Python es usar servidores Web que operen bajo estrategia WSGI, no todos los servidores admiten WSGI de manera directa. Esto ocurre incluso dentro de los servidores web basados en Python.

En estos casos se necesita, entonces, de un puente entre el servidor y el programa correspondiente ejecutado por el intérprete. Estas interfaces, o protocolos, definen cómo las aplicaciones web interactúan con el servidor web.

En la práctica se dispone de diversas interfaces para interactuar con python, que en orden creciente de complejidad y prestaciones, son: CGI, mod_python, FastCGI y SCGI. Cualquiera de estas interfaces, propone una API de bajo nivel, que si bien permiten a un programa comunicarse con el servidor, requieren de considerar algunos detalles de diseño o integración, durante la fase de programación o de mantenimiento posterior. (Fernández Montoro, 2012) (van Rossum, G. y Kubica, M., 2010)

La mejor alternativa está dada por el uso de los servidores WSGI (como Twisted, uWSGI, Waitress WSGI, Aspen, etc.) o frameworks que implementen WSGI (Dyango, Tornado, Flask, Grok, Flup, FlipFlop o Bottle entre otros). Cabe aclarar que algunos de los ejemplos de framework listados presentan un módulo que actúa como servidor WSGI y puede ser usado de manera independiente.

Cuando los servidores WSGI son usados de manera colaborativa con otro servidor web, funcionan solamente como servidores de interfaz WSGI.

PATRONES DE DISEÑO WEB HABITUALES

Existen diferentes patrones de diseño de software que pueden ser aplicados a desarrollos web, como MVC, MTV, Adapter, Facade, FrontController entre los más populares. Su elección depende en general, de las características del problema a resolver.

Se describen, a continuación, los 2 patrones tomados como referencia para este trabajo.

El primero es el patrón MVC, compuesto de 3 componentes o capas con las siguientes consideraciones básicas:

- El Modelo: ligado a los datos que se mostrarán y/o modificarán. A menudo está representado por las clases usadas para mapear el modelo relacional a objetos.
- La Vista: destinado a mostrar los datos del modelo al usuario final. Normalmente, este componente se implementa a través de plantillas.
- El Controlador: cumple la función mediar entre el usuario y el modelo. El código del controlador responde a las acciones del usuario (como invocar alguna URL específica), le dice al modelo que modifique los datos si es necesario, y le dice al código de la vista qué mostrar.

Se puede observar lo mencionado en la siguiente representación:

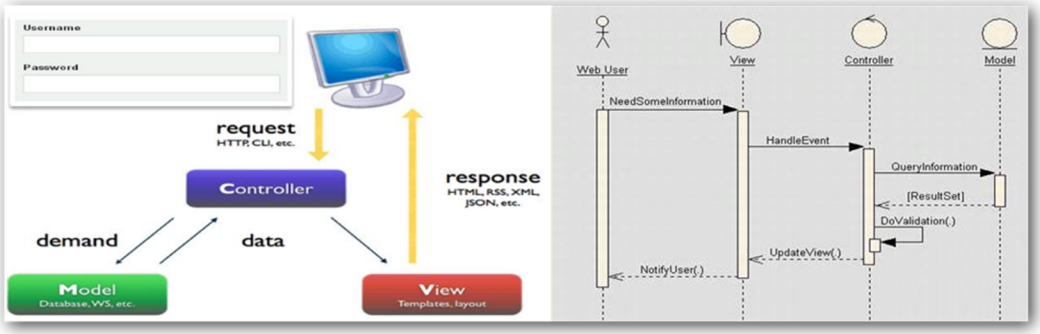

Figura 6: Patrón de diseño MVC.
Fuente: ramj2ee.blogspot.com/2013/08/model-view-controller-design-pattern.html

En los diseños de aplicaciones Web habituales, en general se dispone de tantos controladores como elementos de modelo y de vista se vayan a manipular. Adicionalmente, con el uso de controladores diferentes según se trate de información de una entidad o de una colección de ellas.

En el caso de una API, y particularmente las que respetan los principios REST, el esquema suele simplificarse levemente, ya que el contexto implica proveer de una interfaz para otra aplicación Web. Esto significa que no se trata de una aplicación para usuario final y que no manipula (ni renderiza) en principio ninguna vista, sólo devuelve entidades de modelo (unitaria o colección) representadas en uno o más formatos específicos (JSON, XML, otros).

Podría decirse que en una API REST los componentes administran Representaciones de Datos, como si se tratase de Vistas Lógicas, que serán usadas posteriormente en una Vista de usuario final.

El segundo patrón a describir es el Controlador Frontal. Su necesidad surge de una característica de las aplicaciones desarrolladas en Python.

En este lenguaje, la mayoría de las aplicaciones web tienen un único punto de entrada (un sólo controlador), que funciona como "despachador". Esto es, dependiendo de la URL solicitada, se invoca a una función u otra para atender la petición.

En una estructura de patrón FrontController, aparece una clase especializada encargada no sólo de realizar tareas de validación y control de acceso sino también servir de interfaz a cada petición que se realice. Esta clase auxiliada por otra, denominada despachante, encargada de redirigir el flujo al controlador específico capaz de resolver la petición realizada.

Un contexto típico, se presenta en el siguiente esquema:

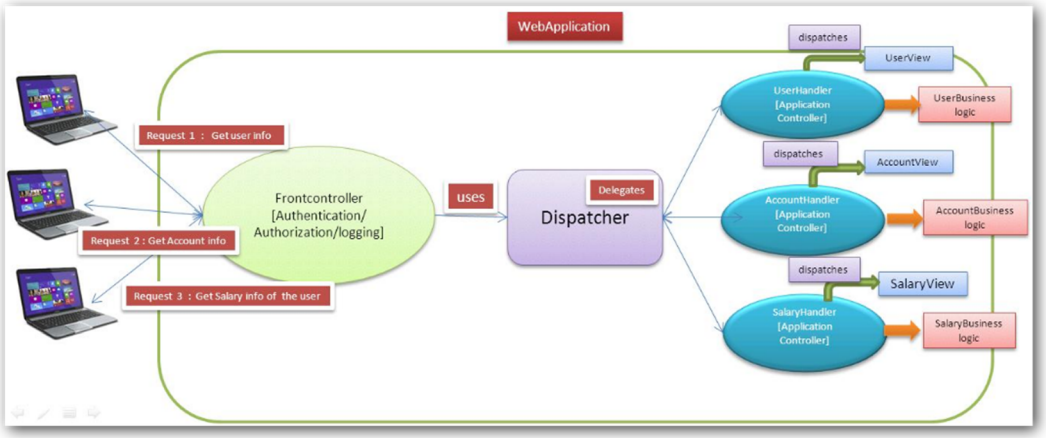

Figura 7: Patrón de diseño Front Controller.
Fuente: https://ramj2ee.blogspot.com/2013/08/front-controller-design-pattern.html

Se puede observar que hay un controlador (o subcontrolador) por cada vista específica que se deba desplegar al usuario

Para el caso del desarrollo de una API REST, caben las mismas consideraciones que las realizadas anteriormente acerca de las vistas.

Una de las características de este tipo de desarrollos, en parte dado por los protocolos usados y en parte por el tipo de lenguajes usados en el cliente, se hace necesario definir algún nivel de seguridad que permitan verificar la identidad de los usuarios y validar las diferentes peticiones. Por esta razón se adopta el uso de JWT.

AUTENTICACIÓN Y AUTORIZACIÓN MEDIANTE JWT

JSON Web Token (JWT) es un estándar abierto definido en la RFC 7519. (Jones, M.B.; Bradley, J. y Sakimura, N., 2013)

Define un modo compacto y autónomo para transmitir de forma segura la información entre las partes como un objeto JSON. Esta información puede ser verificada y es confiable porque está firmada digitalmente. Los JWT se pueden firmar usando un código secreto (con el algoritmo HMAC) o utilizando un par de claves públicas / privadas usando RSA.

El esquema de la Figura 8 muestra el flujo habitual de una aplicación asegurada usando JWT.

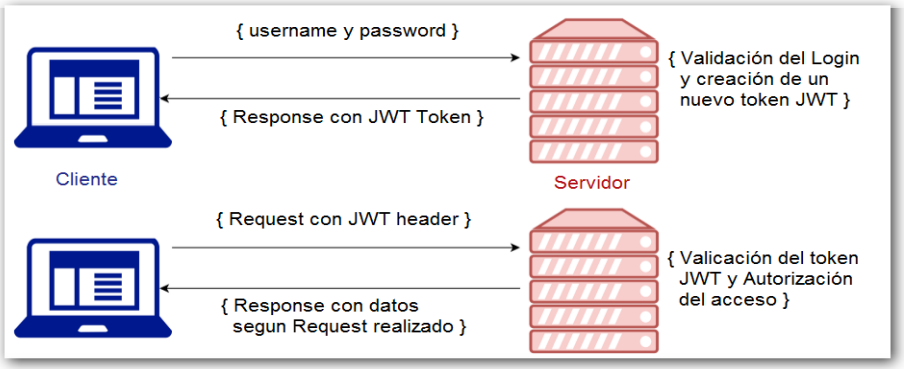

Figura 8: Secuencias de autenticación.
Fuente: medium.com/dev-bits/

Es muy similar al flujo seguido cuando se usa autenticación vía cookies. La mayor ventaja de los tokens sobre ellas es que no tiene estado (stateless). El backend no necesita mantener un registro de los tokens. Cada token es compacto y contiene todos los datos necesarios para comprobar su validez, así como la información del usuario para las diferentes peticiones.

Al usar tokens, la primera tarea del servidor consiste en firmar cada token al iniciar la sesión.

Una vez que el cliente cuenta con el token provisto, se lo debe adjuntar a cada petición.

Finalmente, el único trabajo del servidor es verificar que los tokens intercambiados sean válidos y autorizar la petición. Esta característica favorece la escalabilidad y la interconexión entre diferentes servidores de forma casi autónoma.

La autorización se realiza mediante un algoritmo responsable que verifica la cabecera en busca de un token. En caso de hallar algo, se verifica el token y finalmente se extrae la información del mismo para establecer la identidad del usuario dentro del contexto de seguridad de la aplicación. No

se requieren accesos adicionales a bases de datos ya que al estar firmado digitalmente si hay alguna alteración en el token se corrompe.

El formato indicado por el estándar RFC 7519 responde a lo representado en la Figura 9.

Figura 9: Estructura de un JWT.
Fuente: netlogica.it

En el cual, la cabecera (header) indica el tipo de token y algoritmo de codificación usado. El cuerpo (payload) contiene los datos útiles para identificar el usuario, para controlar la vida del token y los requeridos por la API específica. Finalmente la firma (signature), para verificar que el token es válido, encriptada generalmente en base a una palabra clave (secret).

Dos ventajas de este mecanismo son la menor sobrecarga en la cantidad de peticiones a la base de datos y un acoplamiento más bajo entre aplicaciones.

III. DESARROLLO Y RESULTADOS

El diseño e implementación de una plataforma como la propuesta debe tener la capacidad de brindar una capa middleware con el potencial de permitir a diferentes usuarios aprovechar y eventualmente compartir recursos de software y de hardware preexistentes. Dicho middleware está dado por la API REST que aquella define.

A los efectos de hacer visibles los alcances mínimos de la API se plantean 3 escenarios posibles de utilización de la plataforma.

ESCENARIO DE APLICACIÓN A

Un propietario de una o más impresoras 3D instaladas de manera domiciliaria y local, desea monitorear y controlar sus trabajos de impresión de manera remota usando internet.

A pesar de ser el caso más simple es el más común de encontrar, cuando se buscan referencias en la web o se leen discusiones en los foros especializados.

Es frecuente también encontrar, referencias de usuarios hobbistas que poseen más de una impresora en su hogar.

En la Figura 10, se muestra un posible esquema de la instalación usando la API REST propuesta:

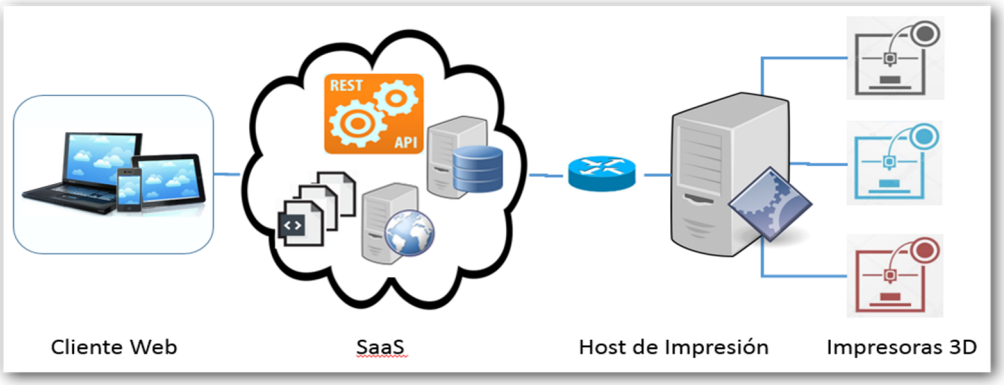

Figura 10: Caso de uso A para la plataforma de impresión 3D.
Fuente: elaboración propia.

Aquí, cada impresora está conectada físicamente a un puerto serie diferente de un mismo equipo servidor, en el cual se ejecutan N instancias (uno por impresora) de la aplicación con las funciones de Host de Impresión (printrun en este caso). Cada instancia se encuentra vinculada a un puerto TCP diferente del equipo. Todas las impresoras poseen la misma dirección IP, la cual corresponde al equipo servidor.

En este contexto, la API RESTful podría también ejecutarse en el mismo equipo, recibiendo las peticiones desde los clientes remotos y derivando la actividad al host de impresión que corresponda.

ESCENARIO DE APLICACIÓN B

Una empresa X, con locales distribuidos, desea prestar servicios de impresión 3D, manteniendo un registro centralizado y políticas comerciales unificadas para todos sus locales.

Un ejemplo posible es el de impresoras 3D para la elaboración de productos de chocolate o decoración de tortas, distribuidas en diferentes locales comerciales.

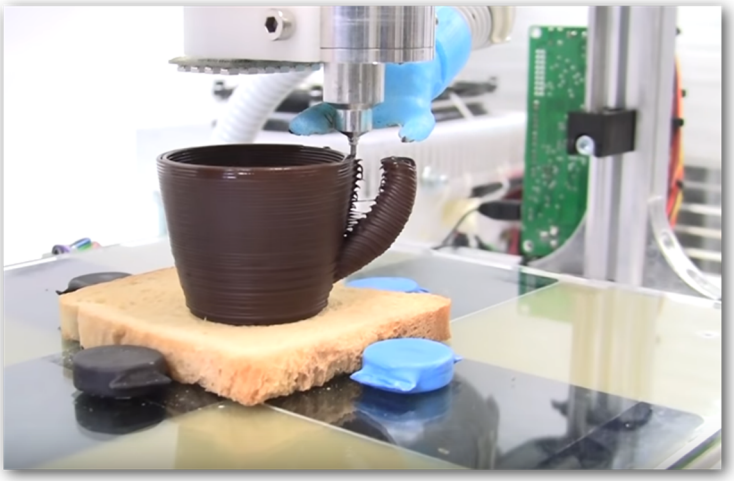

Figura 11: Impresora de chocolate.
Fuente: https://www.youtube.com/watch?v=MKQlys-z7SM

Figura 12: Autoservicio de torta decorada.
Fuente: https://gaia.adage.com

Este escenario también es aplicable al caso del usuario industrial con múltiples impresoras 3D instaladas en un único espacio de trabajo. Como ocurre en la granja de impresoras 3D de Prusa Research.

Se aclara que no ha sido posible obtener información acerca de la forma específica en que se realiza el monitoreo y control en las instalaciones de Joseph Prusa.

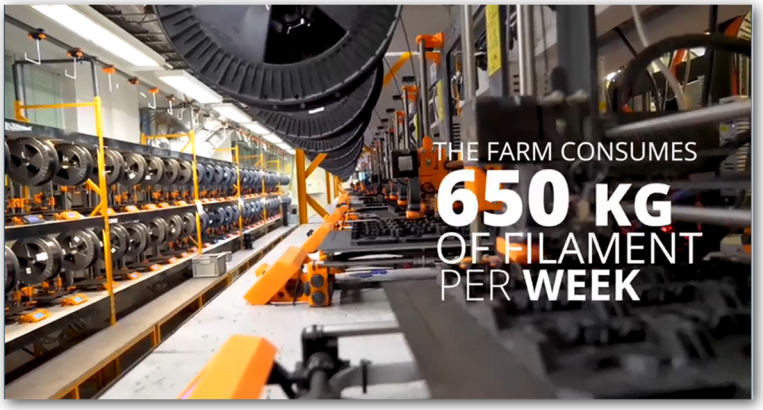

Figura 13: Printing Farm de Prusa Research, USA.
Fuente: https://www.prusaprinters.org/a-quick-look-to-our-printing-farm/

En la Figura 14, se presenta un esquema genérico de la infraestructura del sistema para este escenario, cuando se use la API REST propuesta:

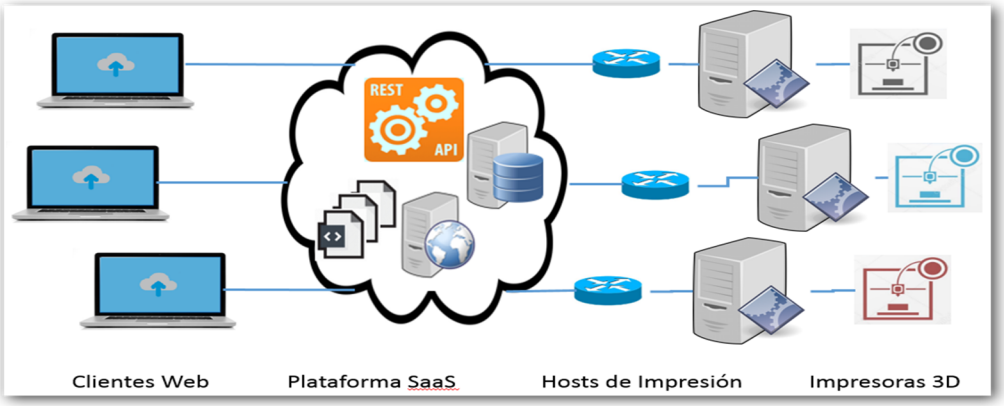

Figura 14: Caso de uso B para la plataforma de impresión 3D.
Fuente: elaboración propia.

En la misma se distinguen, cada impresora conectada un mismo equipo servidor, el cual ejecuta 1 instancia de la aplicación con las funciones de Host de Impresión. Dicha aplicación se encuentra vinculada a un puerto TCP del equipo. Cada impresora posee una dirección IP diferente, que se corresponde con la del equipo de su Host de impresión.

En el contexto dado, posiblemente exista un dispositivo cliente localmente cercano a cada impresora y la API RESTful se ejecute en un servidor

dedicado, para recibir las peticiones desde cada uno los clientes remotos y derivar la actividad al equipo y host de impresión que corresponda.

ESCENARIO DE APLICACIÓN C

Un conjunto amplio de usuarios de impresoras 3D, desean formar parte de una red social ya sea con carácter social o comercial, donde se requiera proveer y/o consumir servicios de impresión 3D.

Algunos sistemas se van orientando en este sentido como OctoPrint, 3DPrinterOS o Astroprint.

Esta última, por ejemplo, es una alternativa provista por 3DaGoGo Inc., empresa privada de tecnología ubicada en San Diego, USA. La misma es ofrecida al consumidor como una plataforma en la nube y con un mercado de aplicaciones diseñado para la impresión 3D.

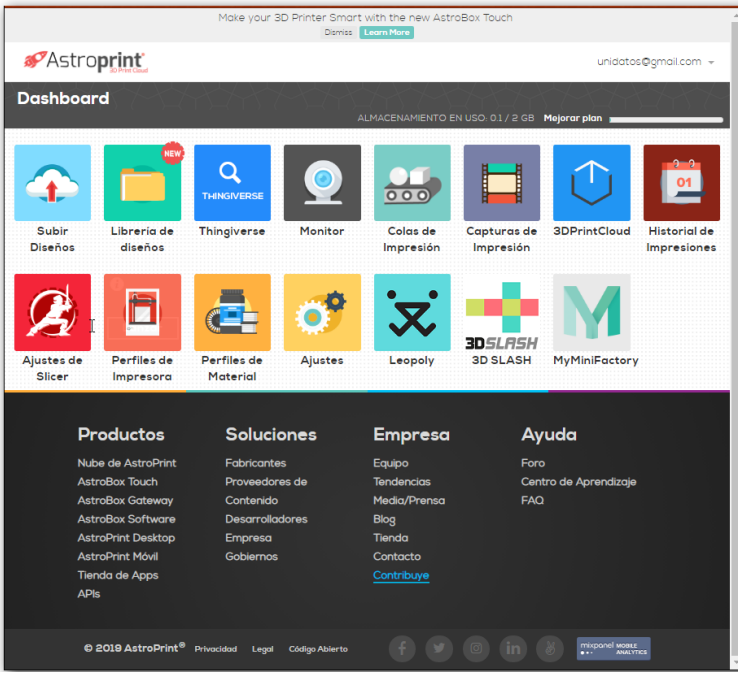

Figura 15: Dashboard de la interfaz de acceso al cloud de AstroPrint. Fuente: https://cloud.astroprint.com/

En la siguiente figura se muestran posibles componentes del sistema, resultantes de ampliar los servicios de la API REST propuesta en este trabajo:

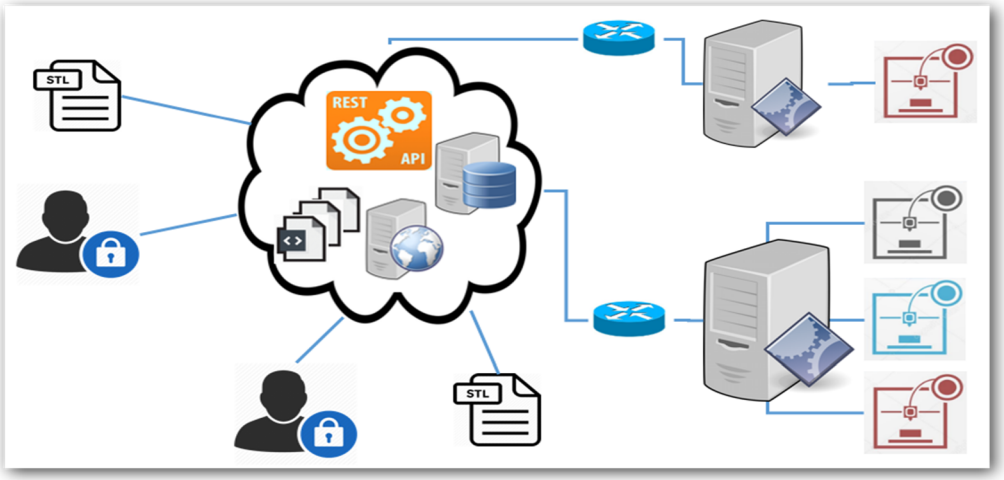

Figura 16: Caso de uso C para la plataforma de impresión 3D.
Fuente: elaboración propia.

En este caso, las impresoras se encuentran conectadas y configuradas en combinaciones de los escenarios anteriores.

La gran diferencia está dada por las características de las interacciones entre usuarios y recursos de impresión.

En este escenario, hay proveedores y consumidores. Tanto de servicios de impresión como de archivos de diseño o de fabricación de piezas 3D. Un usuario puede monitorear y/o controlar las impresiones en una impresora propia o en la impresora de otro usuario, siempre y cuando satisfaga ciertas restricciones de rol, aportes económicos o relaciones contractuales. Puede también, existir la integración con otras plataformas, ya sean redes sociales o comerciales.

OBSERVACIONES

En todos los escenarios se detecta la existencia de usuarios, lo que eventualmente podrían tener roles de acceso y utilización diferenciados con mayor o menor complejidad según el caso.

Comparando los diferentes esquemas, se observa la existencia de un requerimiento fundamental para cada impresora 3D que vaya a administrarse a través de la API propuesta. Es el hecho de su conectividad a internet, ya sea de manera directa o indirecta a través de un dispositivo que cumpla funciones de interfaz, como son los casos de una PC, notebook o SBC. Por lo cual, resulta fundamental conocer los parámetros de conexión a la red.

En general, las plataformas existentes mencionadas, operan con aplicaciones propietarias tanto del lado cliente como servidor y la mayoría no ha documentado de manera específica su API. Sólo unas pocas son de código abierto.

Las diferencias fundamentales entre la plataforma propuesta en este trabajo y ellas son, que aquí:

a) se ha integrado una aplicación host de impresión de terceros preexistente, de código fuente abierto y que aún no se ha orientado a brindar servicios remotos,
b) se ha hecho pública la especificación de la API REST, pudiendo extenderse según las necesidades particulares,
c) queda abierta la implementación, para que puedan desarrollarse aplicaciones clientes capaces de consumir los servicios descritos.

DEFINICIÓN DE REQUISITOS

Este trabajo se orienta a una solución tecnológica específica. Para el prototipo se fijaron los siguientes requerimientos:

- La cantidad de usuarios es variable, aunque en general se trata de un único usuario hobbista que posee una impresora 3D domiciliaria. La simultaneidad sobre una misma impresora limita a 1 (uno) el usuario de monitoreo y control.
- La capacidad de almacenamiento inicial de datos propia del proceso es baja. Aún en el caso de impresoras de ámbitos comerciales.
- Las aplicaciones a usar se orientan a prestar servicios remotos de:
    - Validar el acceso de un usuario de la plataforma
    - Transferencia de archivos de diseño 3D (tipo STL o gcode)
    - Selección de una impresora remota y conexión a la misma
    - Control de una impresora 3D. El usuario de la impresora debe poder realizar operaciones fundamentales como iniciar o parar. Eventualmente, modificar los parámetros de operación
    - Monitorear de manera remota una impresora 3D, especialmente los valores de los parámetros significativos, durante el proceso de impresión.
    - Almacenamiento de los datos básicos de cada pieza impresa.
- El usuario de administración del host no requiere de aplicaciones backend específicas. Se asume que puede operar directamente sobre el sistema.
- No se necesita conectar directamente dispositivos HMI gráficos.
- No se requiere que las aplicaciones realicen manipulación de datos multimedia. Esto significa que, si bien en el sistema podría realizar

captura de video a través de cámaras o sonido mediante micrófonos o también utilizarse el reconocimiento de voz para la entrada de órdenes, el dispositivo usado como Host no se encarga de realizar el procesamiento de las señales, lo cual queda delegado a otra/s placa/s específica/s según el caso.

- No hay exigencias estrictas respecto al tiempo de respuesta en la red, aunque dadas las características del proceso se esperan retardos menores a 2 segundos.
- El costo y el tamaño de los dispositivos que conforman el sistema debe mantenerse reducido dado que el destino final de este trabajo no es comercial ni industrial.
- No se requiere que las aplicaciones desarrolladas manipulen formatos de manera directa para obtener archivos en formato STL o gcode, sino solamente hacer uso de otras herramientas ya construidas.
- En principio, no se realizan tareas de cálculo que exijan fuertemente al procesador. En caso que ocurra, se analizarán y definirán requerimientos adicionales.
- Debe independizarse a la impresora de una notebook o PC manteniendo la conectividad a la red de datos.

SELECCIÓN DE LA IMPRESORA 3D

La impresora sobre la cual se realizarán las pruebas, responde a uno de los modelos del proyecto Reprap, una Prusa I3 Hephestos modificada, con extrusor directo tipo Gregwade y hotend Argento.

El procedimiento de impresión usado es FDM. La misma opera mediante extrusión en caliente de un filamento de plástico. Se basa en empujar un hilo de plástico a través de un dispositivo, denominado extrusor, que se

calienta hasta una temperatura capaz de fundir ligeramente el material de plástico utilizado

Para ello, la impresora realiza la extrusión en caliente de filamento de plástico, ABS (aprox. 220ºC) o PLA (aprox. 170ºC), sin llegar a derretirlo por completo y deposición del material expulsado en la punta del dispositivo, sobre una base, denominada cama caliente (aprox. 110ºC para ABS y 60ºC para PLA), para que el hilo extruido vaya quedando pegado.

Resulta entonces, que la solución consiste en que el cabezal se desplace en un plano X-Y mientras que la base (donde se deposita el material) se desplace en el eje Z, como se muestra en la Figura 17.

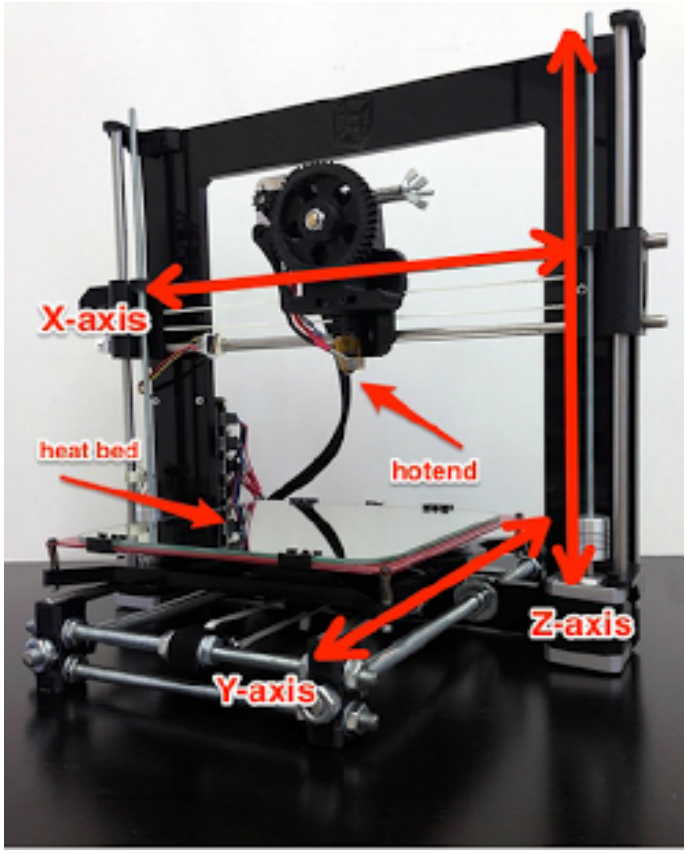

Figura 17: Impresora 3D del tipo Reprap Prusa i3.
Fuente: http://ingenio-triana.blogspot.com/p/impresora-3d.html

En la misma se observa, que el cabezal se mueve sobre un eje denominado X, el que a su vez se desplaza en un segundo eje perpendicular, denominado Z.

Todo el conjunto anterior ubicado sobre una base horizontal o cama (heat bead) que da soporte a la pieza en construcción.

Esta cama tiene la posibilidad de moverse en ambos sentidos de un tercer eje perpendicular a los 2 primeros, denominado Y.

Para llevar adelante sus movimientos, la impresora 3D está provista de un número de componentes electromecánicos, fundamentalmente motores.

Además de ellos, la impresora presenta también otros componentes electrónicos y electromecánicos tales como termistores, ventiladores, sensores de fin de carrera, relés y resistencias.

En una impresora basada en Arduino es habitual usar un módulo (escudo) denominado RAMPS (RepRap Arduino Mega Pololu Shield), encargado de concentrar en él la mayor cantidad de conexiones y de controlar electrónicamente a los componentes antes mencionados.

El control directo mediante señales eléctricas de la impresora lo realiza el módulo anterior, que recibe las órdenes de otro componente electrónico denominado placa controladora.

En el caso particular de la impresora elegida, el software de gestión primaria se aloja, a modo de firmware, en una placa Arduino MEGA 2560.

Entre las alternativas habituales para este software, se encuentran los productos Sprinter, sjfw, Marlin, Repetier Firmware, Aprinter.

SELECCIÓN DEL SOFTWARE DE CONTROL

Un requisito fundamental para que poder cumplir las tareas de impresión es contar con una aplicación de control y una aplicación de laminado. Para la impresora de pruebas de este trabajo se dispone de varias alternativas.

La intención aquí es ofrecer el mejor aporte a la comunidad global, por lo que se analizarán las ventajas y desventajas de cada una.

Tabla 1. Aplicaciones de host de impresión disponibles.
Fuente: elaboración propia.

Aplicación	Código fuente disponible	Lenguaje	Orientado a Web
Marlin Hephestos (1.1.4)	Sí	C++	No
Repetier Firmware (0.92)	Sí	C++	No
SPrinter (0023)	Sí	C++	No
Cura (15.06b)	Sí	Python	No
Slic3r (1.2.9)	Sí	C++ / perl	No
Skeinforge	Sí	Python	No
Astrobox	Sí	Python	Sí
MatterControl (1.7.2) (desde 2016)	Sí	C#	No
OctoPrint (1.3.4)	Sí	Python	Sí
Printrun: Pronterface + Pronsole + Printcore (desde 20140406)	Sí	Python	No
Repetier Host (2.0.1)	Sí (v0.90)	C#	No
Repetier Server (0.85)	Sí (v0.24)	C++	Sí
ReplicatorG (desde 2013)	Sí	Java	No
Visprinter (suspendido en 2013)	Sí	Python, WebGL	Sí

Los 3 primeros productos encaran la programación de bajo nivel, el firmware, directamente vinculado al control del hardware de la impresora. Básicamente, son los encargados de interpretar las órdenes mediante G-codes que son enviadas desde el Host.

Los siguientes 3 productos se orientan al trabajo de laminado.

Las aplicaciones restantes de la lista, presentan la funcionalidad principal como Host de Impresión, y eventualmente incorporan alguno de los 2 anteriores (o ambos) tipos.

Dado que uno de los objetivos de este proyecto plantea obtener una capa software que permita la interacción con otras aplicaciones existentes, la elección del lenguaje está fuertemente vinculada a la/s aplicación/es que se elija/n.

Algunas de las listadas en la tabla anterior ya poseen alguna versión con interfaz orientada a la web (Astrobox, Octoprint, Repetier y Visprinter). De las 3 restantes, ReplicatorG es una aplicación prácticamente abandonada y con una comunidad muy pequeña.

Quedando como opciones de base para este trabajo MatterControl en C# y Printrun en Python.

Desde el punto de vista del lenguaje para desarrollar una API Rest se tienen, preferentemente PHP, javascript, Java o Python, y en un segundo plano C++ o Perl.

Desde el punto de vista de la potencial necesidad de interactuar con el firmware o con otras aplicaciones, se consideran más apropiados los lenguajes de propósito general como C++, Python o Perl. Mientras que, desde el punto de vista de la integración con la plataforma de trabajo, los lenguajes más convenientes resultan Perl o Python.

Se concluye, entonces, que la aplicación elegida es Printrun (particularmente en sus versiones de núcleo y de consola) y que el lenguaje para desarrollar la API Rest es Python.

Un detalle a favor de Printrun y de la elección realizada, es que se trata de una aplicación levemente más robusta y confiable que MatterControl, porque esta última recién ha estado disponible en versión normal para Linux desde noviembre de 2016, si bien la versión beta fue publicada en el año 2015. Mientras que Printrun posee versión estable desde el año 2014.

Las 2 aplicaciones elegidas pueden correr tanto bajo cualquiera de los sistemas operativos habituales (MS Windows, GNU/Linux y Mac OS).

SELECCIÓN DE HARDWARE PARA EL HOST PRINCIPAL

Para dar soporte a las aplicaciones del servidor, se tienen 3 alternativas generales de hardware:

- un equipo orientado a servidor de red,
- una PC estándar de escritorio,
- una placa SBC.

Dado los requerimientos básicos de la plataforma de pruebas, el menor precio comparado con una PC y su menor tamaño, hacen a las placas SBC aplicables al problema en estudio, por lo cual se opta por la tercera opción.

Se suma a lo anterior, la tendencia creciente en el uso de SBC para dar soluciones de IoT.

Desde el punto de vista de la oferta global, existen muchas opciones de SBC (>30). El análisis comparativo se ha hecho sobre 5 modelos disponibles en la oferta frecuente estándar de Mercadolibre en el medio local y considerando la gama de productos intermedios cuando un mismo

dispositivo presenta varios modelos (como son los casos de Raspberry e Intel).

Tabla 2. Comparativa de placas SBC disponibles en el mercado. Fuente: elaboración propia.

	Raspberry Pi 2	**Beagle Bone**	**Intel Galileo Gen 2**	**ASUS Tinker board**	**Arduino Yun**
Procesador	CPU 900 MHz + GPU	CPU 1GHz + GPU	CPU 400 MHz	CPU 600 MHz + GPU	CPU 400 MHz
Superficie ocupada	50 cm^2	50 cm^2	85 cm^2	50 cm^2	35 cm^2
SO soportado	Linux Debian (Raspbian) MS Windows	Linux	Linux Debian (Yocto) MS Windows	Linux Debian (KODI)	Linux OpenWrt (Linino)
Redes	Ethernet 10/100, Wifi	Ethernet 10/100	Ethernet 10/100	Ethernet GBit, Wifi	Ethernet 10/100, Wifi
Video/Audio directo	Sí	Sí	No	Sí	No
Boot	SD	Flash / Flash + SD	Flash / Flash + SD	RAM / RAM + SD	Flash / Flash + SD
Memoria	1 GB RAM + SD	4 GB Flash + SD	8 MB Flash + 256 MB RAM + SD	2 GB RAM	16 MB Flash + 64 MB RAM + SD

Las placas deseables en base a su capacidad de memoria son la Beaglebone y Asus Tinker, mientras que a su capacidad de procesamiento son la Raspberry y la Beaglebone.

Sin embargo, cualquiera de las placas listadas puede satisfacer los requerimientos básicos del host de este trabajo, ya que el factor de video/audio no es importante.

Dado que una de las premisas es realizar el menor desembolso de dinero en la elaboración del prototipo del sistema, se toma la decisión de adoptar una placa Intel Galileo Gen 2, usada anteriormente en un proyecto de automatización industrial.

En las explicaciones que Intel ha dado para estas placas (en cualquiera de sus modelos: Galileo, Edison, Joule o Curie) se ha resaltado el hecho de haber sido pensadas para su utilización en proyectos de IoT. Hay muchos informes de revisión técnica realizados sobre la misma. En uno de ellos, Nayyar afirma que “Intel Galileo se puede utilizar para hacer diferentes tipos de proyectos en términos de Domótica inteligente, Robótica, Drones e incluso se puede utilizar para la implementación exitosa de proyectos de Smart Cities” (Nayyar, A., y Puri, E., 2016).

Al buscar y analizar proyectos que se encuentren terminados, estén disponibles en Internet o en libros publicados, que tengan relación directa con el objetivo de este trabajo y que usen la SBC Galileo (ya sea en su versión 1 ó 2), se detectan sólo algunos: un desarrollo de firmware para control limitado y directo de una impresora 3D (Casper, 2015) y el de control de un CNC (Scull, 2016). En ambos, el software está parcialmente terminado y usan MS Windows como sistema operativo. Un tercer proyecto sigue una línea conceptual similar a este trabajo al usar una SBC Intel Edison para controlar los procesos de impresión 3D (Elgert, M.; Djedid, B.; Elgert, J. y Lindvall, K. , 2016).

La Galileo es una placa de desarrollo, basada en el microprocesador Intel Quark SoC X1000. Esta CPU, de 32 bits, con un solo núcleo funcionando a velocidades de hasta 400 MHz, posee el mismo conjunto de instrucciones que el de los microprocesadores Intel Pentium P54C/i586. (Intel, 2014)

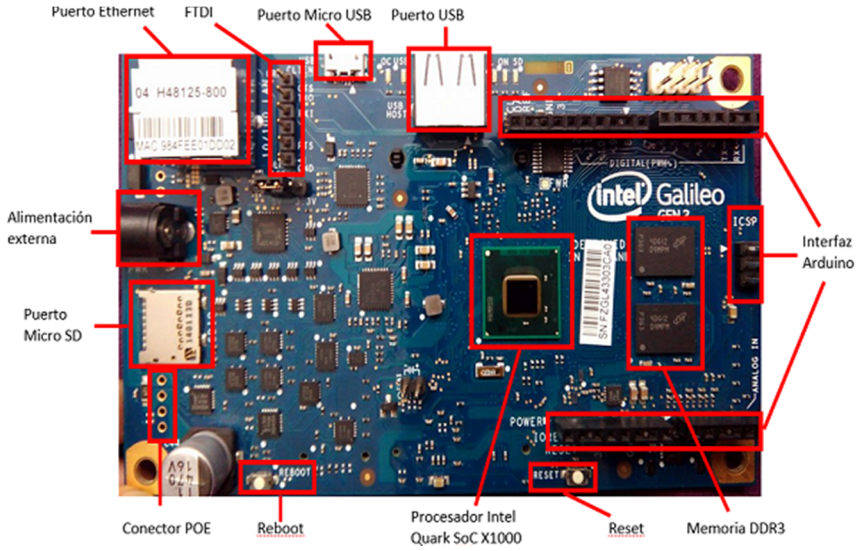

Figura 18: Intel Galileo Gen 2.
Fuente: teslabem.com/nivel-intermedio/intel-galileo-gen-2-ideal-cualquier-proyecto/

La placa ofrece compatibilidad con diferentes interfaces de E/S estándar en la industria, a través de una ranura mini-PCI Express, un puerto RJ45 Ethernet de 100 Mb, conectores USB 2.0 para host (tipo A estándar) y para cliente (tipo B micro USB).

Adicionalmente posee conjuntos de pines para FTDI, UART, ICSP, SDIO, y pines independientes que proveen entradas/salidas digitales, entradas analógicas, salidas PWM de 12 bits de resolución, 1 SPI master, I2C master.

A nivel de memoria presenta 256 MB DDR3, SRAM de 512 kb integrada, Flash NOR de 8 MB (para firmware y bootloader), EEPROM de 8 KB, y una ranura para tarjeta microSD de hasta 32 GB.

Tiene RTC integrado así como compatibilidad de hardware y pines con la placa Arduino Uno R3.

Una de las particularidades del trabajo con placas SBC es que requieren manipular sistemas operativos embebidos.

Si bien, en general, las tareas de instalación y configuración de un SO de servidor son similares al trabajo sobre una PC tradicional, no resultan triviales al momento de preparar el sistema embebido para una solución específica como la dada (Aranda, 2018).

SELECCIÓN DEL SOFTWARE DE SISTEMA

Las placas Intel Galileo Gen 2 pueden correr con sistema operativos MS Windows, GNU/Linux y MacOS.

Se adopta como sistema operativo del servidor una distribución basada en GNU/Linux por tratarse del sistema elegido generalmente, por los desarrolladores para SBCs.

Debido a limitaciones impuestas por el software del núcleo Linux para la SBC disponible, se adopta una distribución GNU/Linux Debian con herramientas de administración básicas pero sistema de archivos estándar. Esto permite preparar el sistema operativo de un servidor completo, configurado sólo con las aplicaciones requeridas por este proyecto y con soporte a los diferentes paquetes a través de repositorios estándares.

Como es habitual, el sistema operativo elegido limita la elección del servidor web a uno que opere sobre aquél de manera nativa.

Entre las alternativas tecnológicas actuales para prestar servicios web se encuentran los servidores genéricos e independientes del lenguaje (como Apache, Lighttp, NGINX), los asíncronos basados en javascript (como node.js o V8), y los basados en lenguaje python, ya mencionados.

Se descarta el uso de Node.js o similar por las características del proyecto, en cuanto a que existe una baja concurrencia y poca interacción asíncrona entre cliente y servidor.

Se decide usar un servidor web independiente, ante las ventajas de poder usarlo para prestar servicios no sólo bajo python sino con otros lenguajes de manera colaborativa, y fundamentalmente de resultar más robusto y estable en sistemas de producción.

En base al detalle de características de los diferentes servidores web listados en Wikipedia (Wikipedia, 2017), y dejando de lado tanto los productos comerciales como los que no aceptan WCGI/FastCGI (requerido para ejecutar Python mediante mecanismos CGI, de manera sencilla y optimizada) las opciones disponibles para GNU/Linux son: Apache, NGINX, Lighttpd y Cherokee.

Se descarta Cherokee ya que la última versión (1.2.103) es del año 2013 y la actividad observada en sus foros es menor que en los otros.

Para decidir el software que cumplirá las funciones de servidor Web se toman los resultados del análisis que considera el tiempo de respuesta, la memoria empleada y la tasa de utilización de CPU mostrados en la Figura 19. (Kunda, K., y Chihana, S., 2017),

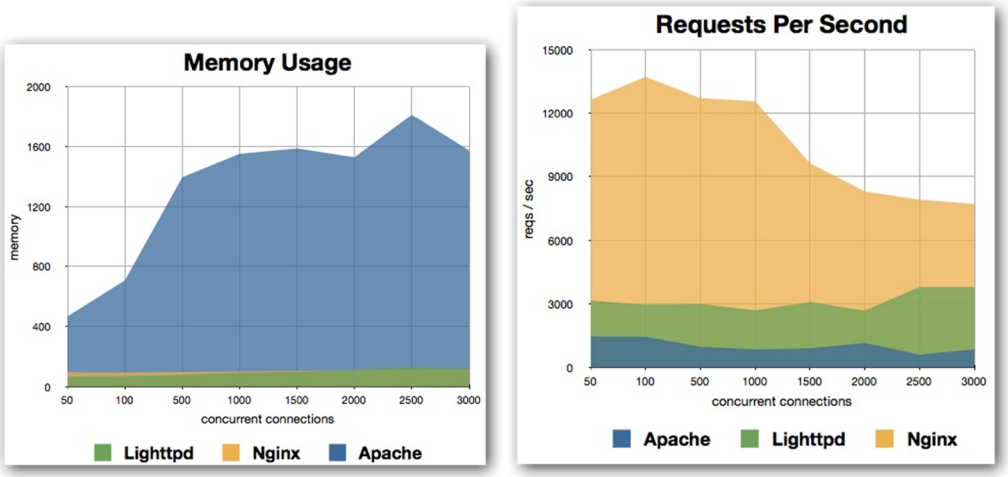

Figura 19: Benchmark de los 3 servidores web más populares.
Fuente: www.researchgate.net

Confirmando lo anterior, TecnoGaming afirma que los 3 servidores más conocidos y utilizados son Apache, Lighttpd y NGINX. (Vojacek, A., 2012)

La diferencia más importante es que Apache trata de proveer la información de manera sincrónica, mientras que NGINX y Lighttpd lo hacen de manera asincrónica y conducida por eventos (peticiones). Esto hace que, los segundos, sean más rápidos, consuman menos recursos y tiendan a soportar una cantidad mayor de conexiones al mismo tiempo.

Características adicionales de NGINX y Lighttpd es que si bien no poseen tanta flexibilidad e integración con otras herramientas y servicios como Apache mediante el uso de módulos, poseen en general una mayor sencillez de configuración.

Finalmente, se elige Lighttpd por ser el que mejor aprovecha la memoria del dispositivo.

Por lo anteriormente descrito, las tecnologías elegidas para satisfacer los requerimientos primarios del sistema son:

- Lighttpd como servidor Web.

- MySQL como servidor de base de datos. Se elige por tratarse de una plataforma madura, estable, escalable, con buen rendimiento general en aplicaciones orientadas a la web, excelente documentación y una comunidad activa.
- Python como lenguaje del lado servidor para el desarrollo de la API REST. Eventualmente, usando además un framework que facilite el desarrollo y eventual despliegue de interfaces web.
- Javascript junto a HTML5 y CSS como lenguajes del lado cliente para el desarrollo de las interfaces de usuario.
- JSON como formato para el intercambio de datos entre los clientes y el servidor.

Los requerimientos actuales del proyecto no exigen contar con un servidor FTP, además el sitio montado no tendrá un grado de actualización importante, por lo que la transferencia de los archivos que se requieran para el mismo se hará mediante los servicios sftp, provistos por el servidor openssh instalado y un cliente remoto.

La aplicación para Host de Impresión Printrun, en la versión elegida, requiere de lenguaje python en sus versiones 2.6.x o 2.7.x

Printrun (Yanev, 2017), está formado por otras aplicaciones denominadas Printcore, Pronsole y Pronterface, además de una colección de scripts con herramientas de ayuda.

- printcore.py: es la librería que permite realizar las operaciones de escritura del host Reprap
- pronsole.py: es la aplicación para el control interactivo de la impresora usando CLI.
- pronterface.py: es la aplicación para el control interactivo de la impresora usando GUI.

Para realizar las operaciones de laminado, se requiere instalar alguna herramienta adicional, ya que Printrun no realiza dicha tarea. Entre las alternativas se encuentran Slic3r y Skeinforge, como las más populares.

Si bien Slic3r es mucho más rápido que Skeinforge, la versión disponible no es compatible con el resto de la plataforma, por lo que se instala Skeinforge.

Las pruebas de laminado realizadas usando la SBC galileo 2 requieren de un significativo tiempo de procesamiento, por lo cual se considera que la mejor forma de trabajo es subir archivos de modelo (generalmente stl) que cuenten con el correspondiente archivo de construcción (en gcodes), ya terminado.

Si bien no contemplado inicialmente, se consideró apropiado colocar una cámara a los efectos de permitir en la aplicación cliente una interfaz de usuario más amigable.

La placa Galileo no dispone de procesador gráfico, por lo que todo el trabajo recae sobre un núcleo capaz de ejecutar 1 único hilo. Como el trabajo principal es realizar tareas de monitoreo de operación sobre la impresora y almacenamiento en la base de datos, la capacidad del mismo para manejar gráficos se ve limitada. Para manipular el servicio de video remoto se realizan pruebas con OpenCV, FSWebcam y MJPG_streamer. Desde el punto de vista de compatibilidad con el hardware, ocupación de memoria de la aplicación, y tiempo de respuesta, los mejores resultados se obtuvieron con el último de los productos software, que produjo un streaming más fluido, evitando el error segmentation fault generado por el microprocesador cuando se ejecutan ciertas aplicaciones en distribuciones Linux. (Debian, 2014)

PROTOTIPO DE LA PLATAFORMA

La plataforma prototipo diseñada para la prueba de la API responde al escenario A y se enmarca en la definición de SaaS, según lo descrito anteriormente.

A través de ella un usuario puede hacer monitoreo y control remoto de una impresora 3D que le pertenece, usando software ubicado en la nube.

En este prototipo, la nube está modelada a través del servidor web, el servidor de base de datos, la API RESTful, la aplicación cliente Web, así como el dispositivo donde los mismos se ejecutan y la red de interconexión.

El esquema de la instalación responde al de la Figura 20.

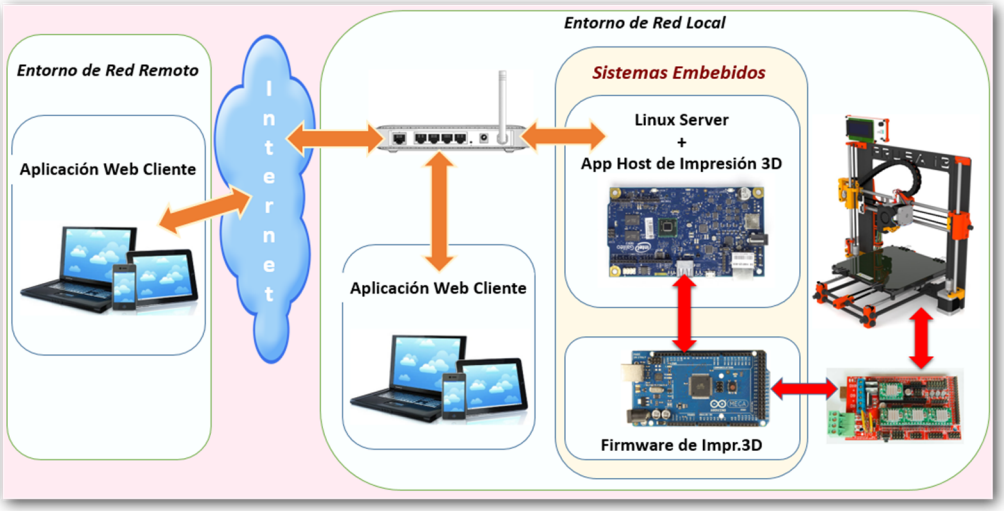

Figura 20: Esquema de la plataforma prototipo para la impresión 3D.
Fuente: elaboración propia.

Luego de las tareas de selección, instalación y configuración del software, este prototipo presenta las siguientes características:

- Hardware:
    - Impresora 3D: Prusa I3
    - Dispositivo Servidor: SBC Intel Galileo 2

    - Router: TP-LINK WR841ND
- Software
    - Firmware: Marlin
    - Aplicación Host de Impresión: pronsole (componente de Printrun)
    - Sistema Operativo: GNU/Linux
    - Servidor Web: Lighttp
    - Servidor de Base de Datos: MySQL
    - API REST: en lenguaje Python (sin frameworks)
    - Cliente Web: HTML5, CSS3 y javascript
- Hardware y software adicional
    - Cámara web: Genius FaceCam 320X
    - Servidor de stream de video: mjpg_streamer
    - Herramienta de laminado: skeinforge

El software usado para el desarrollo, pruebas y documentación ha sido:

- Postman 6.7.3
- WingIDE 6.0
- Chrome 72.0.3626 (herramientas para desarrolladores)
- Bitvise SSH Client 8.24
- Sublime 3.1.1
- Paquete de ofimática

DISEÑO DE LA API REST

En el caso de esta API, muchos de los servicios no se corresponden directamente con acciones sobre la base de datos, sino a operaciones de control sobre la impresora 3D.

En base a ello, los métodos disponibles en una petición HTTP, son usados para:

- GET: recuperar un elemento del sistema de archivos, datos desde la aplicación de Host de impresión o datos provenientes de una sentencia SQL.
- POST: inserción de datos mediante una sentencia SQL o creación de un elemento en el servidor (sea un nuevo archivo o una operación en particular de la impresora).
- PUT: no usado.
- PATCH: modificar el estado de algunos atributos del mismo (es similar al anterior, pero opera de manera parcial sobre un recurso)
- DELETE: eliminar un elemento del sistema de archivos y/o borrado de datos mediante una sentencia SQL.

Los recursos identificados del sistema que serán expuestos por la API en la red, son: la impresora, los archivos de diseño, los usuarios, las conexiones a una impresora, las operaciones que ésta puede realizar y el estado del proceso de impresión.

Cada URL de este trabajo se estructura de la siguiente forma:

{protocolo}://{hostname}[:puerto]/{v1/ruta_recurso}?{consulta de filtrado}

Los URIs han sido definidos en base a las entidades anteriores y considerando que esta API puede evolucionar a futuro, se ha agregado al URL el identificador de la versión: /v1

Sin embargo, salvo que sea estricto, en los siguientes ejemplos se indicará solamente el URI del recurso, sin usar /v1.

Para los recursos expuestos por la API se proponen los métodos detallados en la Tabla 3 y en particular para atender las operaciones de control sobre la impresora y la cámara se listan los códigos admitidos en la Tabla 4.

La recomendación de diseño sugiere que cuando se producen cambios de estado parciales debería usarse el comando PATCH, con una perspectiva puesta en una estructura de datos.

Sin embargo, las operaciones de control de la impresora (y de otros dispositivos adicionales), producen un cambio de estado, sólo si la acción es viable y se pudo realizar.

Por lo que, se adopta vincular el concepto de operación y selección directas al comando POST. Un razonamiento similar se sigue con la alternativa de establecer la conexión con una impresora.

Tabla 3. Descripción de los recursos (URIs) y verbos asociados.
Fuente: elaboración propia.

Recurso	GET	POST	DELETE	PATCH
/files3d	Lista los archivos de diseño existentes	Crea (luego de upload) un nuevo archivo de diseño	-	-
/files3d/4	Recupera los detalles del archivo 4	Selecciona para imprimir el archivo de diseño con id 4, crea un nuevo objeto 3D a imprimir	Elimina el archivo de diseño con id 4	-
/login3d	-	Crea (una vez validado) un nuevo token de acceso	-	-
/objects3d	Lista las piezas 3D ya impresas (historial)	-	-	-
/objects3d/2	Recupera detalles de la pieza impresa con id 2	-	Elimina la pieza	-

			impresa con id 2	
/printers3d	Lista las impresoras 3D disponibles	Agrega una nueva impresora 3D	-	-
/printers3d/1	Recupera detalles de la impresora con id 1	-	Elimina la impresora con id 1	-
/printers3d/1/ connections	Lista las conexiones disponibles a la impresora con id 1	-	-	-
/printers3d/1/ operations	Lista las operaciones disponibles para la impresora con id 1	-	-	-
/printers3d/1/ operations/4	-	Ejecuta la operación 4 sobre la impresora 3D con id 1	-	
/printers3d/1/ states	Recupera los detalles del proceso en curso	-	-	-
/users3d	Listado de los usuarios que pueden usar impresoras 3D	Agrega un nuevo usuario de impresoras 3D	-	-
/users3d/8	Recupera detalles del usuario identificado con id 8	-	Elimina un usuario con id 8	Actualiza un usuario identificado con id 8

Tabla 4. Códigos de operación de la impresora 3D.
Fuente: elaboración propia.

Cód.	Descripción de la Operación	Parámetros adicionales requeridos
1	Reiniciar la impresora	-
2	Comenzar la impresión	-
3	Suspender la impresión	-
4	Reasumir la impresión	-
5	Detener la impresión	-
6	Apagar motores	-
7	Encender ventilador del HotEnd	-

8	Apagar ventilador del HotEnd	-
10	Configurar la temperatura del HotEnd	Valor de temperatura (en ºC)
11	Apagar el HotEnd	-
13	Configurar la temperatura de HotBed	Valor de temperatura (en ºC)
14	Apagar el HotBed	-
15	Cambiar la velocidad del cabezal de impresión	Valor de velocidad (en %)
16	Cambiar la velocidad del flujo de filamento	Valor de velocidad (en %)
17	Extruir	longitud de extrusión (en mm), y velocidad de extrusión (en mm/min)
18	Retraer	longitud de extrusión (en mm), y velocidad de extrusión (en mm/min)
25	Ir al origen de los ejes X-Y-Z	-
26	Ir al origen del eje X	-
27	Ir al origen del eje Y	-
28	Ir al origen del eje Z	-
29	Ir al centro del plano X-Y	-
30	Mover el cabezal en el eje X	Valor del desplazamiento (en mm)
31	Mover el cabezal en el eje Y	Valor del desplazamiento (en mm)
32	Mover el cabezal en el eje Z	Valor del desplazamiento (en mm)
40	Conectar impresora	Denominación de la interfaz y velocidad de transmisión (en bps) del port
41	Desconectar impresora	-
42	Conectar al servidor de stream de video	-
43	Desconectar del servidor de stream de video	-

En las respuestas dadas por un servidor, generalmente, se presenta uno de los siguientes tres tipos de estados:

- Todo funcionó a la perfección
- Ocurrió un error producido por una mala petición del usuario

- Ocurrió un error de nuestro lado

Las buenas prácticas de ingeniería del software indican que una API no debería tener más de 10 estados posibles. Para la API en desarrollo se han considerado 6 estados.

Tabla 5. Descripciones de códigos de estado.
Fuente: elaboración propia.

Código Nº	Descripción del Mensaje
200	OK. Respuesta satisfactoria. El código 200, va siempre acompañado de un subcódigo que indica el resultado interno de éxito o error, de toda petición aceptada. Incluso en los caso de peticiones encargadas de REQUEST que suelen no devolver un body (algunos POST/DELETE/PUT/PATCH)
400	Requerimiento Inválido. La petición tiene algún error, generalmente ocurre cuando los datos proporcionados en POST o PUT no pasan la validación (error en los parámetros sobre el recurso “operations”, o bien el archivo subido como recurso “files3d” posee un formato incorrecto o corrupto, o datos de usuario inválidos).
401	Acceso no Autorizado. Es necesario identificarse primero.
404	No encontrado. Esta respuesta indica que el recurso requerido no existe, es decir la URL no corresponde a un recurso.
405	Método no Admitido. El método HTTP utilizado no tiene soporte para el recurso dado.
500	Error Interno en el Servidor. Puede ser un error inesperado en el servidor y también a ciertas operaciones (de conexión a base de datos, comunicación RPC y FS).

Tabla 6. Códigos de estados por recurso.
Fuente: elaboración propia.

Recurso	Método	Código de estado (Respuesta Satisfactoria)	Código de estado (Respuesta con Error)
/files3d	GET	200	401 y 500
/files3d	POST	200	400, 401 y 500
/files3d/4	POST	200	401 y 500
/files3d/4	DELETE	200	401 y 500

/login3d	POST	200	400, 401 y 500
/objects3d	GET	200	401 y 500
/objects3d/2	GET	200	401 y 500
/objects3d/2	DELETE	200	401 y 500
/printers3d	GET	200	401 y 500
/printers3d	POST	200	400, 401 y 500
/printers3d/1	GET	200	401 y 500
/printers3d/1/connections	GET	200	401 y 500
/printers3d/1/operations	GET	200	401 y 500
/printers3d/1/operations/4	POST	200	400, 401, 404 y 500
/printers3d/1/states	GET	200	401 y 500
/users3d	GET	200	401 y 500
/users3d	POST	200	400 y 500
/users3d/8	GET	200	401 y 500
/users3d/8	PATCH	200	401 y 500
/users3d/8	DELETE	200	401 y 500

Las respuestas a cualquier requerimiento HTTP presentan 2 partes el header y el body.

El encabezado (header) posee básicamente: ‘protocol_version’, ‘status_code’ (el número del código de estado) y ‘status_message’ (una breve cadena descriptiva del resultado).

Los campos ‘status_code’ y ‘status_message’ responden a lo detallado en la Tabla 5.

‘protocol_version’: ‘HTTP/1.1’,

‘status_message": ‘OK’,

‘status_code’: ‘200’

Otros headers que pueden agregarse, según el caso son:

Content-Type: hace referencia al tipo de contenido del cuerpo. En esta API, se ha usado el valor ‘application/json’ para indicarle al cliente que el contenido enviado está en formato JSON.

Cache-Control: especifica la política del mecanismo de caché. En general, se ha usado el valor no-cache para indicar al cliente que no almacene en cache el recurso o colección solicitado.

Date: Fecha y hora del servidor.

Server: utilizado para identificar al servidor.

El cuerpo (body) de la respuesta posee una estructura de 3 campos denominados: ‘error_code’, ‘error_message’ y ‘data’

En los casos de respuesta satisfactoria a la petición, los 3 campos se devuelven siempre, cambiando el contenido en base al requerimiento realizado y si el requerimiento se ha resuelto de manera satisfactoria o ha habido algún error específico.

Por ejemplo, en los casos que no hay error, el valor de ‘error_code’ es 0, y el mensaje es descriptivo de la operación realizada.

Mientras que el campo ‘data’ almacena los datos específicos, que varían en función tanto del recurso como del método usado.

En el Anexo C se muestra el detalle de los servicios como así también la información incluida en el cuerpo de las peticiones y respuestas

En el caso de una respuesta insatisfactoria, los campos ‘error_code’ y ‘error_message’ proveen alguna información del error producido (como puede observarse en la Tabla 7), mientras que el campo ‘data’ puede encontrarse vacío o con breve información útil para desplegar en la interfaz cliente.

Tabla 7: Códigos auxiliares de error.
Fuente: elaboración propia.

error_code	Error_message
1101	Error de acceso a la base de datos
1102	Error de acceso a la cámara
1103	Error de acceso a la impresora
1104	El archivo no pudo ser creado/guardado
1105	Archivo duplicado
1106	La impresora no puede procesar la operación
1107	El archivo no pudo ser cargado/leído
1108	El archivo no pudo ser eliminado
1109	El archivo cargado está corrupto
1110	Deben alcanzarse las temperaturas mínimas
1111	Archivo de formato incorrecto
1201	La lista solicitada está vacía
1202	El usuario buscado no existe
1203	El usuario buscado no existe
1204	El archivo buscado no existe
1205	El objeto buscado no existe
1206	La operación no está implementada.
1207	La impresora buscada no existe
1208	La impresora no está habilitada
1209	La impresora está usada por otro usuario
1210	La impresora no está asignada al usuario
1301	El password es incorrecto
1401	El formato de archivo no es válido
1402	El valor dado para la operación indicada no es válido.
1403	Los parámetros dados son inválidos

Peticiones con criterios de ordenamiento

Para la cadena de consulta, se asumen las siguientes reglas sintácticas:

- files3d?field={name|date}

&order={asc|desc}

&page=número

&size=número

Donde:

- ?: caracter indicador de búsqueda
- field: parámetro indicador de listado sin filtrar, según el campo especificado a continuación.
- name|date: valores de campo disponibles (para el recurso archivo)
- order: denominación del parámetro de criterio de ordenamiento
- asc|desc: valores disponibles (ascendente o descendente). El criterio de orden se aplica al campo indicado en field
- page: denominación del parámetro página deseada, para calcular el primer recurso a devolver en la respuesta
- size: denominación del parámetro de cantidad máxima de ítemes a devolver en la respuesta
- número: es el valor particular deseado

Peticiones con criterios de filtrado

Para la cadena de consulta, se asumen las siguientes reglas sintácticas:

- files3d?search=texto

  &order={asc|desc}

  &page=número

  &size=número

Donde:

- ?: caracter indicador de búsqueda
- search: parámetro indicador de listado con filtrado. Se aplica siempre al campo name.
- texto: cadena conteniendo el patrón de búsqueda para nombres.
- order: denominación del parámetro de criterio de ordenamiento
- asc|desc: valores disponibles (ascendente o descendente). El criterio de orden se aplica al campo indicado para field
- page: denominación del parámetro página deseada, para calcular el primer recurso a devolver en la respuesta
- size: denominación del parámetro de cantidad máxima de ítemes a devolver en la respuesta
- número: es el valor particular deseado

La documentación completa de la API está disponible en https://documenter.getpostman.com/view/6686997/SVn2MvDz?version=latest

IMPLEMENTACIÓN DE LA PLATAFORMA

Implementar la plataforma de impresión 3D implica, en este caso, implementar una API RESTful. Esto es, construir una aplicación que respetando los principios REST sea capaz de proveer servicios los web específicos.

Dado que se trata de la construcción de software, es necesario según las buenas prácticas de ingeniería de software, seleccionar el o los patrones de diseño que guiarán el desarrollo.

En este proyecto se adopta un patrón de diseño del tipo Controlador Frontal orientado a la Web.

Como se trata sólo de proveer servicios REST, los controladores de segundo nivel del patrón no se dedican al renderizado en las vistas de usuario sino sólo al tratamiento de los datos.

Para la denominación de las diferentes entidades se usará la siguiente política:

- El nombre de una entidad de modelo en singular y con estilo CamelCase
- El nombre de una entidad adicional parte del nombre básico anterior y agrega el tipo (o la finalidad) de la misma. Ya sea un controlador, una vista u otro tipo.
- El nombre del módulo corresponde al nombre de la entidad se define en minúsculas sin la letra P. Dado que los módulos se almacenarán en un mismo directorio, se usa un guión bajo para mejorar la legibilidad de tipo (_controller, _helper, _view).

La estructura de la API REST propuesta surge entonces de combinar las características indicadas. Se puede esquematizar la misma como sigue:

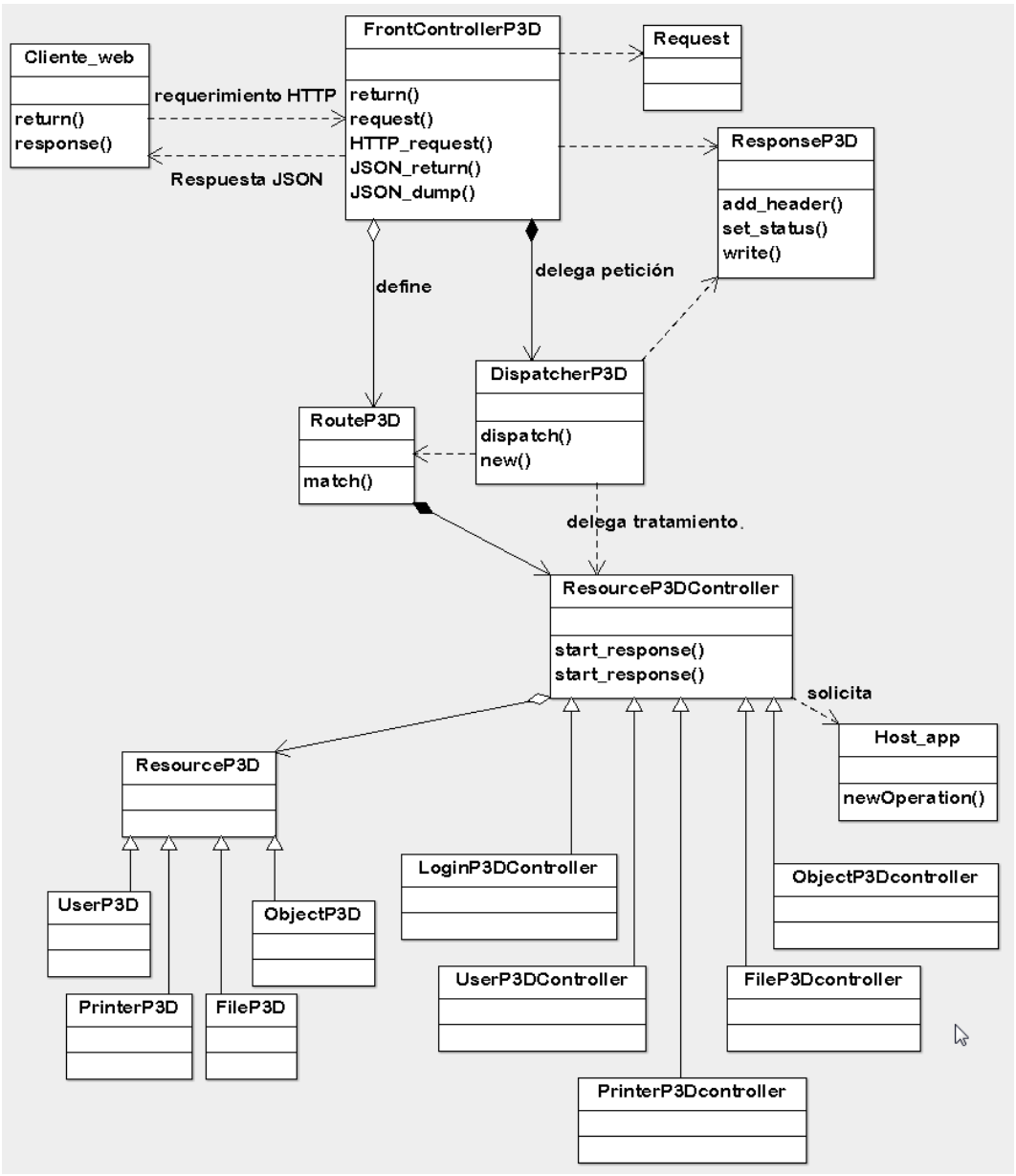

Figura 21: Diagrama de clases de la API REST.
Fuente: elaboración propia.

Donde, las clases representan:

- FrontP3DController: (capa de control, interfaz con capa de presentación) clase que define las rutas y los recursos disponibles, posee el objetivo fundamental de autenticar el origen del requerimiento y delegarlo si es válido.

- DispatcherP3D: (capa de control, interfaz con capa de presentación) clase encargada de verificar cada requerimiento, y lanzar el controlador según el Recurso que corresponda (FileP3D, ObjectP3D, PrinterP3D y UserP3D).
- RouteP3D: (ayudante en la capa de control) clase que registra las rutas de acceso a cada recurso y verifica las operaciones que se realizan sobre ellos.
- ResponseP3D: (ayudante en la capa de presentación) clase que contiene el mensaje devuelto a cada petición.
- FileP3D: (capa de modelo) clase que define las operaciones para el manejo de archivos que contienen el diseño de las piezas 3D así como de las operaciones CRUD en la base de datos.
- FileP3DController: (capa de control) clase destinada a resolver las peticiones de almacenamiento y recuperación de las piezas 3D. Opera en base a las definiciones existentes en la clase FileP3D. Se encarga también de interactuar con las aplicaciones del Host de impresión, particularmente para realizar el slicing.
- ObjectP3D: (capa de modelo) clase que define las operaciones CRUD en la base de datos para gestionar la información de las piezas impresas.
- ObjectP3DController: (capa de control) clase destinada a resolver las peticiones de almacenamiento y recuperación de las piezas 3D impresas.
- PrinterP3D: (capa de modelo) clase que define las operaciones CRUD en la base de datos para gestionar la información de las impresoras 3D.
- PrinterP3DController: (capa de control) clase destinada a resolver las peticiones de conexión y operación de una impresora 3D. Opera en

base a las definiciones existentes en la clase PrinterP3D. Se encarga también de interactuar con las aplicaciones del Host de impresión, especialmente para recuperar o asignar datos vinculados a la impresora durante su funcionamiento.

- UserP3D: (capa de modelo) clase que define las operaciones CRUD en la base de datos para gestionar la información de usuarios.
- UserP3DController: (capa de control) clase destinada a resolver las peticiones de gestión de usuarios del sistema.
- LoginP3DController: (capa de control) clase destinada a resolver las peticiones de acceso y/o control de los usuarios del sistema, aprovecha las definiciones de la clase UserP3D. Fundamentalmente se encarga de validar un usuario y generar una cadena JSON Web Token, codificadas en base64 mediante HMAC SHA 256.
- ResourceP3DController: (capa de control) clase madre de los controladores de cada recurso.
- ResourceP3D: (capa de modelo) clase madre de los modelos de cada recurso.
- Cliente_app y Host_app: no son clases de diseño de esta API, se comportan como actores con los que interactúa este sistema. Se agregan al diagrama para representar a las otras entidades significativas que intervienen.

COMUNICACIÓN ENTRE APLICACIONES DE SERVIDOR

En la siguiente figura se esquematizan las comunicaciones y otras interacciones requeridas entre las aplicaciones participantes:

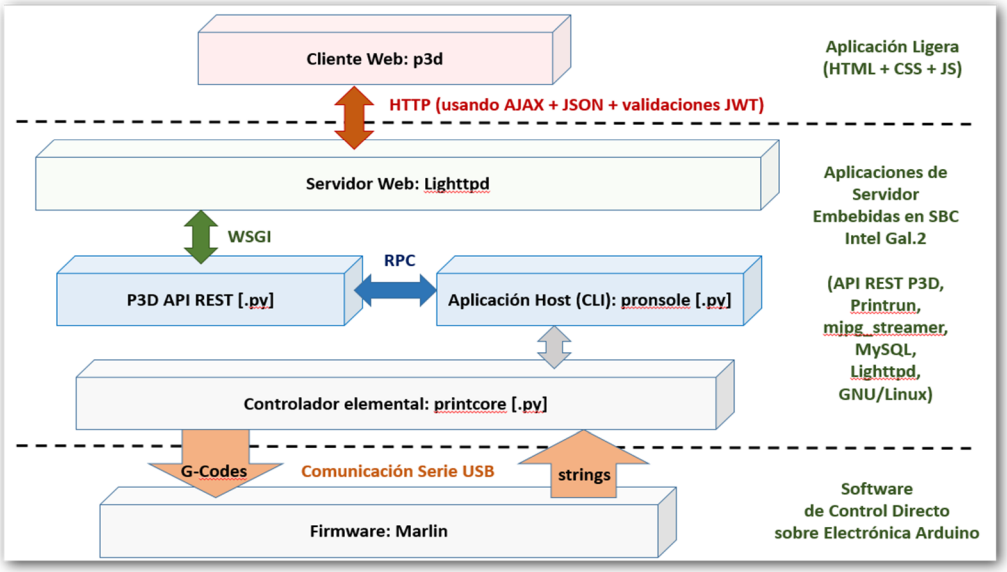

Figura 22: Interacciones entre aplicaciones de la plataforma.
Fuente: elaboración propia.

En la misma se observa que la creación y exposición de recursos mediante una API le permite a una aplicación web interactuar con otras aplicaciones a través de comunicaciones de máquina a máquina.

En este caso, la aplicación proveedora de la API REST debe interactuar, por un lado con la aplicación presente en la máquina del cliente web, y por el otro debe ser capaz de acceder a la máquina controlada: la impresora 3D.

Resulta necesario, entonces, definir los mecanismos por los cuales sea posible ejecutar las instrucciones propias de la aplicación que actúa como Host de impresión, ya sea para recuperar información de estado como realizar acciones de control.

Entre las alternativas de diseño e implementación viables se tienen:

- Desarrollar una aplicación web nueva que cumpla también la función de Host de Impresión.
- Construir una clase que extienda a alguna de las clases disponibles en el paquete Printrun y que conforman la base de las aplicaciones

existentes (Printcore, Pronsole o Pronterface), y que agregue el comportamiento requerido por la API.

- Aprovechar mecanismos elementales de comunicación disponibles en la clase Pronsole y ampliar las definiciones básicas existentes conforme a las necesidades de la aplicación de API propuesta.

Se descarta la primera alternativa, ya que produce un desarrollo redundante e inconsistente al reemplazar a las aplicaciones existentes (en realidad daría lugar a un fork de la implementación original) y conlleva un esfuerzo de desarrollo elevado.

Se opta por la tercera alternativa, ya que produce un acoplamiento menor respecto de la segunda y, si bien no es representativo en este trabajo, favorece la posibilidad de desplegar servidores de manera separada.

La API debe interactuar con otros componentes del sistema, particularmente las aplicaciones que gestionan la impresora y el stream de video. En el primer caso, las comunicaciones se realizan usando mecanismos RPC basados en Python. La aplicación de consola “pronsole” provee algunos servicios a través de uno de sus módulos. Se edita el mismo con el fin de ampliar los servicios definidos y lograr una mejor interacción con la API. En el segundo caso, se hace uso de invocaciones directas a scripts del sistema operativo.

Puede verse en la Figura 23 los elementos que participan y en la Figura 24 las interacciones que se producen en la API cuando se produce la solicitud de un recurso desde el cliente web.

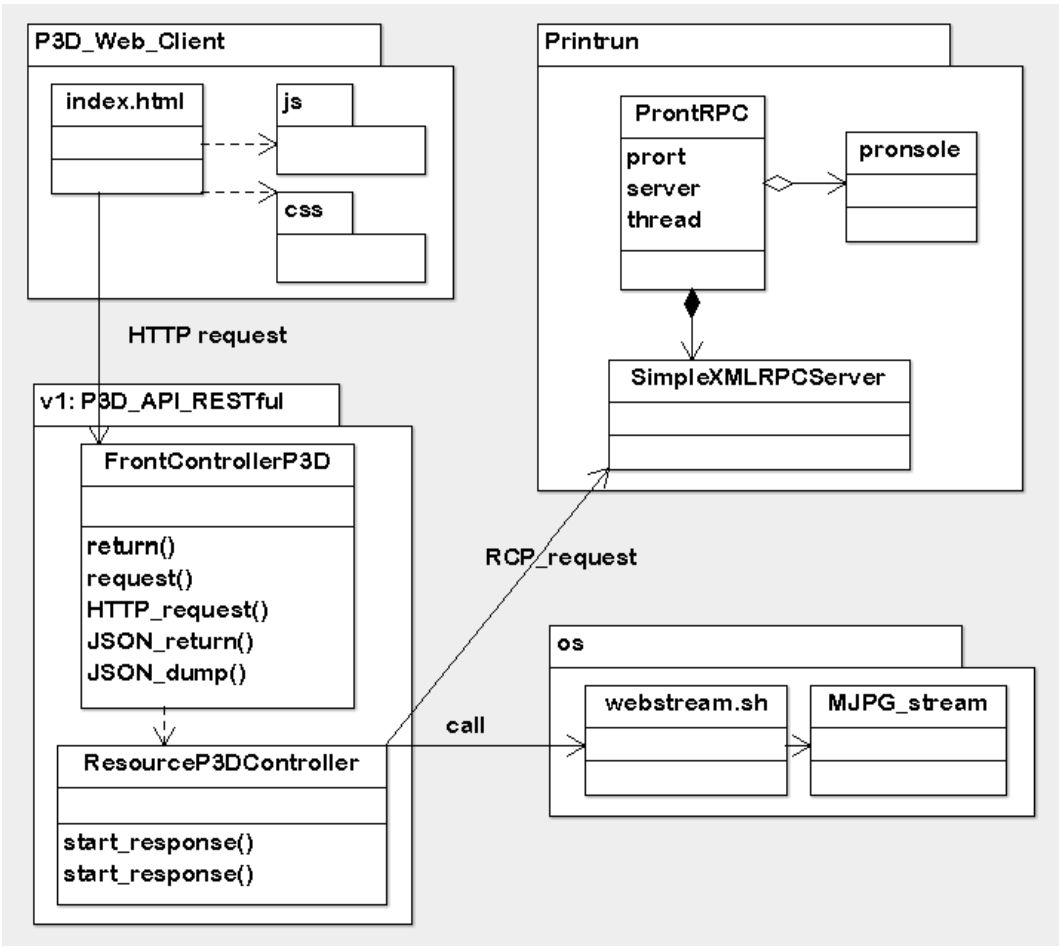

Figura 23: Diagrama de Paquetes que participan de la API RESTful.
Fuente: elaboración propia.

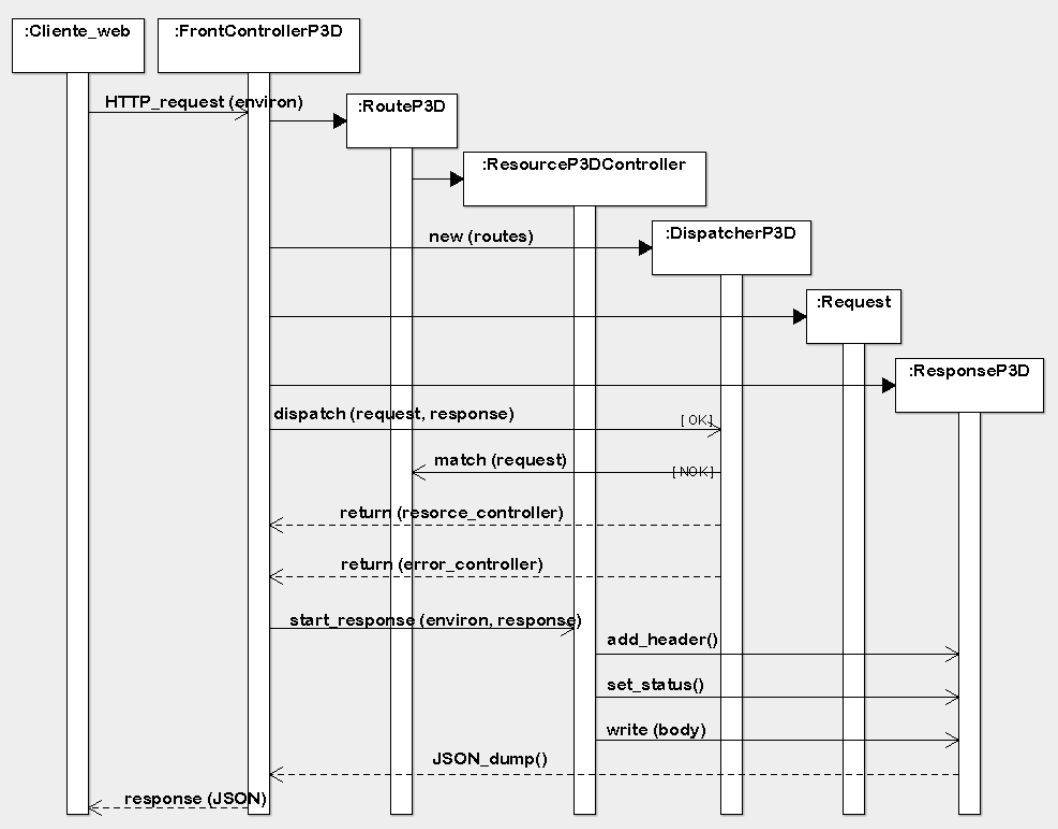

Figura 24: Diagrama de Secuencia en la API para un requerimiento HTTP. Fuente: elaboración propia.

El almacenamiento requerido por la aplicación para realizar la gestión de impresoras, archivos y usuarios utiliza el esquema de datos mostrado en la Figura 25.

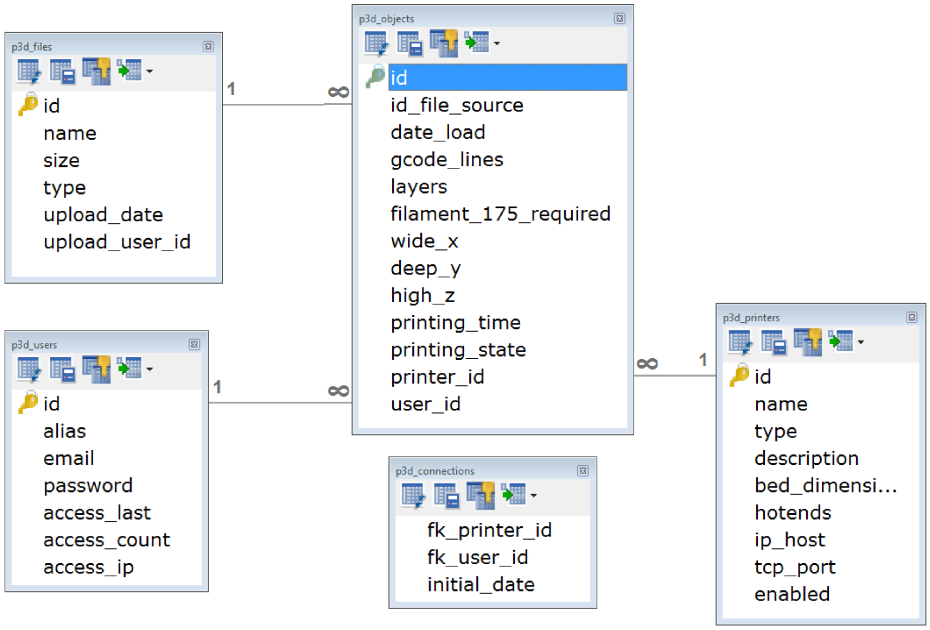

Figura 25: Modelo relacional usado por la API RESTful.
Fuente: elaboración propia.

DISEÑO FÍSICO DE LA PLATAFORMA

Dadas las particularidades de este proyecto, se define un espacio denominado p3dawar (abreviatura de Printer 3D - Another Web API REST) con una estructura de directorios relativamente estándar, alojada en el directorio web por defecto (/var/www), como muestra la Figura 26.

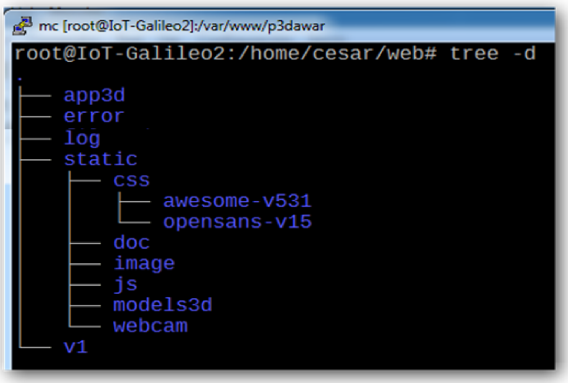

Figura 26: Estructura de directorios de la plataforma.
Fuente: elaboración propia.

En ella se encuentran:

- app3d: es un directorio protegido, previsto para almacenar los scripts de la aplicación para el usuario final, en caso que se utilice un lenguaje diferente al de la API.
- error: es un directorio protegido, con recursos usados cuando se producen mensajes de error.
- log: es un directorio protegido, que contiene archivos con el registro de eventos y errores de uso de la API o aplicaciones asociadas.
- static: es un directorio público (servido), con archivos estáticos. Fundamentalmente contiene archivos html.
- css, image, doc y js: directorios que contienen recursos estándares para las páginas de usuario final, considerando el tipo de archivo complementario usado.
- models3d: es un directorio que almacena tanto los archivos de diseño 3D como los archivos de fabricación conteniendo las instrucciones en g-code, necesarias para imprimir las piezas.
- webcam: directorio usado, opcionalmente, para almacenar imágenes obtenidas por la cámara.
- v1: es un directorio protegido que almacena la API REST, con todos los módulos y paquetes desarrollados en python. Dado que la cantidad de módulos resultantes de la API, en las capas modelo y controlador, es relativamente pequeño se descarta la división en subdirectorios.

IMPLEMENTACIÓN DE JWT

La lógica de negocio usada en este trabajo, para autenticar y autorizar las peticiones, se ha simplificado de manera de verificar únicamente que exista el usuario y la clave en la base de datos. En alguna fase posterior de

ampliación, podría incorporarse un modelo más complejo incorporando permisos y roles.

Conforme al estándar, la estructura del token obtenido tiene 3 partes (header, payload y signature). La cadena de caracteres que conforman el token es generada por la función encode invocada, conforme a la parametrización dada.

En ella, la cabecera almacena los datos fundamentales de tipo de token y codificación usada.

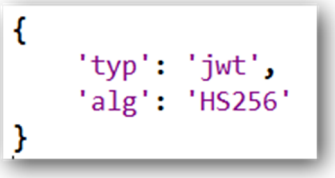

Figura 27: Campos del Header del Token sin codificar.
Fuente: elaboración propia.

En el cuerpo o payload del token se diseñó con 4 claims. Los 2 primeros, corresponden a algunos de los sugeridos por el estándar para identificar al usuario y marcar el token. Los últimos 2, se han previsto para manejar mensajes de interfaz o enviar alertas de proceso al usuario evitando el acceso a la base de datos.

En el fragmento de código de la Figura 28, puede verse la secuencia para obtener el token enviado al cliente, usando el módulo jwt estándar provisto por Python:

```
payload = {
    'sub': int(p3d.id), # usuario
    'iat': datetime.utcnow(), # fecha de creacion en formato UTC
    'alias': p3d.alias, # adicional mensajes personalizados
    'email': p3d.email # adicional mensajes
    }
secret_key='P3Dawar-postlogin-UM-2018'
token = jwt.encode(payload, secret_key, algorithm='HS256').decode('utf-8')
```

Figura 28: Código que genera al JWT.
Fuente: elaboración propia.

El fragmento de código javascript de la Figura 29, muestra la recuperación en el cliente de la cadena del token una vez recibida la respuesta JSON.

La misma es asignada en el header para ser usada en las posteriores peticiones que se realicen, aprovechando el atributo Authorization disponible.

```
var token = window.location.hash;
token = token.substring(1, token.length);
$.ajaxSetup({
  headers: {
    'Authorization': token,
  },
```

Figura 29: Fragmento de Petición AJAX con JWT.
Fuente: elaboración propia.

En el fragmento de código de la Figura 30, se tiene un ejemplo de dichas peticiones AJAX.

```
var uri="printers3d";
var url_req = url_api + uri;
var params="";
var method="GET";
$.ajax({
      method: method,
      url: url_req,
      data: params
})
.done(function( dataret, textStatus, jqXHR ) {
    // llenado select
    for (var i in dataret.data){
      $("#printer_sel").append('<option value="'+ dataret.data[i]['id'] + '">' +
    }
  }
});
```

Figura 30: Petición HTTP a la API usando AJAX.
Fuente: elaboración propia.

ESCALABILIDAD A DIFERENTES ESCENARIOS

La plataforma es escalable a cualquiera de los escenarios planteados inicialmente.

Un elemento que facilita lo anterior es la elección de los parámetros de red que definen a cada impresora 3D.

En el caso más simple, el nodo Host cumple las funciones de servidor Web, servidor de base de datos, servidor de aplicaciones y controlador de las impresiones. Todas las comunicaciones entre ellos son locales y el almacenamiento centralizado.

Basta con conocer la dirección IP del nodo Host y del puerto TCP donde se realizarán las peticiones RPC.

En casos más complejos, los servidores se pueden encontrar distribuidos. Corresponderá ajustar las direcciones IP de cada servidor.

En este contexto cabe aclarar que existe un host de impresión por cada impresora físicamente separada.

Al igual que en el caso simple, es fundamental conocer la dirección IP de cada nodo que cumple las funciones de controlador de impresión y del puerto TCP asociado a RPC.

Dado que el controlador de impresión requiere que los archivos de fabricación se encuentren almacenados de manera local. Deberá incorporarse un servicio que permita la transferencia y copia del archivo correspondiente desde el servidor de aplicaciones hasta el servidor de impresión.

DISEÑO DE LA APLICACIÓN WEB CLIENTE

Para que el usuario realice las tareas de monitoreo y control remoto, se optó por una aplicación web diseñada para su uso en un dispositivo móvil, totalmente desarrollada en lenguaje javascript con despliegue gráfico basado en HTML5 y CSS3.

Se trata de una estructura de página única, segmentada mediante elementos de capa en 5 bloques de interacción, denominados:

Archivos, Conexión, Impresora, Cámara e Información.

La interfaz de Archivos (en la Figura 31), permite una gestión básica de archivos de diseño de piezas 3D, con operaciones para ver algunos detalles, listar o buscar bajo diferentes criterios, agregar o eliminar un archivo, o seleccionar un archivo para su carga en la impresora 3D.

Figura 31: GUI correspondiente a la opción Archivos de P3D_App_test. Fuente: elaboración propia.

La interfaz de Conexión (en la Figura 32), permite seleccionar la impresora remota, establecer los parámetros del puerto de conexión del host a la impresora, y despliega datos de estado de la conexión y del archivo cargado para su impresión.

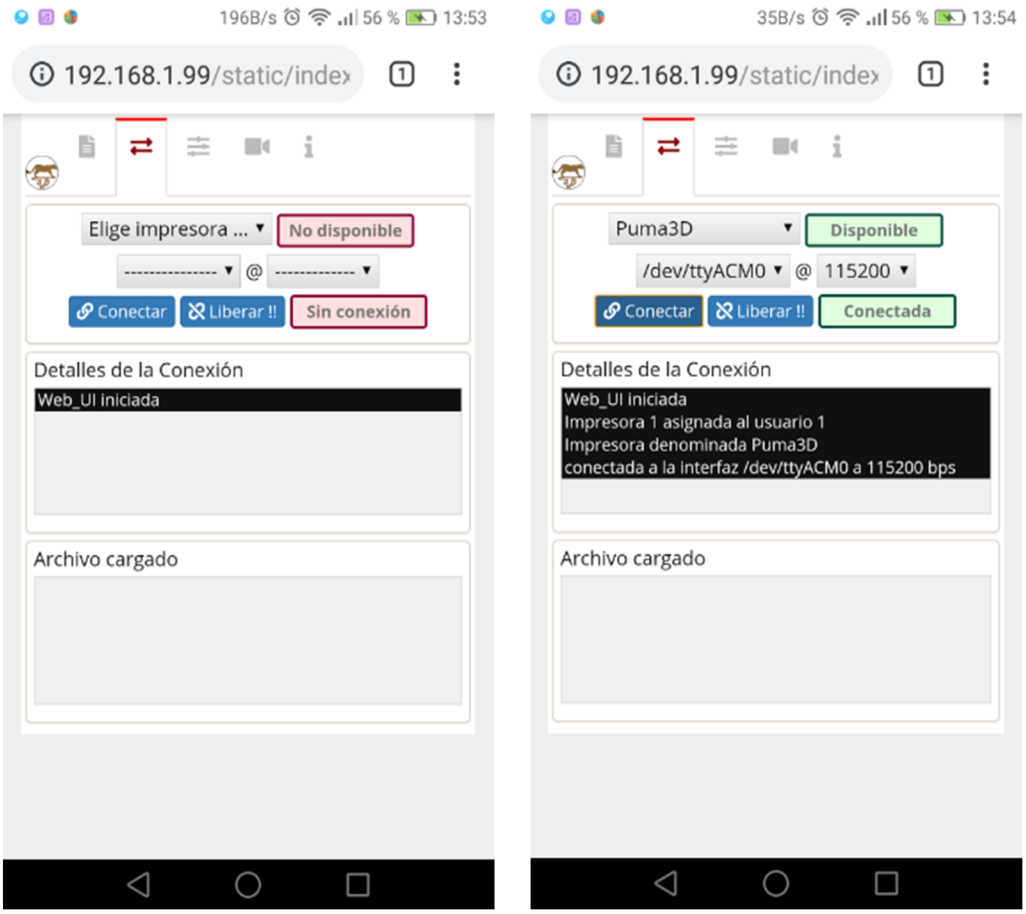

Figura 32: GUI correspondiente a la opción Conexión de P3D_App_test.
Fuente: elaboración propia.

La interfaz de Impresora (en la Figura 33), despliega la mayor cantidad de controles de la impresora 3D, tanto para pruebas, calibración como impresión final.

Dichos controles permiten tareas de encendido y apagado de diferentes motores, movimientos del cabezal y operaciones de avance o retroceso del filamento en el extrusor, y en especial de las tareas fundamentales de la impresora al imprimir una pieza.

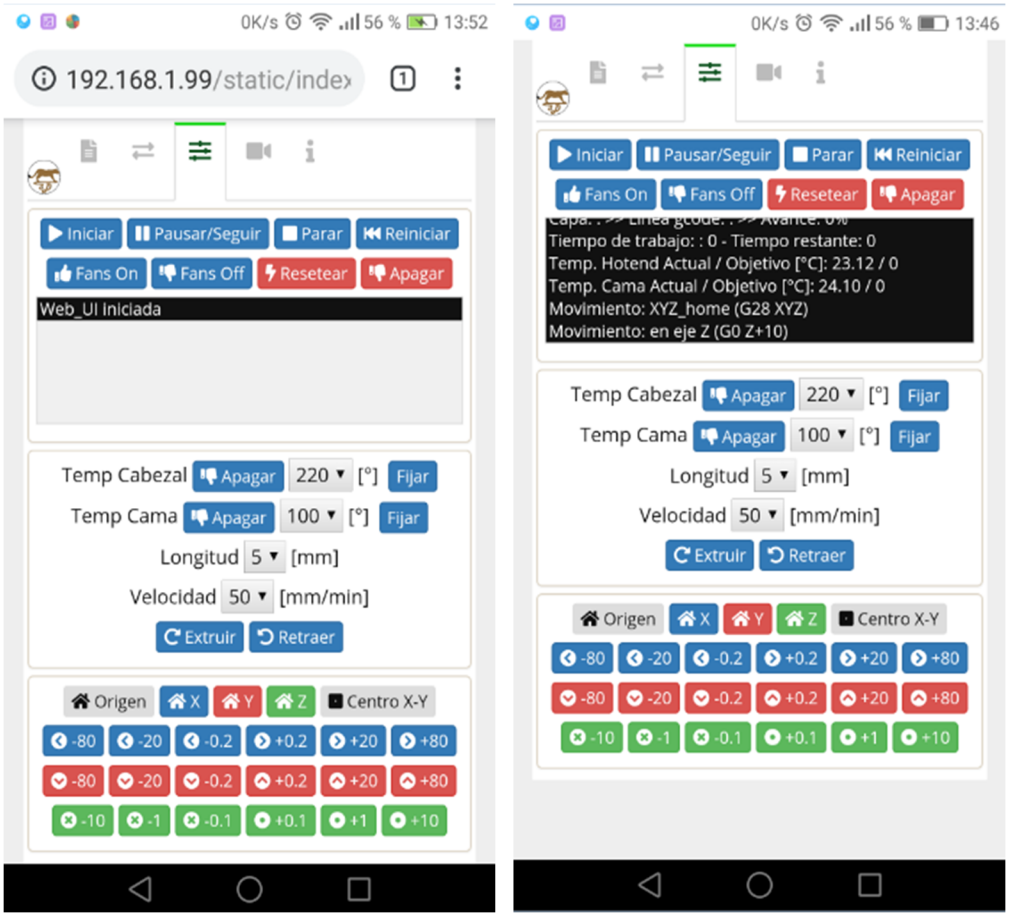

Figura 33: GUI correspondiente a la opción Impresora de P3D_App_test. Fuente: elaboración propia

La interfaz de Cámara (en la Figura 34), permite activar o desactivar tanto la visualización del proceso de impresión remoto como el reporte de los parámetros de operación durante la impresión.

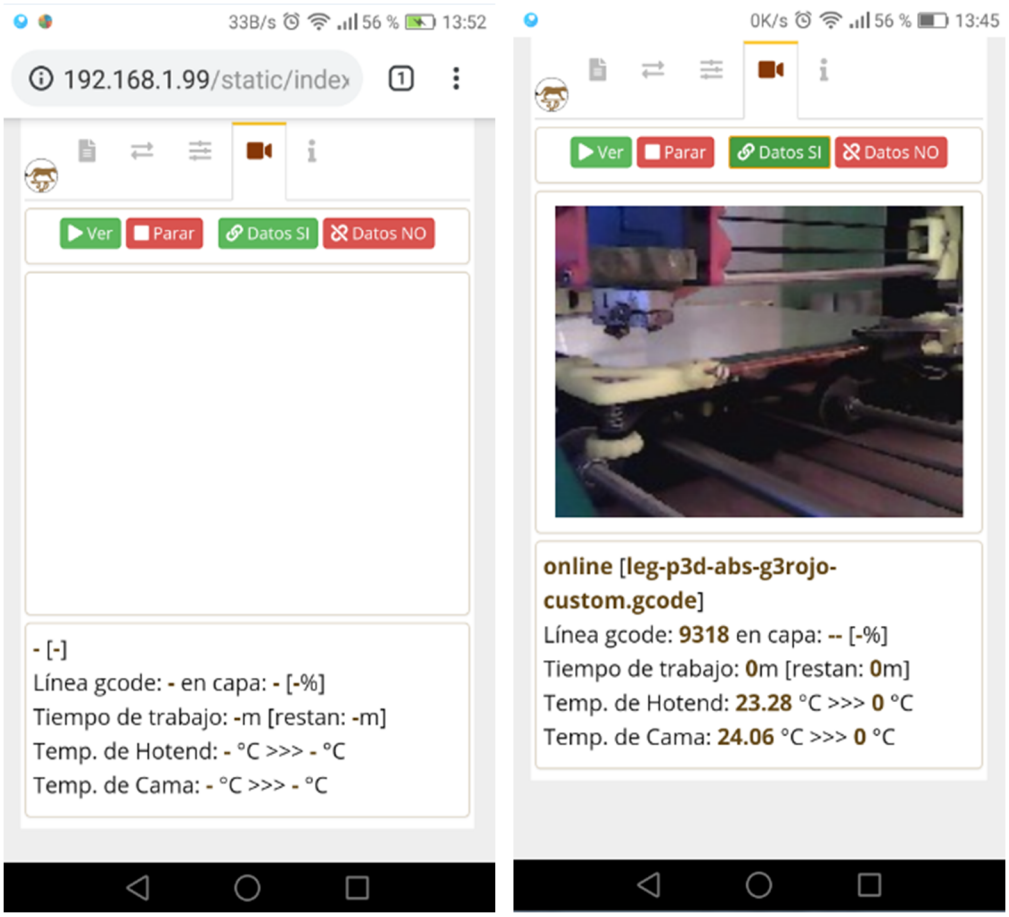

Figura 34: GUI correspondiente a la opción Cámara de P3D_App_test. Fuente: elaboración propia.

La interfaz del menú Información (ver Figura 35), muestra detalles de la versión del aplicativo web implementado.

Figura 35: GUI correspondiente a la opción Información de P3D_App_test. Fuente: elaboración propia.

Para cada una de las ventanas anteriores, se presenta al menos un panel con mensajes de estado asociados a las acciones del usuario.

El acceso a los menús anteriores se realiza luego de una primera fase de validación, a partir de un formulario estándar que solicita usuario y clave (ver Figura 36).

Figura 36: Formulario de acceso a la aplicación prototipo (P3D_App_test).
Fuente: elaboración propia.

En los siguientes URL se ofrecen videos que muestran el uso de la aplicación cliente y el comportamiento de la plataforma:

https://www.youtube.com/watch?v=ktHmrxyklIk

https://www.youtube.com/watch?v=Q7u8hg_qWmg&t=88s

IV. CONCLUSIONES

Los resultados obtenidos demuestran la capacidad de la Plataforma SaaS propuesta, para gestionar el trabajo en tiempo real de la impresora 3D con un desempeño aceptable.

Esto es, que a pesar de las limitaciones del hardware usado para el prototipo, se pudo gestionar y observar el proceso con mediciones y reportes en la interfaz del cliente (incluyendo el streaming de video), con lapsos de tiempo de 1 segundo en una LAN interna.

La definición realizada permite que la Plataforma resulte escalable a diferentes escenarios, desde el ámbito domiciliario con única impresora 3D hasta un ámbito industrial/comercial que ofrezca los servicios de impresión.

Para los casos más simples, de manera secundaria, los resultados muestran la viabilidad de realizar un control remoto de la impresión 3D usando una SBC de bajo costo.

Se ha logrado, con pocas modificaciones integrar a la Plataforma una aplicación de control de impresora 3D preexistente realizada por terceros.

El aplicativo cliente desarrollado es totalmente funcional, permitiendo realizar tanto las tareas de monitoreo y control fundamentales como la administración de los archivos de impresión.

Dado que la API permite registrar en la base de datos los datos asociados a cada requerimiento, incluyendo datos periódicos de temperatura, de velocidad y de otros eventos que ocurran durante el proceso de impresión 3D, así como registrar todos los objetos construidos incluyendo los recursos usados durante el proceso (como tipo y cantidad del filamento consumido, tiempo de impresión) una de las líneas de acción futuras más directa es la de mejorar el aplicativo cliente incorporándole mecanismos de representación gráfica que muestren las curvas de variación de los

parámetros de proceso (on-line e históricos), agregando opciones que permitan desplegar reportes históricos con el detalle de los objetos impresos, cantidad de piezas y estadísticas de uso.

Otras líneas de trabajo futuro, son las de gestionar usuarios con diferentes privilegios de impresión y roles, realizar las tareas de laminado, incorporar otros mecanismos que mejoren la calidad de las imágenes y la velocidad de cada stream de video.

Como conclusión general a este trabajo, se considera el objetivo principal y los secundarios han sido alcanzados y se ha logrado diseñar e implementar una Plataforma que permitirá construir sobre ella extensiones y nuevos casos de uso.

V. BIBLIOGRAFÍA

Amodeo, E. (2013). *Principios de diseño de APIs REST.* Lean Publishing.

Aranda, C. (2018). Instalación y Configuración de Sistemas Embebidos en una SBC Intel Galileo 2 como Base de Plataforma IOT. En *Anales CICCSI 2018* (págs. pp. 195-204). Mendoza: FUSMA ediciones. doi:978-987-45683-6-6

Casper, P. (2015). *Galileo-3D-Printer-Firmware*. Recuperado el 2017, de github.com: https://github.com/vadlak/Galileo-3D-Printer-Firmware

Chavis, B. y Jones, T. (Abril de 2015). *Amazon Web Services.* Recuperado el 2018, de awsstatic.com: https://d0.awsstatic.com/whitepapers/docker-on-aws.pdf

Debian. (2014). *Mensajes #25, #50 y s.s.* Obtenido de debian.org: https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=738575=738575

Eby, P. J. (7 de Diciembre de 2003). *PEP 333 -- Python Web Server Gateway Interface v1.0*. Recuperado el 2016, de python.org: https://www.python.org/dev/peps/pep-0333/

Elgert, M.; Djedid, B.; Elgert, J. y Lindvall, K. . (6 de Abril de 2016). *3D Controller Bot, Intel Edison expands a new 3-D printer market*. Recuperado el 2017, de hackster.io: https://www.hackster.io/4378/3d-controller-bot-8e0ee1

Fernández Montoro, A. (2012). *Python 3 al descubierto.* Madrid: RC Libros.

Intel. (2014). *Datasheet Intel Galileo Gen 2 Development Board*. Recuperado el 2017, de intel.com: https://communities.intel.com/docs/DOC-22795

Jones, M.B.; Bradley, J. y Sakimura, N. (12 de Noviembre de 2013). *JSON Web Token (JWT).* Recuperado el 2016, de www.ietf.org: https://www.ietf.org/archive/id/draft-ietf-oauth-json-web-token-13.pdf

Kunda, K., y Chihana, S. (2017). Web Server Performance of Apache and Nginx. *Computer Engineering and Intelligent Systems*, 8(2), 43-51. Recuperado el 2018, de https://www.researchgate.net/publication/329118749_Web_Server_Performance_of_Apache_and_Nginx_A_Systematic_Literature_Review

Miller, L. (2017). *Public PaaS for Dummies* (2ª edición especial Oracle ed.). NewJersey: John Wiley & Sons. Recuperado el 2019, de http://www.oracle.com/us/products/applications/oracle-saas-for-dummies-2ndedition-4409697.pdf

Nayyar, A., y Puri, E. (2016). A Review of Intel Galileo Development Board's Technology. *Journal of Engineering Research and Applications*, 6(3), 34-39. Recuperado el 2017, de https://www.researchgate.net/publication/305671432_A_Review_of_Intel_Galileo_Development_Board's_Technology

Panetta, K. (18 de Octubre de 2016). *Gartner's Top 10 Strategic Technology Trends for 2017*. Recuperado el 2017, de gartner.com: https://www.gartner.com/smarterwithgartner/gartners-top-10-technology-trends-2017/

Richardson, L. y otros. (2013). *RESTful Web APIs.* Sebastopol, CA: O'Reilly Media.

Scull, B. (2016). *Intel Galileo Controlled CNC Plotter*. Recuperado el 2017, de instructables.com: https://www.instructables.com/id/Intel-Galileo-Controlled-Cnc-Plotter/

Techtarget. (Agosto de 2014). *techtarget.com*. Recuperado el 2017, de https://searchdatacenter.techtarget.com/es/foto-articulo/2240224759/10-definiciones-de-modelos-de-servicios-en-la-nube-que-debe-conocer/9/XaaS-cualquier-cosa-como-servicio

van Rossum, G. y Kubica, M. (28 de Setiembre de 2010). *HOWTO Use Python in the web.* (J. Fred L. Drake, Ed.) Recuperado el 2017, de https://docs.python.org/3.0/howto/webservers.html: https://www.rose-hulman.edu/class/csse/archive/csse120-old/csse120-old-terms/201110robotics/Resources/python-3.1.2-docs-pdf-letter/howto-webservers.pdf

Wikipedia. (2017). *Comparison of web server software: Features*. Recuperado el 2017, de wikipedia.org: https://en.wikipedia.org/wiki/Comparison_of_web_server_software

Yanev, K. (2017). *Pronterface*. Obtenido de pronterface.com: https://github.com/kliment/Printrun

APÉNDICE

ANEXO A: MÓDULO RPC

El paquete de la aplicación Printrun posee un módulo, básico, previsto para conexiones mediante mecanismos RPC.

El mismo ha sido modificado y ampliado para que pueda ser usado en colaboración con la API REST diseñada.

A continuación, el listado del código fuente correspondiente:

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

# Este archivo forma parte del paquete API REST Puma3D
# Es una modificación del archivo fuente correspondiente
# al paquete Printrun,
#
# Este módulo modificado respeta los términos de licencia GNU/GPL
# como ha sido publicado por la Free Software Foundation.
#
# Para más detalles, ver <http://www.gnu.org/licenses/>.

from SimpleXMLRPCServer import SimpleXMLRPCServer
from threading import Thread
import socket
import logging

from .utils import install_locale, parse_temperature_report
install_locale('pronterface')

RPC_PORT = 7978
```

```
class ProntRPC(object):

    server = None

    def __init__(self, pronsole, port = RPC_PORT):
        self.pronsole = pronsole
        used_port = port
        while True:
            try:
                self.server = SimpleXMLRPCServer(("localhost", used_port),
                                                 allow_none = True,
                                                 logRequests = False)
                if used_port != port:
                    logging.warning(_("RPC server bound on non-default port %d") % used_port)
                break
            except socket.error as e:
                if e.errno == 98:
                    used_port += 1
                    continue
                else:
                    raise
```

```
        self.server.register_function(self.get_status, 'status')
        self.server.register_function(self.set_extruder_temperature,'settemp')
        self.server.register_function(self.set_bed_temperature,'setbedtemp')
        self.server.register_function(self.load_file,'loadfile')
        self.server.register_function(self.start_print,'startprint')
        self.server.register_function(self.pause_print,'pauseprint')
        self.server.register_function(self.resume_print,'resumeprint')
        self.server.register_function(self.start_print,'startprint')
        self.server.register_function(self.send_home,'sendhome')
        self.server.register_function(self.connect,'connect')
        self.server.register_function(self.disconnect,'disconnect')
        self.server.register_function(self.send,'send')
        self.server.register_function(self.reset,'reset')
        self.server.register_function(self.off_printer, 'offprinter')
        self.server.register_function(self.on_cooler,'oncooler')
        self.server.register_function(self.off_cooler,'offcooler')
        self.server.register_function(self.go_center_xy,'gocenterxy')

        self.thread = Thread(target = self.run_server)
        self.thread.start()

    def run_server(self):
        self.server.serve_forever()

    def shutdown(self):
        self.server.shutdown()
        self.thread.join()


def get_status(self):
    if self.pronsole.p.printing:
        eta = self.pronsole.get_eta()
        z = self.pronsole.curlayer
        duration, layers = self.pronsole.fgcode.estimate_duration()
    else:
        eta = None # y coincide con p.printing False
        z = ''
        duration = ''
        layers = ''

    temps = parse_temperature_report(self.pronsole.tempreadings) if self.pronsole.tempreadings else None

    if self.pronsole.filename:
        filename = self.pronsole.filename
        gcodes = len(self.pronsole.fgcode)
    else:
        filename = ''
        gcodes = ''

    return {'online': self.pronsole.p.online,
            'filename': filename,
            'gcodes': gcodes,
            'duration': duration,
            'layers': layers,
            'z': z,
            'eta': eta,
            'temps': temps
            }


    def set_extruder_temperature(self, targettemp):
        logging.info('Orden RPC: temperatura del Hotend a ' + targettemp)
        self.pronsole.p.send_now("M104 S" + targettemp)

    def set_bed_temperature(self,targettemp):
        logging.info('Orden RPC: temperatura de la cama a ' + targettemp)
        self.pronsole.p.send_now("M140 S" + targettemp)
```

```
def load_file(self,filename):
    self.pronsole.do_load(filename)

def start_print(self):
    logging.info('Orden RPC: ' + command)
    self.pronsole.do_print('')

def pause_print(self):
    logging.info('Orden RPC: ' + command)
    self.pronsole.do_pause('')

def resume_print(self):
    if self.pronsole.p.online:
        logging.info('Orden RPC: ' + command)
        self.pronsole.do_resume('')

def send_home(self, command):
    logging.info('Orden RPC: ' + command)
    self.pronsole.do_home(command)

def connect(self, port):
    self.pronsole.do_connect(port)

def disconnect(self):
    self.pronsole.do_disconnect('')

def send(self, command):
    logging.info('Orden RPC: ' + command)
    self.pronsole.do_move(command)


def reset(self):
    logging.info('Orden RPC: Reset')
    self.pronsole.do_reset('')

def off_printer(self):
    logging.info('Orden RPC: APAGADO TOTAL')
    self.pronsole.do_off('')

def on_cooler(self):
    self.pronsole.p.send_now('M106 S127')
    logging.info('Orden RPC: Cooler ON')

def off_cooler(self):
    self.pronsole.p.send_now('M107')
    logging.info('Orden RPC: Cooler OFF')

def go_center_xy(self, comando):
    logging.info('Orden RPC: ir al centro X-Y ')
    self.pronsole.p.send_now(comando)
```

ANEXO B: LANZAMIENTO DE LA PLATAFORMA

Se describen aquí los pasos a seguir durante la preparación de la plataforma de pruebas, previos al trabajo de conexión y uso desde el cliente Web.

Este es un aspecto importante a tener en cuenta para lograr un acceso sin dificultades al control de la impresora 3D.

- Encender la placa SBC Galileo Gen 2 (es indistinto si encuentra conectado el cable de red Ethernet o se conecta luego),
- Conectar el host USB a la placa,
- Conectar la cámara de video al host USB,
- Conectar la impresora 3D al host USB,
- Encender la impresora 3D,
- Lanzar la ejecución de la aplicación mjpg_streamer para iniciar el servidor de video,
- Lanzar la ejecución de la aplicación Printrun (Host de Impresión), a través de su versión de consola denominada pronsole,
- Acceder a la aplicación cliente mediante el navegador Web, usando el URL: http://192.168.1.99/static
- Escribir el nombre de usuario y la clave específicos (por defecto para pruebas: cesar/cesar), lo cual permite acceder a la interfaz de monitoreo y control de la impresora 3D.