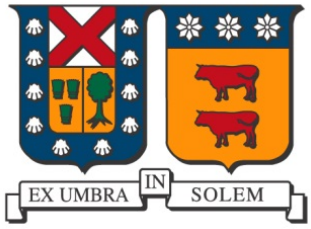
UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
VALPARAÍSO - CHILE



“ESTUDIO APLICADO DE TECNOLOGÍAS WEB REACTIVAS PARA LA MANIPULACIÓN DE COMPONENTES DE CONTROL DISTRIBUIDO”

HERNÁN RAÚL HERREROS NIÑO

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN INFORMÁTICA

Profesor Guía: Mauricio Araya López
Profesor Correferente: Horst von Brand

Agosto - 2019

DEDICATORIA

A mis padres, que siempre me apoyaron incondicionalmente durante esta difícil etapa universitaria.

Esto es gracias a ustedes y para ustedes.

AGRADECIMIENTOS

A mis amigos, por siempre transformar ocasiones de “dolor y sufrimiento” en risas y buenos momentos. Gracias por esas tardes de estudio que siempre terminaban en ocio y comida.

A mi familia, por ser comprensivos y soportar mis momentos de estrés y cansancio. Por confiar en mis decisiones y apoyarme a como dé lugar.

A mi profesor guía y profesor correferente, que sin su ayuda y experiencia, terminar esta memoria hubiera sido extremadamente difícil.

A los miembros del CSRG, por proveerme de los equipos y conocimientos necesarios para avanzar en un camino tan desolador como lo es el de ACS.

A todos los que alguna vez formaron parte de mi vida en estos últimos años, gracias.

RESUMEN

Resumen— El presente documento tiene como objetivo detallar el trabajo realizado para estudiar la factibilidad de actualizar las interfaces gráficas de las aplicaciones de ACS. En vista de la futura instalación del nuevo observatorio CTA, se realizó un estudio aplicado mediante el desarrollo de un prototipo funcional, el cual incluye los requerimientos básicos del software. En el desarrollo se tomó en cuenta la experiencia de usuario y la capacidad preatentiva de las personas, con el fin de disminuir la dificultad y los tiempos necesarios para la realización de tareas. En base a las pruebas realizadas, comparando las actuales aplicaciones de ACS con el prototipo desarrollado, se logró demostrar en base a resultados empíricos que las mejoras implementadas en la nueva aplicación disminuyen los tiempos necesarios para realizar tareas. Los resultados obtenidos podrán guiar la toma de decisiones del observatorio CTA en cuanto a la implementación de mejoras en ACS.

Palabras Clave— ACS; Aplicación Web; Reactivo; Experiencia de Usuario; Prototipo

ABSTRACT

Abstract— The main goal of this document is to explain the study carried out to prove the feasitilibty of changing the current user interfaces of the software used by the ALMA Observatory. Due to the construction of the new CTA Observatory, and applied study was carried out developing a working prototype that includes the main features of the ALMA software. During the development, it was taken into consideration the user experience and the preatentive capabilites of the people, in order to decrease the difficulties and the required times when using the software. Based on the performed test, it was possible to demonstrate with empirical results that the improvements implemented in the prototype decreased the usage times. The obtained results will be useful in the decision making of the CTA Observatory when software improvements are required.

Keywords— ACS; Web Application; Reactive; User Experience; Prototype

GLOSARIO

ALMA: Atacama Large Millimeter Array
ACS: ALMA Common Software
API: Application Programming Interface
CORBA: Common Object Request Broker Architecture
CSRG: Computer Systems Research Group
CTA: Cherenkov Telescope Array
REST: Representational State Transfer
UTFSM: Universidad Técnica Federico Santa María
GUI: Graphical User Interface
IDL: Interface Definition Language
DOM: Document Object Model
IOR: Interoperable Object Reference
OMG: Object Management Group
ORB: Object Request Broker

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

La institución a cargo del próximo observatorio Cherenkov Telescope Array (CTA) a instalarse en el norte de Chile y La Palma, Islas Canarias, se encuentra en un proceso de renovación de tecnologías, en donde deberán resolver el cómo y el por qué actualizar las interfaces gráficas del software que utilizarán para la administración y control de sus telescopios. El software en cuestión, llamado ALMA Common Software (ACS), es un framework que fue presentado en el año 2001 por el observatorio ALMA y sus componentes gráficos no han sido drásticamente actualizados desde su creación. ACS posee una gran cantidad de componentes gráficos desarrollados en Swing, una biblioteca gráfica de Java, que para ser usados, se debe poseer un equipo con ACS instalado, o en su lugar, acceder de manera remota a algún equipo que si lo posea, mediante SSH y la redirección gráfica del servidor X de la máquina o compartiendo el escritorio de manera virtual. Por otro lado, es bastante probable que la obsolescencia de sus interfaces genere a futuro problemas de compatibilidad, mantenimiento y soporte. Además, la experiencia de usuario que estas aplicaciones entregan es de muy bajo nivel, disminuyendo la efectividad y facilidad de uso. A raíz de lo anterior, se realizó un estudio sobre la factibilidad técnica de actualizar dichas aplicaciones Java, utilizando tecnologías web modernas que permitan la implementación de experiencias reactivas, mediante la elaboración de un prototipo funcional que permitió dilucidar la complejidad de la implementación de la propuesta de solución y comprobar el incremento en la efectividad de la realización de tareas básicas. Para la elaboración del prototipo se seleccionó un conjunto de tecnologías que permitieran un desarrollo sencillo y rápido, además de cumplir con los requerimientos de esta propuesta de solución, es decir, el uso de tecnologías web de tipo reactivo.

Objetivo general

Realizar un estudio sobre la factibilidad de desarrollar interfaces web que entreguen experiencias reactivas con tecnologías actuales para la visualización de información y manipulación de componentes de un sistema de control distribuido como lo es ACS. Se busca proponer una solución tecnológica y elaborar un prototipo funcional, utilizando tecnologías que permitan cumplir con el carácter reactivo de la solución.

Objetivos específicos

- Desarrollar un diseño de interfaz reactiva basada en una de las aplicaciones de ACS.
- Elaborar documentación útil para CTA Consortium para probar la factibilidad de aplicar el estudio al próximo observatorio CTA.
- Desarrollar un prototipo funcional basado en el Object Explorer

CAPÍTULO 1
DEFINICIÓN DEL PROBLEMA

En los últimos años se ha visto un crecimiento exponencial en el desarrollo de aplicaciones web para distintos fines, desde el entretenimiento, con plataformas capaces de ofrecer servicios de películas, series y música, hasta la ciencia, con plataformas que permiten mostrar de manera efectiva grandes cantidades de información. Gracias a la implementación de bibliotecas especialmente diseñadas para construir visualizaciones, es posible mostrar al consumidor datos de una manera sencilla, tanto para el consumidor como para el desarrollador. Según estudios realizados en los últimos años, se ha visto un incremento en el uso de tecnologías web en empresas, con el fin de mejorar la eficiencia de sus procesos y entregar un mejor producto a sus clientes. [Brennan, 2015]

Múltiples empresas e instituciones de gran importancia han puesto su confianza en estas tecnologías, tales como Spotify y Netflix, quienes manejan una gran cantidad de datos sensibles y protegidos por derechos de autor. La seguridad y efectividad de la infraestructura es primordial a la hora de ofrecer este tipo servicios y las tecnologías web, a pesar de los múltiples desafíos y riesgos que conlleva la distribución de contenido a través de internet, han logrado evidenciar que ofrecen un medio confiable, seguro y efectivo. [Aswani y de Larquier, 2019]

Y es que hoy en día las aplicaciones web no sólo tienen como objetivo entregar información a quienes lo requieran, sino que también están destinadas a ser usadas como herramientas que ayudan a resolver problemas y necesidades de gran complejidad en distintos rubros. Su extensivo uso en los últimos años ha evidenciado el gran potencial que estas tienen, lo que lleva a pensar y analizar si es necesario migrar aplicaciones que fueron construidas para ser usadas en computadores específicos y de manera local a páginas web dinámicas y modernas que puedan ser accesibles por cualquier usuario dentro de la red, ya sea pública o privada. Claramente, esto conlleva ventajas y desventajas que deberán ser puestas en la balanza por aquellas instituciones que quieran adentrarse en esta área.

A raíz de de la instalación del próximo observatorio que tendrá lugar en Chile y La Palma, Islas Canarias, el Cherenkov Telescope Array (CTA), la institución se encuentra en un proceso de renovación y actualización del software que se utilizará para la administración y control de sus telescopios, por lo que se busca conocer que tan factible es realizar dicho cambio y a que tipo de tecnologías recurrir. Este caso en particular es la razón de realizar un estudio aplicado de factibilidad, proponiendo el uso de tecnologías web como principal y única opción.

1.1. Descripción de la institución

Fue en el año 2005 cuando un grupo de científicos con la idea de seguir ampliando los conocimientos sobre el universo, propusieron la creación de una instalación detectora de rayos

gamma llamada Cherenkov Telescope Array (CTA). La institución fundada el año 2014 bajo el nombre de CTA Observatory gGmbH (CTAO gGmbH) [Hofmann, 2017] [1], trabaja en estrecha cooperación con el CTA Consortium (CTAC) [2], compuesto por alrededor de 1.350 miembros pertenecientes a 32 países, quienes están a cargo de dirigir los objetivos científicos, además de participar del diseño del arreglo de telescopios y en el suministro de los componentes necesarios para la construcción. Por otro lado, el CTAO está liderado por el CTA Council, el cual está compuesto por instituciones accionistas de nueve países diferentes: Austria, República Checa, Francia, Alemania, Italia, Japón, España, Suiza y Estados Unidos.

Más en detalle, uno de los objetivos del proyecto es la instalación de arreglos de telescopios en dos locaciones, sur y norte, para poder abarcar un área más grande del cielo, es por esto que entre los años 2010 y 2013, el CTA Consortium condujo un programa para la búsqueda de sitios de máxima calidad que se adaptaran a las necesidades de los telescopios [Hofmann, 2017]. Como resultado, para el sitio del sur se iniciaron negociaciones con el European Southern Observatory (ESO) para hacer uso del Cerro Paranal en Chile. Para la ubicación del norte, se negoció con el Instituto de Astrofísica de Canarias (IAC) para la instalación del arreglo en la isla La Palma en España.

En la actualidad, el proyecto se encuentra en fase de pre-producción, en donde los fondos de inversiones deben ser asegurados y los sitios geográficos deben ser preparados para avanzar a la siguiente fase [3] (Ver Figura 1).

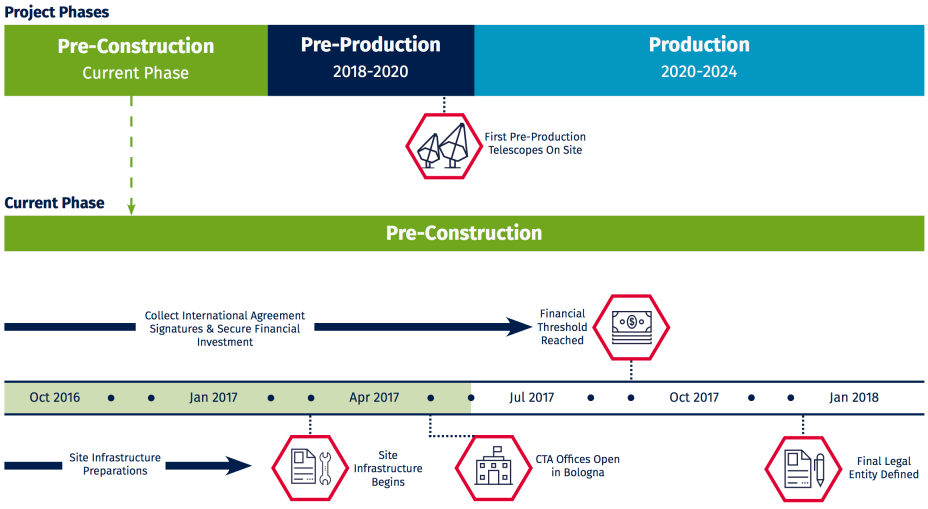


Figura 1: Fases del proyecto CTAO.
Fuente: CTA Observatory.

[1] Página web oficial: www.cta-observatory.org

[2] CTA Consortium: www.cta-observatory.org/about/cta-consortium/

[3] A mayo de 2019, las fases del proyecto se encuentran atrasadas. Actualmente la fase de pre-producción se desarrollará entre 2019 y 2021. La fase de producción de 2021 a 2025

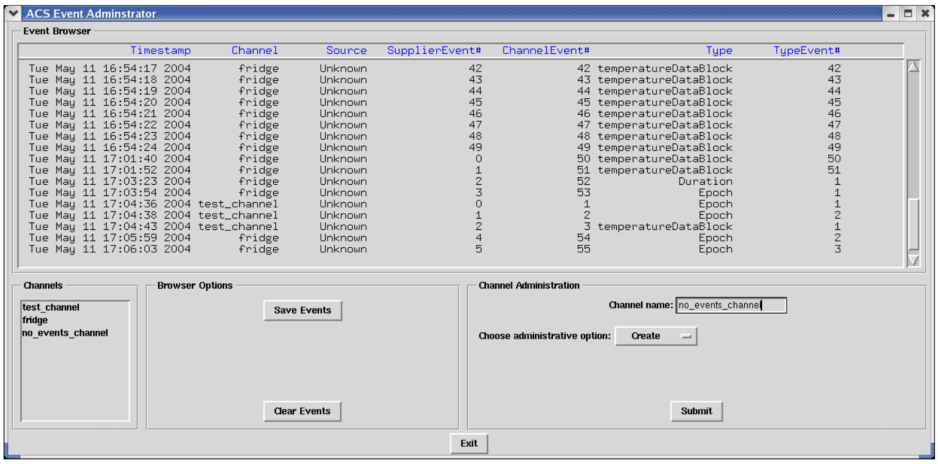


Figura 2: Interfaz gráfica del Event Browser
Fuente: ALMA Common Software Overview [Chiozzi y Šekoranja, 2004].

1.2. Software para el control de telescopios

El software seleccionado por el CTAO para el el desarrollo de su sistema de control de sus telescopios fue el ALMA Common Software (ACS), el cual es una herramienta que trabaja sobre CORBA [4] para el desarrollo de sistemas distribuidos basados en el paradigma de componentes/contenedores [Chiozzi et al., 2005]. Una de sus características, es la capacidad de abstraer la complejidad del middleware CORBA, haciendo de este un entorno de programación mucho más sencillo. Además, es posible crear componentes de ACS en tres diferentes lenguajes: C++, Java y Python.

En la actualidad, ACS se encuentra en fase de producción en el observatorio ALMA, por lo que no está en sus planes realizar actualizaciones importantes. Considerando que fue anunciado en 2001, es evidente que ciertos aspectos de ACS son obsoletos, y otra gran cantidad puede llegar a serlo en un futuro próximo, el cual, a modo general, es uno de los problemas que intenta resolver esta memoria.

1.3. Problema a resolver

A lo largo de los años se ha podido evidenciar que en el área del desarrollo de software, desde el punto de vista del usuario, las interfaces gráficas han tenido cambios significativos. Una de las principales razones de estos avances son las mejoras en la experiencia de usuario y la estética con la que la institución se identifica, provocando que las interfaces sean un elemento diferenciador en el desarrollo de software destinado al usuario común, quien no

[4] Este estándar permite comunicar diversos componentes de software escritos en diferentes lenguajes de programación y que corren en diferentes computadores.

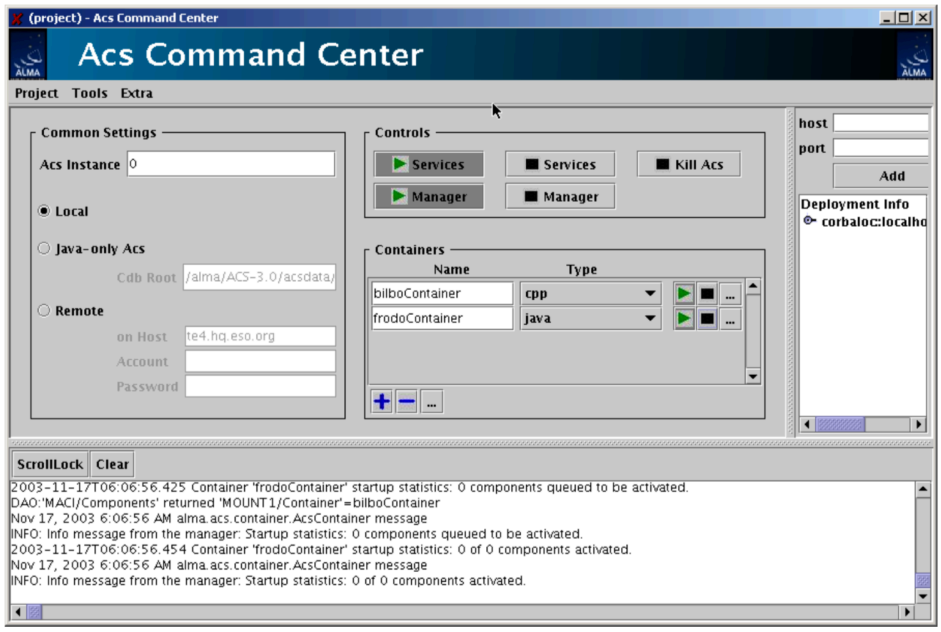


Figura 3: Interfaz gráfica del componente Command Center
Fuente: ALMA Common Software Overview [Chiozzi y Šekoranja, 2004].

tiene grandes conocimientos en computación y requiere de una interfaz gráfica para ejecutar sus tareas. Por otro lado, dejando a un lado la estética, las tecnologías con las que se construyen estas interfaces han cambiado bastante, ampliando sustancialmente el abanico de posibilidades. Con cada herramienta creada por la comunidad, los desarrolladores pueden crear aplicaciones más fácilmente, permitiendo crear aplicaciones más complejas en el mismo tiempo. De esta forma, se da la posibilidad de experimentar y hacer mucho más intuitiva la interacción entre usuario y máquina. Es aquí donde entran en juego las aplicaciones de tipo reactivas, en las cuales las acciones que el usuario realiza tienen un efecto instantáneo sobre los datos que se visualizan, haciendo que la aplicación entregue un valor agregado a la interacción e información mostrada. Además, es posible usar tecnologías que permitan el acceso a estas interfaces desde cualquier parte a través de internet.

Las interfaces gráficas de usuario son necesarias para abstraer al usuario de la complejidad que implica el funcionamiento de una aplicación. Mientras más intuitiva es la interfaz, la experiencia de usuario incrementa. De esta forma, mientras menor sea la curva de aprendizaje, mejor es la interfaz. ACS es un software muy complejo de control distribuido. En efecto, ACS como tal es una abstracción de la complejidad del middleware CORBA. Las interfaces de ACS son de gran utilidad para ayudar a los astrónomos y colaboradores de ALMA a cumplir con sus deberes sin la necesidad de entender de programación o administración de sistemas.

ACS posee una serie de interfaces gráficas que fueron construidas en Swing [5], una biblioteca

[5]Biblioteca gráfica Swing: `https://users.dcc.uchile.cl/~lmateu/CC60H/Trabajos/edavis/swing.html`

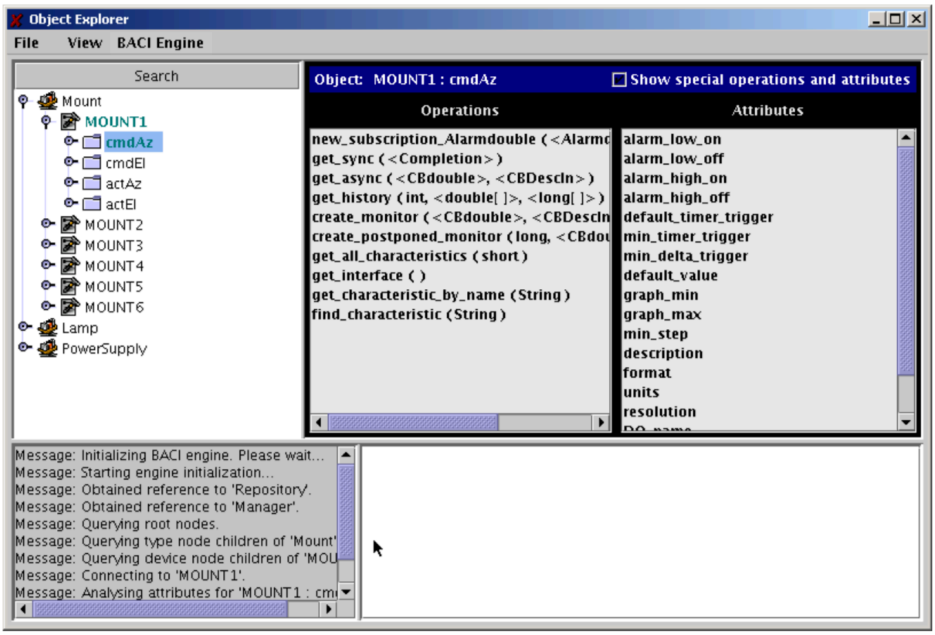


Figura 4: Interfaz gráfica del componente Object Explorer
Fuente: ALMA Common Software Overview [Chiozzi y Šekoranja, 2004].

para la construcción de GUI's en Java, destinadas al control y administración de telescopios.

Las principales utilidades gráficas de ACS pueden ser definidas de la siguiente forma:

- **Command Center** (Figura 3): Utilizada para iniciar y detener servicios de ACS, como también administrar contenedores.
- **Object Explorer** (Figura 4): Usado para navegar a través de la jerarquía de componentes del sistema.
- **Logging User Interface** (Figura 6): Usada para mostrar mensajes del sistema de registros.
- **CDB Browser** (Figura 5): Usada para navegar por las configuraciones de la base de datos en tiempo de ejecución.
- **Event Browser** (Figura 2): Utilizada para navegar por los eventos generados por las APIs de eventos de ACS.

Como se puede apreciar de las figuras 3, 4 y 6, las interfaces gráficas expuestas no son lo bastante intuitivas, requiriendo que el usuario deba leer con atención a que corresponde cada elemento. Este tipo de interfaces no permiten al usuario recurrir a su capacidad visual preatentiva [Treisman, 1985], por lo que la realización de tareas se vuelve más lenta y agotadora. El proceso analítico visual de un ser humano está dividido en etapas, en que la primera

de ellas, la persona es capaz de reconocer inconscientemente características básicas sin un esfuerzo adicional, lo que le permite captar más fácilmente la información [Treisman, 1985]. Algunas de las características preatentivas son: Colores, formas, posición espacial y movimiento. A raíz de esto, es necesario codificar de manera efectiva la información que debe ser mostrada en una aplicación web, con el fin de entregar una buena experiencia de usuario, permitiendo que los usuarios puedan desempeñar sus tareas efectivamente.

Por otro lado, la calidad visual de estas aplicaciones no cumplen con lo acostumbrado por las aplicaciones web de hoy en día, desaprovechando las variadas técnicas que permiten facilitar la visualización y manejo de información. Al haber sido desarrolladas en una versión antigua de Swing, las posibilidades de mejorarlas e incrementar su efectividad están limitadas, lo que llevará eventualmente a problemas de escalabilidad y mantenibilidad. Además, actualmente no es posible actualizar de forma simple las aplicaciones nativas de ACS (Object Explorer, Command Center, etc). Debido a lo anterior, nace la necesidad de construir interfaces modernas, escalables, de fácil mantenimiento y adecuadas a la realidad actual, haciendo que la forma en que la información es visualizada y manipulada sea una experiencia mucho más enriquecedora. Otro punto importante es la ausencia de una funcionalidad remota que permita consultar estas aplicaciones en cualquier lugar sin la necesidad de instalar software específico de manera local. En pocas palabras, lo que se busca es una interfaz de tipo reactiva, remota e intuitiva.

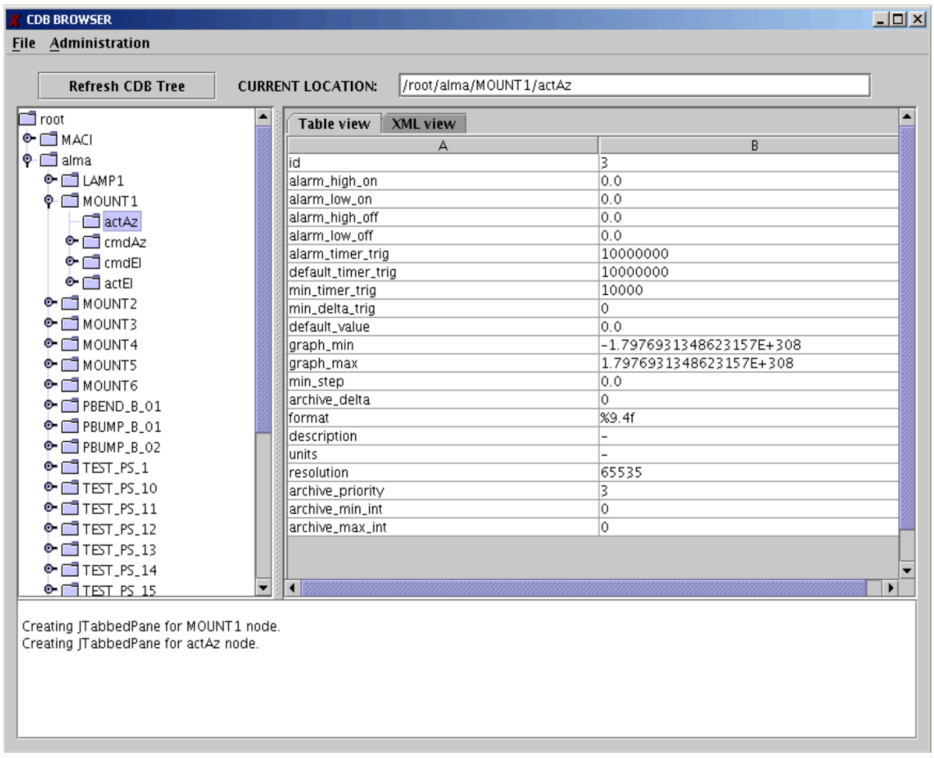


Figura 5: Interfaz gráfica del CDB Browser
Fuente: ALMA Common Software Overview [Chiozzi y Šekoranja, 2004].

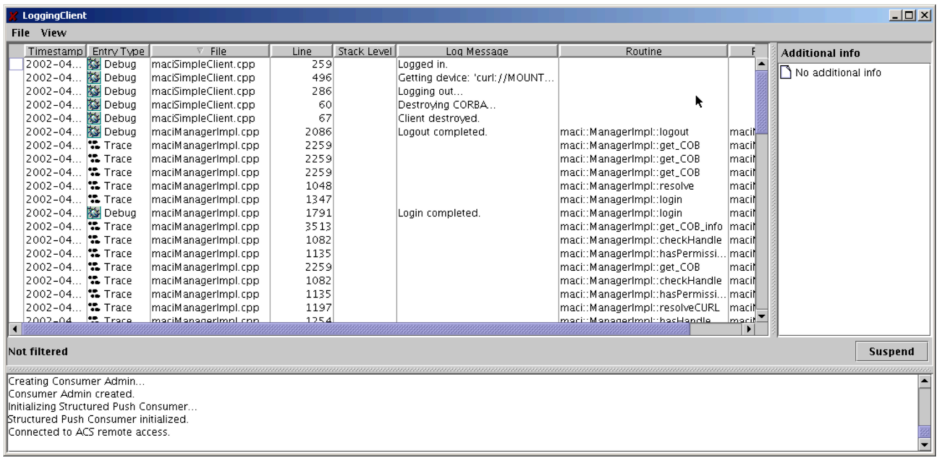


Figura 6: Interfaz gráfica del componente Logging User Interface
Fuente: ALMA Common Software Overview [Chiozzi y Šekoranja, 2004].

1.4. Objetivos de la memoria

En base a lo expuesto en el capítulo anterior, se abre la posibilidad de actualizar o reemplazar las actuales aplicaciones de ACS. Por lo tanto, esta memoria intenta mostrar el por qué utilizar tecnologías web y no aplicaciones nativas de escritorio. Además de probar la factibilidad técnica mediante la elaboración de un prototipo funcional basado en una de las aplicaciones de ACS, el Object Explorer, que será presentado en la sección 1.3.

Para sacar el máximo provecho al carácter distribuido de ACS, se propone utilizar tecnologías web que permitan entregar al usuario una experiencia reactiva, en donde cualquier acción u evento tenga un efecto inmediato en la información mostrada por la aplicación. El título de esta memoria habla de tecnologías web reactivas haciendo alusión a aquellas tecnologías que permitan lograr esto de manera efectiva y eficiente para un equipo de desarrollo, en donde los tiempos de desarrollo e implementación sean bajos. Si bien, es posible lograr estos objetivos con la mayoría de los lenguajes existentes, esta memoria se enfocará más en las herramientas, frameworks o bibliotecas, que permitan hacer más cosas con menos recursos, sin sacrificar calidad.

En cuanto a la calidad de la aplicación, se busca desarrollar una interfaz debe permitir que al usuario observar de manera instantánea los cambios que ocurren en el sistema y que pueda interactuar con los elementos de una manera más intuitiva y natural. A la vez, la aplicación debe ser lo bastante genérica y modularizable para poder incluir nuevas funcionalidades a futuro.

Finalmente, se espera que el documento elaborado, junto con los resultados obtenidos, pueda servir como guía en la toma de decisiones del observatorio CTA al momento de analizar la implementación de ACS.

1.5. Estructura del documento

El presente documento se estructura de la siguiente forma: Comienza con una introducción que contextualiza el problema a resolver, continuando con una explicación detallada del problema y la institución involucrada en el capitulo 1. Luego se da paso al Marco tecnológico en el capitulo 2, en donde se definen una serie de conceptos y tecnologías útiles, que serán utilizados en el desarrollo de la memoria.

Una vez introducido el tema y los conceptos, se procede a detallar una propuesta de solución en el capitulo 3, junto con una explicación de la metodología usada para la validación de esta solución y los resultados obtenidos en el capítulo 5. Finalmente, se cierra el documento con una conclusión en el capítulo 6 en base a la experiencia y los resultados obtenidos.

CAPÍTULO 2
MARCO TECNOLÓGICO

Antes de comenzar a definir una solución concreta al problema, será necesario establecer las bases tecnológicas y las alternativas que hay a disposición para desarrollar los distintos componentes de software que podrían ser necesarios para la actualización de los componentes de ACS.

Cuando hablamos de desarrollar aplicaciones con tecnologías web, es inevitable tener que hacer una división en la estructura del software, es por esto que en el presente capítulo se partirá haciendo una distinción entre las partes del desarrollo Back-end y Front-end [Adhikari, 2016], con el objetivo de realizar una investigación más detallada.

2.1. ALMA Common Software

ACS es el software base utilizado por el observatorio ALMA para la administración y control de su observatorio, el cual además será utilizado por el próximo observatorio CTA.

Para la construcción de componentes y el desarrollo de clientes, existen tres APIs disponibles para el desarrollador, una por cada lenguaje, C++, Java y Python, que entregan las herramientas necesarias para interactuar con ACS [Roberts, 2004].

Mediante estas APIs es posible establecer conexión con el Manager de ACS y así acceder a los Componentes y Contenedores del sistema distribuido. Adicionalmente, es posible tener acceso al Logging System [Žagar y Georgieva, 2007], Notification Channel [Pisano et al., 2009], Error System [Jeram et al., 2007], Alarm System [Caproni, 2007], entre otros.

De todos los paquetes disponibles para el lenguaje Python, existe uno llamado ACSpy [Fugate, 2003], que entrega una implementación simple de un cliente ACS, el cual permite obtener referencias de los componentes y acceder a sus métodos e información básica. Este paquete representa alrededor de un 90 % de todo el código Python de ACS.

2.2. Frameworks y bibliotecas

Un framework, en el área de software, es definido como un generador base de aplicaciones o un esqueleto que implementa las funcionalidades básicas necesarias para que una aplicación funcione. Se compone de diversas bibliotecas preconfiguradas para que puedan funcionar en conjunto. Por lo tanto, un framework es un diseño reusable por desarrolladores con el fin de acelerar y facilitar el desarrollo. [Vojislav et al., 2011]

Por otro lado, una biblioteca es una implementación de código reutilizable que cumple una o varias funcionalidades especificas. Puede ser aplicado en distintas aplicaciones junto a otras bibliotecas.

2.3. Common Object Request Broker Architecture

Tal como se mencionó en la sección 1.2, ACS trabaja sobre CORBA, un estándar definido por la OMG para el ORB, el cual es un mecanismo utilizado para invocar operaciones en otros objetos de manera remota y distribuida. [McHale, 2007] De esta forma, distintas aplicaciones, en distintos lugares y lenguajes pueden comunicarse entre si mediante invocaciones remotas, las cuales abstraen a cada una de las aplicaciones de la complejidad de la red. CORBA utiliza IDLs para definir de manera pública la API de cada uno de los objetos presentes en la red.

Para realizar la comunicación con un objeto CORBA, se utiliza un IOR, el cual es una referencia al objeto, la cual puede ser transmitida por la red de forma binaria o serializada en un string de dígitos hexadecimales. Para obtener esa referencia, se utilizan direcciones URL llamadas corbalocs, son similares a las utilizadas comúnmente en la web. Un ejemplo de corbaloc es el siguiente:

```
bcorbaloc:iiop:1.2@host1:3075/NameService
```

2.4. Tecnologías Back-end

Como se especificó en el capítulo anterior, el software ACS cuenta con soporte para tres lenguajes: C++, Java y Python. Existen diferencias entre los tres lenguajes, siendo el rendimiento una de las más importantes. A pesar de lo anterior, en este estudio el rendimiento de la comunicación con ACS no será un tema relevante, ya que unas pocas fracciones de segundo de retraso no serán perjudiciales para la experiencia de usuario. Mientras que Python ofrece un entorno simple y sin mayores complicaciones, Java y C++ entregan un ambiente robusto y con más posibilidades, debido a que las APIs de ACS para estos lenguajes están más completas que la de Python.

En la figura 7 es posible apreciar la conexión entre ACS, Back-end y Front-end.

En la actualidad existe una amplia gama de opciones para el desarrollo de software, por lo que elegir entre todas las alternativas disponibles se vuelve una tarea extensa. Por otro lado, la construcción de aplicaciones sin la ayuda de un Framework puede provocar problemas de seguridad y estabilidad si no se posee la experiencia necesaria, además los tiempos de desarrollo son elevados. Es por esto que a continuación se presentarán diversos frameworks y bibliotecas para cada uno de los tres lenguajes, seleccionados en base a popularidad y

Figura 7: Esquema básico de la comunicación entre los componentes de la aplicación.
Fuente: Elaboración propia.

comunidad, con el fin de describirlos y luego decidir cual es el que mejor se adapta a las necesidades del estudio.

2.4.1. Lenguaje C++

El lenguaje C++ fue introducido por Bjarne Stroustrup en el año 1980 bajo el nombre de "C with Classes" [Stroustrup, 1999], con la intención de extender el lenguaje C. Sus diferencias son que C++ tiene soporte para tipo de dato abstractos, programación orientada a objetos y programación genérica.

Para desarrollar soluciones web, existen algunas herramientas disponibles a través de la red, aunque no son tan conocidas como las de otros lenguajes. A continuación, se presenta una selección de cuatro herramientas que podrían ayudar a solucionar el problema planteado en el capítulo 1:

- **CppCMS**: Framework de desarrollo web de alto rendimiento, que sigue el patrón de diseño MVC, enfocado en sitios que requieren manejar grandes cantidades de solicitudes en cortos periodos de tiempo. Según su autor, una de las razones de por qué usar CppCMS es que hace al desarrollo una tarea rápida y fácil. Además, este framework cuenta con su propio lenguaje de plantillas, el cual es traducido a código C++ y luego transformado a HTML.

  Por otro lado, el autor recomienda recurrir a otros frameworks si el objetivo del desarrollador es construir un sitio web que reciba pocas solicitudes por segundo [Beilis, 2008].

- **C++ Web Toolkit**: Es una biblioteca gratuita para programar aplicaciones web modernas que incorpora muchas similitudes con herramientas de desarrollo de interfaces gráficas para otras plataformas, por lo tanto, los desarrolladores pueden diseñar las interfaces web tal como en otras plataformas (Windows, Linux, etc). Aquí, una aplicación esta definida como una jerarquía de Widgets, en donde cada uno encapsula la vista y su comportamiento, consume y produce eventos y hasta puede participar del manejo de las URL. [Dumon y Deforche, 2008]

- **Beast**: Es una biblioteca de código abierto con soporte de HTTP y Websockets construida sobre Asio, ambas pertenecientes al mismo conjunto de bibliotecas llamado Boost. [Falco, 2016]
- **WebSocket++**: Es una biblioteca Open Source que implementa el protocolo Websocket RFC6455 [6]. Esta biblioteca permite integrar clientes y servidores de Websocket en programas escritos en C++ [Thorson, 2015].

Tabla 1: Comparación de alternativas C++. Información extraída el 07/05/2019 desde repositorios públicos de Github

Fuente: Elaboración Propia.

	CppCMS	C++ Web Toolkit	Beast	WebSocket++
Número de contribuidores en Github	2	44	75	41
Rating en GitHub	218	829	2.007	3.020

En la Tabla 1 se recopilaron datos correspondientes a los repositorios de Github de cada uno de los 3 proyectos con la intención de mostrar su popularidad y apoyo de la comunidad. Github es una plataforma colaborativa que sirve para almacenar el código fuente de proyectos privados y públicos utilizando el sistema de control de versiones Git. Cada uno de estos repositorios posee un rating de estrellas, en donde cada una representa la aprobación de un usuario, muy similar al botón Me Gusta de Facebook. A modo de referencia para la tabla comparativa, el repositorio con más estrellas al 07/05/2019 es freeCodeCamp con 302.411 estrellas.

2.4.2. Lenguaje Java

Entre las características que describen al lenguaje Java, se destaca su simpleza respecto a C++, intentando mantener un sistema comprensible (Por ejemplo: en 1995 la mayoría de los desarrolladores utilizaba C o C++). Además, es orientado a objetos, robusto y neutral a la arquitectura, permitiendo que los programas escritos puedan ser ejecutados en cualquier máquina independiente de la CPU o el sistema operativo [Gosling, 1995].

Debido a su alta popularidad y gran uso en la industria, solo se considerará Spring, el cual es un framework de código abierto que posee módulos para la construcción de aplicaciones web basadas en el patrón de diseño MVC y aplicaciones reactivas. Este último tiene la intención de crear aplicaciones no bloqueantes. [Spr, 2017].

[6] RFC6455: Identificador que fue asignado al protocolo Websocket por la internet Engineering Task Force (IETF)

2.4.3. Lenguaje Python

Python es un lenguaje de programación multi-paradigma e interpretado desarrollado por Guido Van Rossum. Soporta los paradigmas de orientación objetos, imperativo y funcional. Su filosofía es hacer que el código sea fácilmente leíble por las personas. [2007]

A continuación, se presentan los tres frameworks más conocidos para el desarrollo de aplicaciones web:

- **Django**: Es un framework de código abierto que sigue el patrón de diseño Modelo-Vista-Controlador (MVC), en el cual se divide la estructura del software en tres secciones principales [Curry y Grace, 2008]. Fue creado con la intención de crear páginas orientadas al consumo de contenido, tales como tiendas virtuales o de noticias, además de evitar tener que construir páginas desde cero una y otra vez. Favorece la reutilización y además ofrece un administrador visual del modelo de base de datos [Holovaty et al., 2007].
- **Flask**: Se define como un pequeño framework, suficientemente pequeño como para ser llamado ”micro framework”. Fue creado para ser una herramienta extensible, por lo que el framework como tal solo ofrece los servicios básicos, mientras que las extensiones ofrecen el resto. Flask no tiene soporte nativo para acceder a bases de datos o validar formularios. A diferencia de otras herramientas como Django en donde el desarrollador se ve obligado a utilizar las extensiones que ya vienen incluidas, Flask es una herramienta modular, que permite al desarrollador elegir que extensiones instalar [Grinberg, 2014].
- **Pyramid**: Es un framework de código abierto que intenta hacer el desarrollo web más simple. Al igual que Flask, solo intenta resolver los problemas fundamentales como el mapeo de las URL, el uso de plantillas y seguridad, haciendo que Pyramid sea una herramienta simple y minimalista, en donde el desarrollador no se ve obligado a entender conceptos que no usará. Pyramid provee lo básico y el desarrollador toma la decisión de que extensiones añadir o no [McDonough, 2011].

La tabla 2 muestra la comparación de los tres frameworks nombrados. A diferencia de las herramientas de C++ nombradas en la sección 2.4.1, la popularidad y actividad por parte de la comunidad crece.

En vista de lo expuesto sobre los tres lenguajes que soporta ACS, la información mostrada en esta sección debe se tratada considerando que esta es una de las partes vitales para el éxito de la solución. Es aquí, en el Back-end, donde se debe realizar la comunicación con el framework ACS. La decisión de cual biblioteca y/o lenguaje usar puede marcar la diferencia entre un proyecto factible o no.

Tabla 2: Comparación de alternativas Python. Información extraída el 07/05/2019 desde repositorios públicos de Github

Fuente: Elaboración Propia.

	Django	Flask	Pyramid
Tipo	Framework MVC	Micro Framework	Web Framework
Primer lanzamiento	2005	2010	2008
Número de contribuidores en Github	1.734	507	263
Rating en GitHub	41.333	43.836	3.117

2.5. Tecnologías Front-end

Todo lo que funciona del lado del cliente es parte del Front-end, es decir, todo lo que se ejecuta en el navegador, por lo tanto, al momento de desarrollar, se debe tener en consideración a los usuarios en todo momento ya que se deben definir las formas de interacción que estos tendrán con la aplicación. Como el propósito de esta memoria es mejorar la efectividad del Object Explorer (Figura4), la cual es una aplicación que no tiene una gran cantidad de vistas, se hace natural pensar en el desarrollo de una aplicación single-page para aprovechar su mejor rendimiento y de esta forma entregar una experiencia de usuario mejorada. Una aplicación single-page (SPA) es una aplicación que es entregada al navegador y no recarga la página durante su uso [Mikowski y Powell, 2013], por el contrario, solo recarga lo que se necesita, por ejemplo, paneles específicos del sitio.

Al usar SPA, cambia la estructura de la aplicación, ya que la lógica de negocios se traspasa desde el Back-end al Front-end en la mayoría de los casos. En plataformas web tradicionales, cuando se hace clic sobre un enlace, toda la página es recargada para mostrar la nueva información, aún cuando hay elementos que se mantienen igual. En el caso de una SPA, solo se redibuja lo que es necesario, por lo que los tiempos de carga son menores, mejorando considerablemente la experiencia de usuario.

En la figura 8 se puede ver como cambian las responsabilidades de la arquitectura desde una aplicación tradicional a una aplicación SPA, en donde la lógica del negocio y la generación de HTML ahora están a cargo del software ejecutado en el cliente.

2.5.1. Lenguaje Java

Además de tener herramientas para el desarrollo de aplicaciones Back-end, Java permite la construcción de aplicaciones que son ejecutadas en el lado del cliente.

Una de las herramientas relevantes es **Google Web Toolkit** es un framework de código abier-

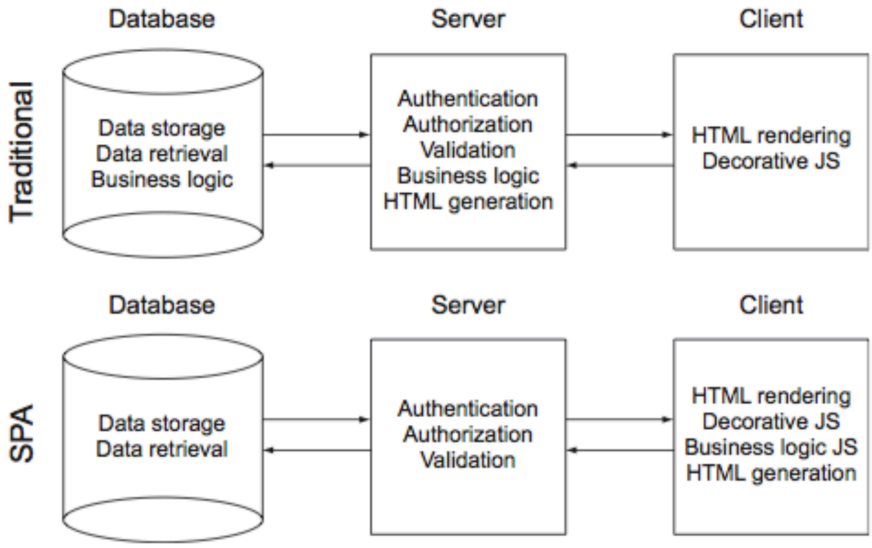


Figura 8: Diferencia entre una arquitectura tradicional y una SPA.
Fuente: Single Page Web Application with Restful API and AngularJS [Ho Ngoc, 2014].

to desarrollado por Google con la intención de construir RICH Internet Applications (RIA) [7] de alto rendimiento y gran escala, y a la vez, haciendo que sean de fácil mantención. Permite a los desarrolladores crear aplicaciones del lado del cliente, compilando el código escrito en Java a JavaScript optimizado compatible con la mayoría de los navegadores [Panirahi, 2016].

2.5.2. Lenguaje JavaScript

JavaScript es un lenguaje usado hoy en día para la construcción de una gran variedad de sistemas, incluido las aplicaciones web con avanzadas interfaces de usuario [Mikowski y Powell, 2013]. Con el tiempo han surgido una gran cantidad de herramientas que permiten enfrentar distintos paradigmas y mantener una estructura consistente en el código de las aplicaciones. A continuación, se presentarán tres herramientas dedicadas a la construcción de aplicaciones SPA, las cuales se encuentran entre las más usadas en la industria del desarrollo web.

- **AngularJS**: Es un framework Modelo-vista-controlador (MVC) de código abierto desarrollado y mantenido por Google. Permite construir interfaces interactivas y modernas mediante la extensión del código HTML con etiquetas propias. Además, entrega herramientas para la actualización de información de forma instantánea. Adopta el estándar de componentes web. [Ho Ngoc, 2014]

[7] RICH Internet Applications: Aplicación web que tiene la mayoría de las características de una aplicación de escritorio

Por otro lado, también existe Angular 2, la cual funciona con el lenguaje TypeScript [8], por lo que el framework hace la transformación de TypeScript a JavaScript en tiempo de compilación.

- **ReactJS**: A diferencia de Angular, esta es una biblioteca de JavaScript de código abierto desarrollada por Facebook. Es reconocido por ser simple y efectivo, con una curva de aprendizaje corta y fácil. Una aplicación React es una colección de componentes que manejan su propio estado y representan una vista específica de la aplicación. [Islam Naim, 2017]. Esta tecnología será vista en más detalle en la sección 3.3.
- **Vue.js**: Es una biblioteca para la construcción de interfaces web interactivas [Kyriakidis et al., 2016]. El objetivo de Vue.js proveer una API con los beneficios del enlace reactivo de datos y vistas basadas en componentes. Está enfocado en las vistas, por lo que no puede ser catalogado como un Framework propiamente tal.

Otros factores importantes que considerar, es la presencia de las herramientas en la comunidad virtual, ya que mientras mas desarrolladores las ocupen, habrá documentación actualizada y de calidad, además de bastante soporte como en sitios similares a StackOverflow. Lo anterior provoca que utilizar herramientas que cumplan con esas características sea una experiencia con menos contratiempos (Ver Tabla 3).

Tabla 3: Comparación de alternativas JavaScript. Información extraída el 07/05/2019 desde repositorios públicos de Github

Fuente: Elaboración Propia.

	AngularJS	Angular 2	React.js	Vue.js
Tipo	Framework MVW	Framework MVC	Biblioteca JavaScript	Framework MVC
Primer lanzamiento	2009	2016	2013	2014
Número de contribuidores en Github	1.595	918	1.296	273
Rating en GitHub	59.520	47.801	128.483	137.717

En la Tabla 3 se aprecia que Vue.js es el framework más popular para el lenguaje JavaScript, mientras que React.js es el segundo. Ambos son el tercer y quinto repositorio más popular a nivel global, respectivamente.

2.6. Tecnologías Adicionales

A continuación, se nombrarán algunas tecnologías que serán de ayuda en la formulación de la solución que esta memoria propone. Fueron elegidas en base a las necesidades de ACS y de lo expuesto en el capítulo 2.

[8]TypeScript: Lenguaje basado en JavaScript que realiza algunas mejoras en cuanto sintaxis y tipificado.

2.6.1. WebSockets

Uno de los objetivos de esta memoria es hacer una interfaz reactiva, es decir, que reaccione a eventos del sistema y que sean visibles inmediatamente por los usuarios. Para lograr esto, una API REST mediante HTTP con polling o long-polling [Liu y Sun, 2012] no resuelve el problema, por lo que surge la necesidad de incluir una nueva tecnología que soporte canales de comunicación bidireccionales con traspaso de mensajes instantáneos y la respuesta a eso son los WebSockets. [Fette y Melnikov, 2011]

Una de las herramientas disponibles para el desarrollo de soluciones basadas en WebSockets es Socket.io [Rauch, 2013], la cual ayuda a abstraer y facilitar la conexión con el servidor, haciendo que el desarrollador no tenga que lidiar con pérdidas de conexión y problemas asociados. Una de las grandes ventajas de usar Socket.io, es que si el protocolo de WebSockets no está disponible, internamente usará otros métodos para realizar la conexión persistente, permitiendo que una mayor cantidad de usuarios tenga acceso a la aplicación.

2.6.2. Visualizaciones con D3.js

Debido a la gran cantidad de datos que deben ser analizados y visualizados en la actualidad, es necesario contar con herramientas que permitan manejar de forma simple y efectiva tal cantidad de datos. Debido a estas necesidades, se han creado herramientas como D3.js, una biblioteca para el lenguaje JavaScript que permite generar documentos basados en datos. Da la posibilidad a los desarrolladores de crear visualizaciones dinámicas con grandes cantidades de datos. Es posible definir tipos de interacción, tipos de visualización, colores, animaciones, entre otros. La biblioteca está dividida en módulos con diferentes funcionalidades, los cuales pueden ser encontrados en los repositorios de Github. [Bao y Chen, 2014]

2.6.3. Manejo de estados con arquitectura Flux

Al crear aplicaciones web utilizando la biblioteca React para JavaScript, es posible dar cuenta de lo poco eficiente y poco mantenible que puede llegar a ser cuando la cantidad de componentes usados es muy grande. La forma en que se consume la información con una arquitectura como la MVC llevará eventualmente a una aplicación poco escalable. Para solventar este problema, Facebook desarrolló una arquitectura de aplicación centrada en un flujo de datos uni-direccional llamada Flux, la cual simplifica la forma en que se consumen los datos dentro de una aplicación [Gackenheimer, 2015]. En la arquitectura Flux, los datos son manejados fuera de los componentes de React, en stores.

Existen diversas implementaciones de la arquitectura Flux. La que ha ganado más popularidad en el último tiempo es la biblioteca Redux, que a pesar de estar basada en Flux, no es exactamente igual. A diferencia de Flux, Redux elimina el dispatcher y añade los reducers,

simplificando así su funcionamiento. [Banks y Porcello, 2017]

Para actualizar el estado dentro de un store, un componente debe ejecutar un action creator, el cual genera un action. Cada action posee un tipo y un payload, lo que indica qué debe cambiar dentro del estado. Luego, los reducers se encargan de entregar un estado modificado en base al estado anterior y el action generado.

2.7. Modelo de Contenedores y Componentes

ACS está estructurado en un modelo de Contenedor - Componente, permitiendo a los desarrolladores enfocarse en el dominio del problema y no tanto en los problemas de la ingeniería de software.

ACS intenta asemejarse a lo que hace CCM (CORBA Component Model), especificación creada por la institución a cargo de CORBA. En el año 2000, CCM aún estaba en discusión, por lo tanto, no era una opción utilizarlo.

En la actualidad, .NET y Enterprise JavaBeans (EJB) son algunas de las soluciones más comunes en la industria para el desarrollo de aplicaciones distribuidas.

Más en detalle, ACS posee un Manager, que es la entidad que se encarga de supervisar y manejar la comunicación entre componentes, entregando información bajo demanda sobre los componentes disponibles en la red. En la Figura 9, se muestra el diagrama de clases de la arquitectura de ACS, en donde se ven las interacciones que tienen los distintos elementos.

Un contenedor se encarga de instanciar y proveer un entorno a los componentes, entregando una interfaz para que estos puedan ser accedidos desde afuera y que estos puedan comunicarse al exterior del contenedor. Para lograr lo anterior, cada contenedor le entrega a sus componentes un objeto ContainerServices, con el cual pueden acceder a información del sistema y de otros componentes.

Un componente es una pieza de código que es ejecutada, la cual puede servir como cliente y/ o medio de comunicación con elementos de hardware, de esta forma, otros componentes podrán acceder a los métodos expuestos para, por ejemplo, ver información sobre hardware o incluso controlarlo. Cada componente debe implementar una interfaz ComponentLifeCycle, la cual define las siguientes operaciones: initialize, execute, cleanup. Estas operaciones son ejecutadas por los contenedores cuando es necesario activar un componente (Ver Figura 10).

Los contenedores y componentes definen sus interfaces mediante IDL, con el fin de permitir a aplicaciones implementadas en otros lenguajes acceder a sus métodos. Por otro lado, sus implementaciones son objetos CORBA.

Entonces, independiente del lenguaje de programación que se utilice en cada componen-

te, estos serán capaces de comunicarse entre si, gracias a las interfaces de comunicación provistas por los contenedores. [Chiozzi et al., 2006]

Además, existe un tipo adicional de componente llamado Characteristic Component, los cuales son una subclase de los componentes, enfocados en describir una serie de valores numéricos. Son usados para representar dispositivos físicos o lógicos, tales como sensores de temperatura, y cada uno posee propiedades y características. Cada Characteristic Component posee cero o más propiedades y cero o más características. Una propiedad corresponde a un valor (de solo lectura o lectura/escritura) que define algún estado del componente, como por ejemplo: posición, velocidad, corriente eléctrica. Por otro lado, una característica corresponde a un dato estático, entre los que se pueden encontrar: nombre, descripción, resolución, unidad. [Chiozzi et al., 2009]

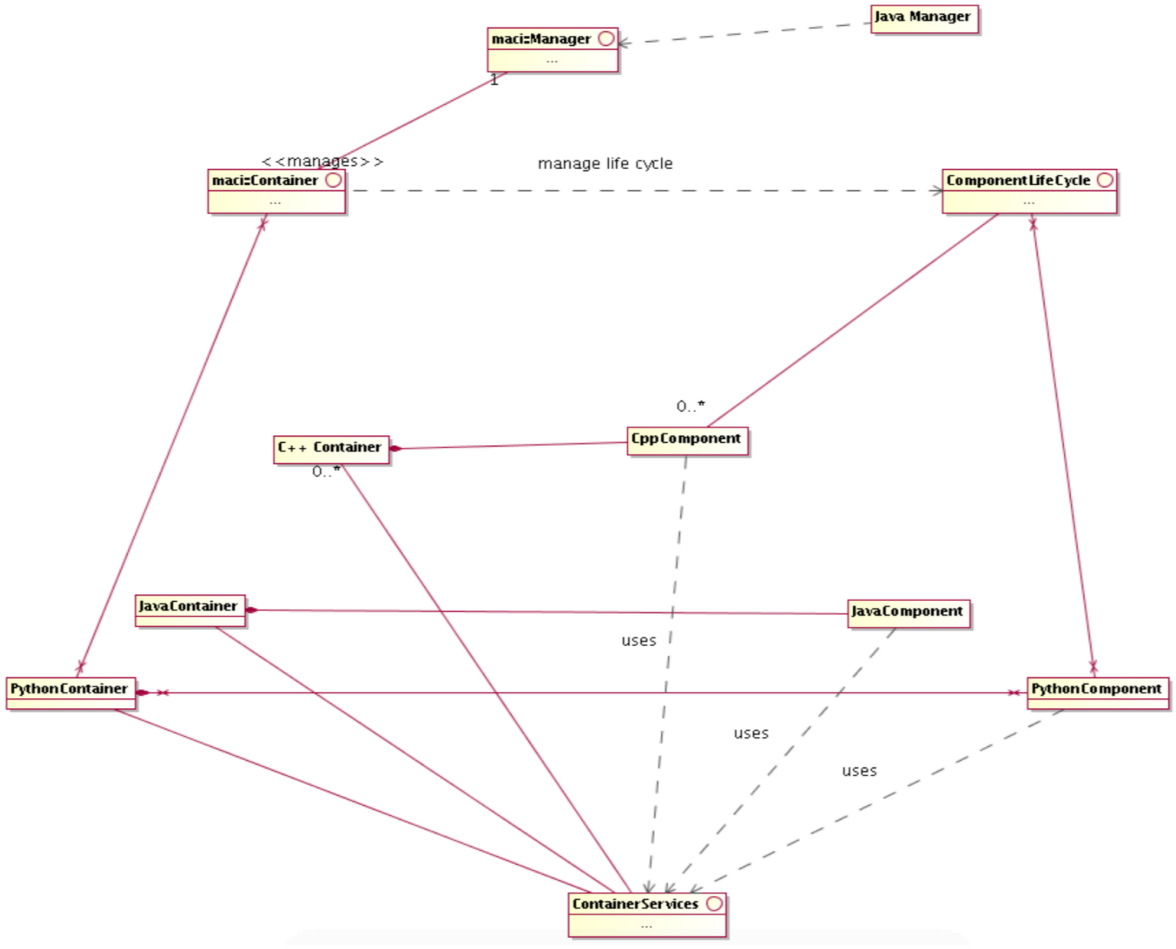


Figura 9: Diagrama de clases del modelo contenedor-componente de ACS.
Fuente: ALMA Common Software Architecture [Chiozzi et al., 2009].

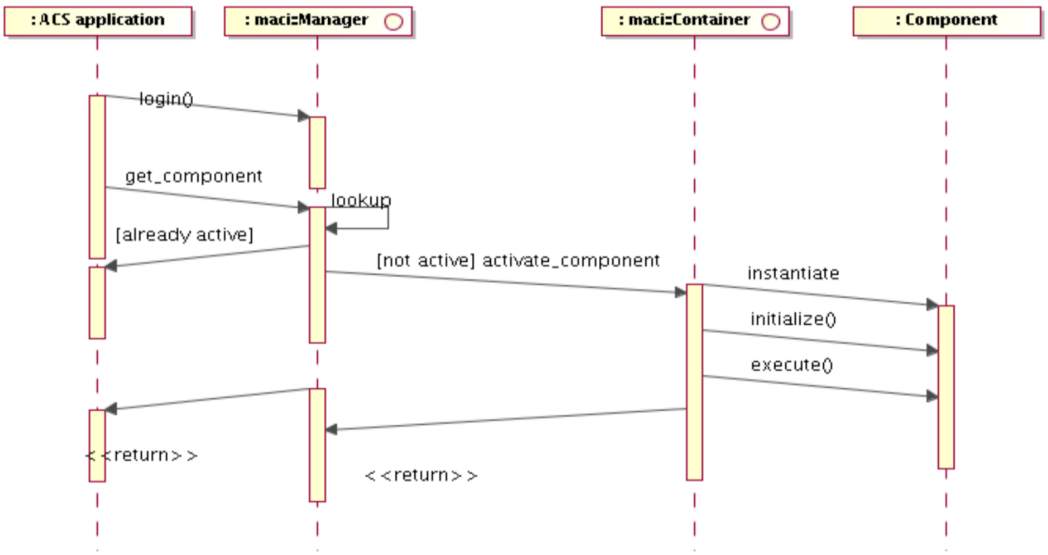


Figura 10: Secuencia de activación de un componente.
Fuente: ALMA Common Software Architecture [Chiozzi et al., 2009].

CAPÍTULO 3
PROPUESTA DE SOLUCIÓN

En el presente capítulo, se explicará la solución propuesta para enfrentar el problema presentado en capítulos anteriores, incluyendo diseños de interfaces de usuario, arquitectura de software y un prototipo funcional elaborado para validar la propuesta.

Haciendo uso de algunos conceptos introducidos en el capitulo 2, se detallarán los distintos elementos que conforman la arquitectura de la solución, incluyendo las áreas de desarrollo de frontend y backend.

3.1. Solución

Tal como se mencionó en capítulos anteriores, uno de los problemas de las interfaces que implementa el framework ACS es que no pueden ser accedidas desde lugares en donde el framework no esté disponible. Es necesario tener una máquina con el software instalado o tener acceso remoto mediante escritorio remoto o acceso ssh a alguna máquina que si lo posea. Además de lo poco conveniente en la forma de acceder, el hecho de tener que instalar el sistema provoca un problema de compatibilidad con los distintos sistemas operativos. Por otro lado, tener aplicaciones diferentes por cada una de las tareas que se pueden realizar en ACS no es eficiente, que por lo demás, es necesario ejecutar comandos tales como acscommandcenter, objexp o cdbBrowser para iniciarlas.

La solución que se propone en esta memoria es el desarrollo de un proyecto web que incluya la elaboración de una aplicación web y múltiples aplicaciones Backend que permitan alimentar con información y eventos a la aplicación web. La idea principal, es condensar todas las funcionalidades de las actuales interfaces hechas en Swing en una sola y gran aplicación web, aprovechando el modelo de componentes que ofrecen algunas de las actuales bibliotecas y web-frameworks. Por otro lado, también se propone rediseñar la forma en la que se presenta la información a los usuarios, aprovechando las capacidades cognitivas del ser humano con el fin de disminuir el tiempo que requiere una persona para realizar determinada acción sobre el sistema.

Esta solución significará una disminución en los tiempos de respuesta y en la efectividad a la hora de realizar tareas visuales sobre ACS, tales como el acceso a la información de componentes, ejecución de funciones y visualización de eventos de forma instantánea. Además, al ser una sola aplicación centralizada y en línea, es mucho más simple mantener las funcionalidades actualizadas y en un correcto funcionamiento.

Todo esto va de la mano con la actual tendencia en la industria del desarrollo de aplicaciones web [Brennan, 2015], que con las tecnologías de hoy en día es posible construir aplicaciones complejas, de fácil acceso, simples de mantener y con bajos tiempos de deployment. Además, los usuarios no tienen la necesidad de instalar paquetes adicionales. Solo basta con acceder al dominio web para obtener acceso a todas las funcionalidades.

Adicionalmente, esta solución presenta grandes opciones de escalabilidad, permitiendo que a futuro, las interfaces de ACS puedan migrar a otras plataformas sin la necesidad de construir todo desde cero. Al separar frontend y backend, se abre la puerta a nuevas plataformas, reutilizando las implementaciones de backend y solo creando las nuevas aplicaciones de frontend para las nuevas plataformas.

A raíz de lo anterior, la solución se divide en dos ramas: frontend y backend. A continuación, se explicará la solución en base a cada una de estas ramas.

3.2. Arquitectura

La arquitectura de la aplicación fue elaborada teniendo en cuenta un sistema modularizado de tres partes principales, en donde cada una de ellas son independientes entre sí y lo único que es relevante es el protocolo y formato de comunicación utilizado entre los servicios. Además, se espera que gracias a este tipo de arquitectura, la mantención de los servicios sea relativamente fácil en comparación a una arquitectura en donde Intermediario y Proveedor están incluidos en un solo lugar.

- **Aplicación web frontend:** Aplicación web desarrollada con alguna biblioteca o framework basado en componentes que se ejecuta en el navegador y se encarga de extraer la información directamente desde el Intermediario mediante solicitudes HTTP y una

conexión persistente mediante WebSockets. Para la conexión mediante WebSockets se utiliza la biblioteca Socket.io por las ventajas que esta presenta en el desarrollo de aplicaciones web con conexiones persistentes (Ver Sección 2.6.1). Cuando la aplicación se ejecuta, se realiza una solicitud al intermediario para obtener el estado actual del sistema de ACS, a su vez, se establece una conexión persistente con el intermediario con el fin de recibir eventos.

- **Intermediario backend:** Es una aplicación que se ejecuta en un servidor. Implementa endpoints que sirven como punto de entrada para las solicitudes hechas desde el Frontend. Esta aplicación utiliza principalmente dos protocolos de comunicación: HTTP y WebSockets (cuando está disponible). Para tareas sincrónicas o muy específicas tales como la ejecución de funciones en componentes ACS y la solicitud del estado inicial de ACS se utiliza una API REST. Para el sistema de eventos se utiliza una API de WebSockets (Socket.io). Cuando esta aplicación recibe la primera solicitud desde Frontend, se realiza una solicitud HTTP a la aplicación Proveedor para luego redireccionar esta respuesta a Frontend. Por otro lado, cada vez que se recibe un mensaje desde la aplicación Proveedor, se emite un mensaje con el evento en modalidad Broadcast a todos los clientes web conectados al intermediario.
- **Proveedor backend:** Esta aplicación se compone de tres servicios principales. El primero es un servicio REST que se encarga de recibir las solicitudes realizadas por el Intermediario, una vez recibidas, las procesa y realiza las consultas correspondientes a ACS para finalmente responder a la solicitud inicial. El segundo servicio se encarga de mantener una conexión persistente con el Websocket del intermediario y a la vez consumir una cola sincronizada de eventos que deben ser transmitidos al intermediario. El tercer servicio hace uso del módulo LoggingConsumer de ACS para escuchar todos los logs/eventos que se generan en el sistema. Cada evento recibido es procesado y agregado a la cola sincronizada.

Por lo tanto, la arquitectura de la propuesta de solución se compone de un total de cinco servicios separados en tres grandes secciones. En la ilustración 11 se muestra la arquitectura propuesta.

3.2.1. Frontend

En la actualidad existen múltiples bibliotecas y frameworks disponibles para la construcción de aplicaciones web reactivas y basadas en el modelo de componentes (Ver sección 2.5). Estas permiten alcanzar una alta reusabilidad y escalabilidad, por lo que esta solución se apoya exclusivamente en el uso de estas tecnologías.

Las unidades centrales del framework ACS son los Contenedores y Componentes, por lo que estos también serán las unidades centrales de la aplicación web. Para aprovechar las capacidades cognitivas de las personas y utilizar el procesamiento preatentivo [Treisman, 1985], en

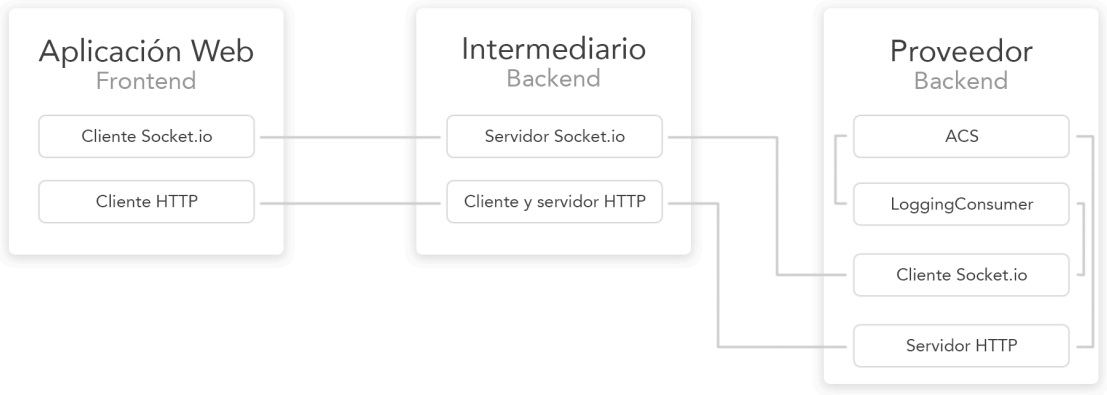


Figura 11: Arquitectura de la solución propuesta.
Fuente: Elaboración propia.

vez de listas o estructuras de árbol, se deben utilizar analogías que permitan relacionar estas unidades básicas con formas, colores o rellenos. En la Figura 12, se presenta el primer prototipo de la analogía de Contenedor y Componentes. A simple vista resulta sencillo identificar que los círculos de color verde son los Componentes asociados al Contenedor. Las líneas punteadas muestran las conexiones que poseen los componentes entre sí y los círculos de la porción derecha de la imagen representan botones que permiten ejecutar ciertas acciones sobre un componente, tales como, ver su configuración, ver sus métodos o desactivarlo.

Esta ilustración es una visión idealizada de un Contenedor, el cual solo administra cuatro Componentes. En la realidad, los Contenedores se encargan de muchos más Componentes, por lo que esta ilustración no es realista. Para solucionar este problema se propone utilizar un botón en la parte inferior del contenedor que permita expandir una cuadrícula de Componentes, tal como se muestra en la Figura 13, con el fin de hacer un mejor uso del espacio.

Adicionalmente, en un caso realista, no todos los Contenedores están en la misma máquina, por lo tanto, este es un punto relevante que debe ser considerado en la solución. En la ilustración 14 se muestra el resultado final de la visualización central de la aplicación web, en donde cada máquina corre los Contenedores correspondientes.

Las figuras mostradas, solo representan el elemento central de esta aplicación web. Es la analogía que se propone para la visualización de Máquinas, Contenedores y Componentes. Adicional a lo anterior, es necesario definir el layout de la aplicación, el cual es la estructura que encerrará todas las funcionalidades e interacciones. Al ser una propuesta, no se presentará un diseño completo de la aplicación, solo una sugerencia con la intención de que futuros trabajos sigan o modifiquen esta visión. En la ilustración 15, se presenta el prototipo final de lo que sería la aplicación web, con una zona exclusiva para la visualización de los elementos de ACS y un panel lateral de contenido dinámico que se adapta al contexto.

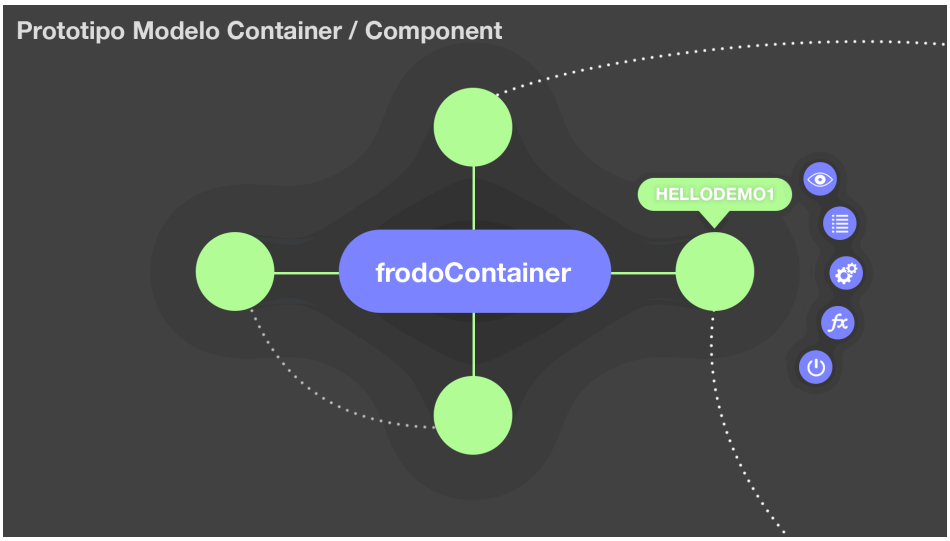


Figura 12: Primera ilustración prototipo de un Contenedor y sus Componentes.
Fuente: Elaboración propia.

En el aspecto técnico de esta sección, se propone el uso de alguna de las bibliotecas más populares para la construcción de interfaces de usuario en ambientes web. En base a lo expuesto en la sección 2.5, Vue.js y ReactJS son los candidatos seleccionados para esta propuesta. En cuanto al manejo del flujo de datos de la aplicación web, se propone alguna biblioteca basada en el modelo Flux (Ver sección 2.6.3), debido a la gran cantidad de elementos que debe manejar la aplicación, utilizar un modelo de flujo de datos básico podría generar un problema en el rendimiento y escalabilidad de la aplicación.

El resto de las bibliotecas que deban utilizarse para la construcción de las interfaces web, se dejan a criterio de los encargados de evaluar y llevar a cabo esta propuesta.

3.2.2. Backend

La solución propone la creación de una API REST con el fin de exponer ciertas funcionalidades de ACS y así ser accedidas desde la aplicación web. Esta propuesta no considera ninguna medida de autenticación o seguridad ya que no es relevante para la etapa actual, además, se considera un área que no es necesaria abarcar, debido a que los observatorios astronómicos manejan redes internas aisladas del exterior, por lo que teóricamente no es posible que personas mal intencionadas ingresen a la red y la información.

De manera simple, una API REST es una forma de exponer ciertos recursos de un sistema mediante una comunicación basada en solicitudes y respuestas. También existe la posibilidad de modificar estos recursos. En el caso de la solución propuesta, este tipo de API es útil para solicitar el estado inicial del sistema, ejecutar funciones u obtener más detalles de un elemento en especial. Lamentablemente, no es posible construir una plataforma catalogada

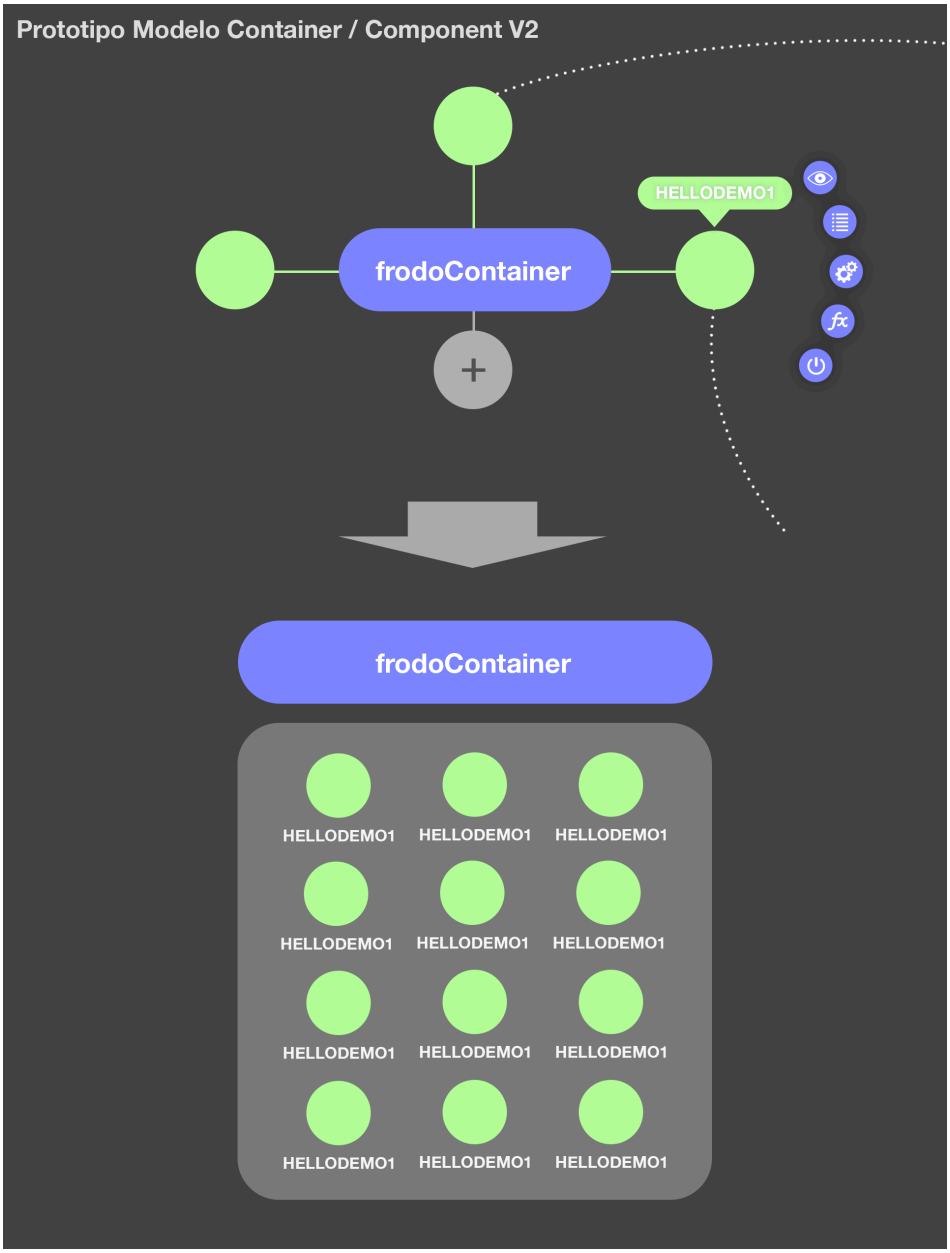


Figura 13: Segunda ilustración prototipo de un Contenedor y sus Componentes.
Fuente: Elaboración propia.

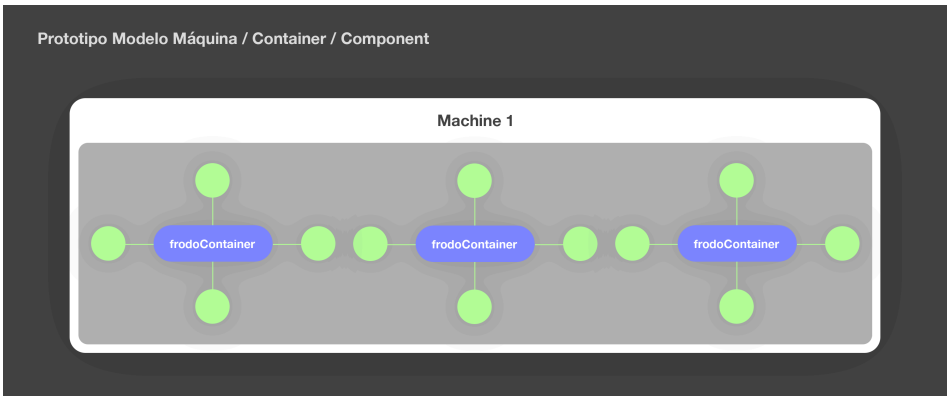


Figura 14: Ilustración prototipo de Máquinas, Contenedores y Componentes.
Fuente: Elaboración propia.

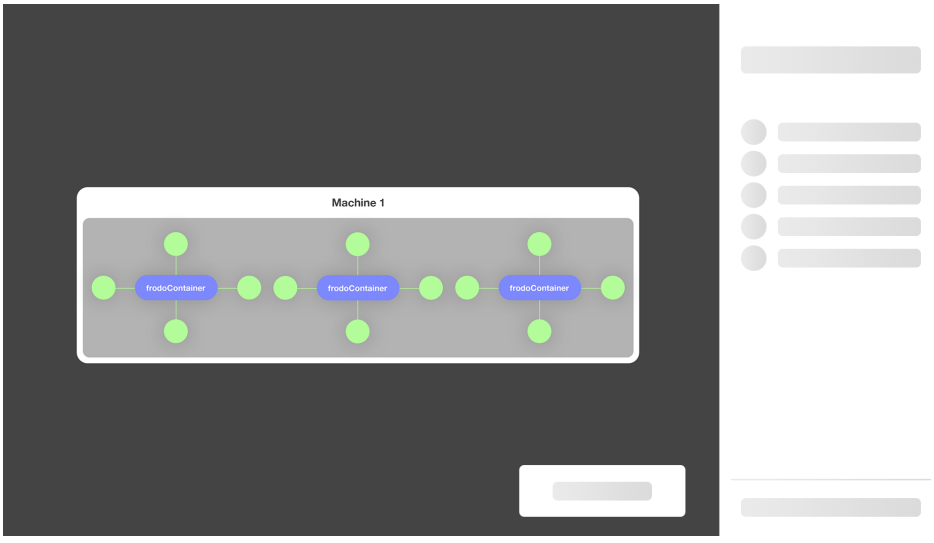


Figura 15: Ilustración prototipo de interfaz web.
Fuente: Elaboración propia.

como reactiva con solo una API REST. Existen técnicas llamadas Polling y Long Polling que permiten simular un sistema en base a eventos, pero es el cliente quien debe iniciar la conexión y por cada solicitud se abre una nueva conexión TCP, por lo que no es realmente una comunicación reactiva.

ACS de por sí, es un sistema reactivo. Funciona en base a eventos y todos sus elementos se ven afectados por estos eventos. Para llevar este tipo de comportamiento a la aplicación web y que pueda recibir de manera instantánea eventos emitidos desde ACS, es necesario mantener una conexión abierta y disponible en cualquier momento. Para esto existe un protocolo llamado Websocket que con una sola conexión TCP es posible transmitir mensajes de manera bidireccional y en cualquier momento. De esta forma es posible construir un verdadero sistema que reaccione a eventos de manera instantánea.

Por lo tanto, el backend de la solución propuesta debe exponer dos tipos de servicios, una API REST para acceder a los estados del sistema, y un Websocket para la transmisión de eventos de forma instantánea.

Respecto a la tecnología a utilizar en Backend, a pesar de que el lenguaje C++ posee una serie de herramientas para el desarrollo de plataformas web, estas no parecen tener mucho apoyo de la comunidad, o al menos no tanto como las alternativas de Java y Python, y si a esto se le suma el hecho de que C++ es un lenguaje de bajo nivel en comparación a Java y Python, se estima que no es conveniente usarlo ya que el esfuerzo adicional en su implementación versus el valor agregado, no es lo suficientemente grande. Adicionalmente, Crear una aplicación de alto rendimiento no es prioridad. En conclusión, se recomienda descartar C++ de las opciones.

3.3. Prototipo

El prototipo desarrollado para esta propuesta de solución intenta demostrar empíricamente la factibilidad técnica de una plataforma web reactiva utilizando el estado actual de ACS.

Actualmente, ACS no está diseñado para ser usado fuera su ecosistema, por lo tanto, tampoco está pensado para ser usado como servicio web, por lo que la integración con diferentes tecnologías puede ser un gran problema e incluso ser un impedimento para la elaboración de la solución propuesta. Es por esto que es necesario demostrar que lo que se propone es posible, incluso sin hacerle modificaciones al código fuente de ACS.

Haciendo uso de todos los recursos que ofrece ACS, fue posible desarrollar un prototipo funcional que implementa una serie de funcionalidades que permitirían mejorar la experiencia de usuario y los tiempos requeridos en la ejecución de tareas especificas. Existen ciertas funcionalidades, que serán explicadas en secciones posteriores, que no se llevaron a cabo, ya sea porque no son esenciales, el tiempo de desarrollo es demasiado elevado para un prototipo o simplemente aún no se encuentra una forma adecuada de hacerlas realidad.

Una forma adecuada de llevar a cabo una funcionalidad implica no realizar modificaciones a las implementaciones de ACS. Si bien, todas las funcionalidades desarrolladas se hicieron de una forma adecuada, las herramientas de ACS no ofrecen soluciones simples para casos de uso básicos como la obtención de la información completa de un Contenedor, por lo que fue necesario indagar en los códigos fuente de las aplicaciones Java ya existentes para entender como funcionaban y de que forma se podían adaptar estas soluciones al prototipo desarrollado.

3.3.1. Preparación Previa

Al comienzo, uno de los mayores desafíos fue establecer un ambiente de desarrollo que permitiera entender el funcionamiento de ACS y realizar pruebas. No existen instrucciones claras de como realizar una instalación limpia del sistema, y por lo tanto, la única opción es buscar en todos los rincones disponibles para ir uniendo piezas para al final tener un sistema que apenas es funcional. Cabe destacar que ACS no está pensado para funcionar en CentOS, pero gracias al trabajo del equipo de CSRG de la UTFSM, se cuentan con versiones capaces de correr en una máquina con CentOS.

Luego de varios intentos fallidos en la instalación de ACS, se logró cumplir con la tarea juntando información de diversos sitios, tanto de la documentación de ALMA como del trabajo realizado por el CSRG. Sin este objetivo cumplido, no hubiera sido posible realizar un estudio aplicado sobre la actualización de las interfaces gráficas de ACS.

3.3.2. Funcionalidades

A continuación, se definirán las funcionalidades implementadas en el prototipo de esta memoria.

- **Vista espacial de elementos:** Corresponde a la ventana principal de la aplicación web. En ella se pueden visualizar los tres tipos de elementos que componen ACS: Máquinas, Contenedores y Componentes. La posición de las Máquinas y Componentes está basada en un Grid System que permite posicionar nuevos elementos de manera automática y con animaciones que dan un sentido a lo que está ocurriendo en la visualización. Mientras que el Grid de Máquinas permite una cantidad de columnas indefinida y una cantidad de filas igual a dos, el Grid de Contenedores permite una cantidad de filas indefinida y una cantidad de columnas igual a cinco. Cabe destacar que todos los eventos que ocurren en esta visualización cuentan con sus respectivas animaciones que agregan un sentido de orientación y consistencia. Los Componentes, por su parte, se posicionan de manera estática en la órbita de los Contenedores, actualizando su posición de manera automática a medida que se agregan o eliminan Componentes. Esta visualización es interactiva, ya que permite enfocar y cambiar la posición de

la cámara solo con hacer clic en los elementos mostrados. Cuando un Contenedor es focalizado, sus Componentes se expanden para mostrar la cantidad de clientes que tiene cada uno. Al hacer clic sobe un Componente, es posible acceder a sus funciones, y en el caso de un Characteristic Component, es posible ver los valores actuales de sus propiedades.

- **Barra lateral multi-uso:** El concepto se refiere a una barra que permite visualizar contenido de forma dinámica. Actualmente solo implementa una funcionalidad, que es mostrar las funciones del Componente seleccionado, ejecutarlas y mostrar el resultado de la ejecución. Además, posee un selector booleano que permite cambiar el modo de la visualización entre automático y manual.
- **Modo automático:** Este modo fue pensado para situaciones en que se requiera usar la aplicación web como herramienta visual para el monitoreo de la actividad en ACS. Este modo funciona en base a eventos que son recibidos desde el Intermediario. El modo automático controla la cámara de la aplicación, enfocando los elementos que van siendo afectados por los eventos recibidos. Estos eventos son: Activación y desactivación de Máquinas, Activación y desactivación de Contenedores, Activación y desactivación de Componentes.
- **Ejecución de funciones de un Componente:** Cuando un Componente es seleccionado, la barra lateral multi-uso actualizará su contenido para mostrar una lista de las funciones disponibles y sus respectivos botones de ejecución. Cuando una función es ejecutada, el resultado es mostrado en un cuadro blanco dentro de la barra lateral. Además de funciones, también es posible acceder a atributos del Componente.
- **Propiedades de un Characteristic Component:** A diferencia de un componente normal, un Characteristic Component posee propiedades que van siendo actualizadas en el tiempo. En el caso de un telescopio, una propiedad sería la orientación actual, la cual se va actualizando a medida que el telescopio se mueve. En la visualización, esta información es mostrada cuando un componente es enfocado, mostrando el valor actual de las propiedades, el cual va siendo actualizado cada cierta cantidad de segundos. Se utiliza la técnica conocida como Polling, para extraer la información desde ACS.

3.3.3. Tecnologías

Las tecnologías utilizadas para la elaboración del prototipo fueron elegidas en base a la simpleza de los lenguajes involucrados y a la rapidez de desarrollo que estas herramientas entregaban. Otro factor considerado fue el utilizar tecnologías que tuvieran más altas probabilidades de ser usadas en un ambiente de producción real. A pesar de lo anterior, existen excepciones que serán explicadas más adelante.

A continuación, se nombrarán y explicarán las tecnologías utilizadas para el desarrollo del prototipo:

- **ReactJS:** Utilizado para la construcción de la interfaz gráfica web. Su paradigma basado en componentes facilita el desarrollo de interfaces complejas y con muchos elementos en pantalla, haciendo uso de su DOM Virtual para mantener un buen rendimiento en casos de muchas actualizaciones por segundo. Se utilizó esta biblioteca ya que posee una amplia gama de componentes que facilitan el trabajo en ciertas áreas, además, gran parte de los elementos que se muestran en pantalla corresponden a componentes constituidos de elementos SVG, por lo que le paradigma de componentes es un perfecto aliado para casos de uso como el de este prototipo.

- **Flask:** Dos de los servicios de este prototipo fueron desarrollados utilizando la biblioteca Flask para Python. La razón principal fue la simplicidad en la construcción de APIs y la gran cantidad de bibliotecas que posee el lenguaje Python. Además, Python es conocido por ser un lenguaje ideal para la construcción de prototipos.

- **Spring:** Utilizado exclusivamente para el desarrollo de una API para reutilizar código Java extraído desde el código fuente del Command Center de ACS. La API se encarga de traducir Corbalocs o Interoperable Object Reference (IOR) a direcciones IP o hostnames para ser utilizados como nombres de las máquinas de ACS (Ver sección 2.3). Esta funcionalidad no es nativa de las bibliotecas de ACS, por lo que no existe en ACSpy.

- **Socket.io:** Existen múltiples técnicas que permiten una comunicación instantánea entre dos o más clientes. Algunas entregan mejores resultados que otras. Algunos ejemplos son: Polling, Long Polling y WebSockets. Mientras que las dos primeras son técnicas, WebSocket es un protocolo de comunicación bidireccional que permite el intercambio de mensajes de forma instantánea. No siempre se puede asegurar que tanto un cliente como servidor soportarán este protocolo, por lo que es necesario tener opciones adicionales en caso de que esto ocurra. Una solución a este problema es la biblioteca Socket.io que permite la comunicación mediante WebSockets entre clientes y servidor, y en caso de que este no esté disponible, se utilizarán otras opciones para simular está conexión instantánea. El prototipo desarrollado hace uso de esta biblioteca para el intercambio de mensajes correspondientes a los eventos que ocurren dentro de ACS.

- **D3.js:** Uno de los puntos diferenciadores del prototipo desarrollado, es la gran cantidad de animaciones que se utilizan para darle un sentido a las cosas que van ocurriendo en la visualización. Dan un sentido de orientación y coherencia. D3.js provee una amplia cantidad de herramientas, desde la generación de gráficos complejos hasta simples animaciones de elementos SVG. Es utilizado principalmente para la elaboración de visualizaciones con gran cantidad de datos. Afortunadamente, se ha decidido dividir las herramientas de D3.js en distintas bibliotecas para quienes deseen hacer uso de solo algunas porciones de esta herramienta. En el caso de este prototipo, D3.js solo es utilizado para la generación de animaciones en los elementos SVG de la visualización.

- **Redux:** Existen casos en los que es necesario mantener un estado centralizado de una aplicación web, debido a que es necesario compartir datos entre muchos niveles de

componentes. El prototipo desarrollado utiliza Redux para almacenar el estado de todos los elementos de la aplicación: Máquinas, Contenedores y Componentes. De esta forma es simple acceder a la posición o datos adicionales de ciertos componentes en cualquier lugar de la aplicación.

3.3.4. Desafíos

A lo largo del desarrollo del prototipo aparecieron una serie de desafíos que dificultaron la programación de las funcionalidades descritas anteriormente. Debido a estos desafíos, también existen otras funcionalidades que no se pudieron llevar a cabo.

A continuación, se definen algunas de las dificultades que aparecieron en el camino y de que forma fue posible superarlas:

- **Suscripción a eventos de manera instantánea:** No se logró encontrar una forma clara y correcta de suscribirse a un sistema de eventos con el lenguaje Python. A pesar de que la biblioteca ACSpy ofrece un módulo para suscribirse al sistema de notificaciones de CORBA, se optó por hacer uso del sistema de logs utilizando el módulo LoggingConsumer debido a su simplicidad y a que los mensajes poseen una estructura consistente. Posteriormente, se seleccionó un conjunto de mensajes que serían utilizados para el sistema de eventos del prototipo y se utilizaron expresiones regulares para identificar que tipo de evento es y a que máquinas, contenedores o componentes involucra. En la Tabla 4 se especifican los cuatro tipos de eventos o rutinas utilizadas, junto con las expresiones regulares utilizadas:

Tabla 4: Expresiones regulares utilizadas para identificar eventos.
Fuente: Elaboración Propia.

Rutina	Expresión Regular
containerLogin	`r"Container'(.*)' logged in\."`
containerLogout	`r"Container'(.*)' logged out\."`
internalNoSyncDeactivateComponent	`r"Component'(.*)'.*`deactivated\."
getComponent	`r"'(.*)'`requested component 'curl:///(.*)'\."

- **Identificación de Máquinas:** La biblioteca de Python, ACSpy, no implementa alguna característica que entregue la ubicación de un Contenedor, es decir, no entrega la dirección IP o hostname de la Máquina de un Contenedor. Para solucionar este problema, fue necesario indagar dentro del código del software Command Center, descubriendo que la funcionalidad fue implementada exclusivamente para ese componente y no está incluida en las bibliotecas que ACS entrega. Por lo tanto, fue necesario crear un nuevo servicio haciendo uso del framework Spring y del código utilizado en Command

Center para exponer una API que traduzca una dirección Corbaloc o Interoperable Object Reference (IOR) a un hostname. Para obtener la dirección de CORBA de un Contenedor fue necesario agregar un método al PySimpleClient de ACSpy, el cual traduce la referencia de un Contenedor a una dirección de CORBA. A continuación, se muestra el método agregado a la clase PySimpleClient:

```
def getContainerLocation(self,reference):
     location = omniORB.obr.object_to_string(reference)

     location = urlopen(self.machineResolverUrl + '?location=' +
        location).read()

     return location
```

Cabe destacar que el código utilizado no sigue las reglas del PEP 8 ya que se debe ser consistente con el estilo de programación de ACSpy.

3.3.5. Problemas Sin Resolver

Como se trata de un prototipo funcional, no es raro que existan problemas que se dejen sin resolver, siempre y cuando no afecten por completo a la funcionalidad del software. En base a lo anterior, el desarrollo de este prototipo dejó una serie de problemas que deberían ser mitigados a la hora de realizar una implementación real de la solución propuesta.

Es probable que para solucionar algunos de los problemas presentados, se deban realizar cambios en el código de ACS.

A continuación, se nombran los problemas que requieren solución:

- **Rendimiento de Aplicación Web:** Al comienzo del desarrollo, el rendimiento general de la aplicación web estaba dentro de los límites aceptables para un usuario común. La aplicación daba una sensación de fluidez y rapidez. A medida que se fueron incluyendo funcionalidades y animaciones, los cuadros por segundo de la aplicación fueron decayendo drásticamente. Se presume que el problema se debe principalmente por la gran cantidad de elementos en pantalla que deben animarse en cada renderizado de la aplicación. También, pero menos probable, se cree que el uso intensivo de Redux al actualizar y acceder el estado de la aplicación puede afectar al rendimiento general. Se deja como trabajo futuro el dilucidar que factores son los que afectan realmente al rendimiento de la aplicación y buscar una solución al problema.
- **Creación de nuevo cliente en cada consulta:** Cada vez que se consulta la información de algún elemento de ACS, ya sea un Contenedor o un Componente, se crea un nuevo cliente, el cual es informado por el sistema de eventos, generando datos inconsistentes. Se implementó una solución temporal en que el sistema de eventos no considera

a los servicios del prototipo como clientes de ACS. Una mejor forma de abarcar el problema podría haber sido crear un solo cliente por servicio, pero el problema sigue existiendo. Una solución real sería implementar una forma en las bibliotecas de ACS para consultar información de Contenedores o Componentes sin la necesidad de crear un nuevo cliente.

A pesar de los problemas, dificultades y lo dicho en esta conclusión, el prototipo logra demostrar que integrar el framework ACS con una interfaz web reactiva es completamente posible gracias a bibliotecas como ACSpy.

CAPÍTULO 4
VALIDACIÓN DE LA SOLUCIÓN

En este capítulo, se exponen los métodos utilizados para comprobar que la solución propuesta por esta memoria, mediante la implementación del prototipo (Ver sección 3.3), significa una mejora real a la situación actual del ALMA Common Software. Junto con los métodos, se dan a conocer los resultados obtenidos, realizando una comparación entre un antes y un después de la implementación.

4.1. Metodología

Para validar la solución se propone realizar mediciones sobre el comportamiento del mouse a la hora de realizar tareas predefinidas sobre ACS y el prototipo funcional. De esta forma, se podrá comprobar si las mejoras implementadas en el prototipo disminuyen el esfuerzo que deben realizar los usuarios y si los tiempos requeridos disminuyen.

Para realizar las pruebas se utilizó un sistema con Windows 10, ejecutando una máquina virtual con CentOS 7 y ACS instalado. Todos los servicios requeridos para el prototipo funcional fueron ejecutados en la máquina virtual, incluyendo la aplicación web. En cada prueba, ACS está ejecutándose de antemano. Para las pruebas que necesitaron el uso del navegador, se utilizó una versión estable de Google Chrome.

Al realizar las pruebas se utilizó más de una aplicación nativa de ACS, debido a que las tareas propuestas no podían ser realizadas por solo una de estas aplicaciones. Estas aplicaciones son: Command Center y Object Explorer.

Las pruebas propuestas para la validación de la solución y prototipo son las siguientes:

1. Localizar al Componente Característico MOUNT, el cual pertenece al Contenedor py-Container.
2. Consultar los valores de las propiedades del Componente Característico MOUNT.
3. Consultar el atributo ComponentState del Componente Característico MOUNT.

Las comparaciones se realizarán en base a parámetros que describan el comportamiento del Mouse del usuario. En un comienzo, se estableció que la herramienta a utilizar sería Mousetron[9] para Windows 10. Lamentablemente, la herramienta es bastante limitada e intervenía

[9] `http://www.blacksunsoftware.com/mousotron.html`

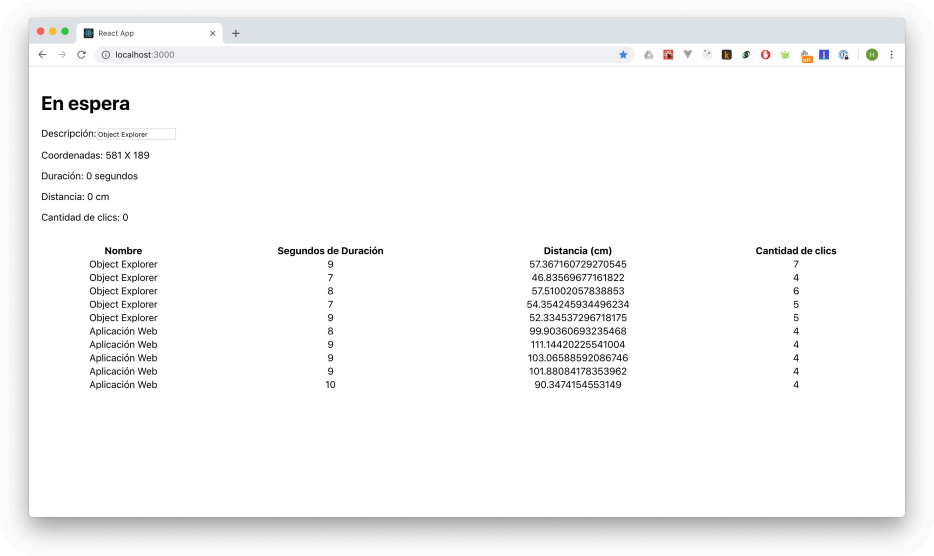


Figura 16: Aplicación web para capturar comportamiento de un Mouse remoto.
Fuente: Elaboración propia.

de manera negativa en la obtención de resultados, ya que el mismo usuario que resuelve la tarea debe ejecutar y detener el software de rastreo.

Debido a que no se encontró software adecuado para la realización de las pruebas, se optó por desarrollar una herramienta propia basada en Websockets y tecnologías web reactivas, la cual permite obtener información instantánea del Mouse de un equipo remoto a través de una aplicación web construida con ReactJS. La herramienta permite capturar los datos del comportamiento del Mouse con solo presionar la tecla espacio, por lo que no interviene de ninguna forma con las tareas que debe realizar el usuario de prueba. Los datos capturados por la herramienta son: Duración en segundos, distancia recorrida en centímetros, cantidad de clics y las coordenadas del cursor (Ver figura16). Para obtener la distancia recorrida en centímetros se consideró la resolución del monitor del equipo remoto y el tamaño en pulgadas.

Las pruebas fueron realizadas por una sola persona y fueron repetidas 5 veces con el fin de obtener una muestra más amplia para la comparación.

4.2. Desarrollo y resultado de las pruebas

Las pruebas descritas en las siguientes secciones representan tareas básicas que pueden ser realizadas con ACS. Se consideró que las tareas fueran simples de realizar con las interfaces ya existentes de ACS, con el fin de establecer límites más exigentes para el prototipo desarrollado.

4.2.1. Localizar al Componente Característico MOUNT

El propósito de esta tarea es comparar los tiempos y esfuerzos para acceder a ciertos Componentes si es que se quisiera consultar la información de estos. Por esta razón, la aplicación Object Explorer no era adecuada para esta tarea.

Las condiciones de esta prueba para cada aplicación fueron de la siguiente forma:

- **Aplicación Web:** La aplicación web se encuentra precargada en el navegador. El usuario solo debe hacer clic en el botón de recarga. El usuario debe comenzar con el Mouse relativamente al centro de la pantalla.
- **Command Center:** La aplicación se encuentra cerrada al momento de iniciar. El comando para iniciar la aplicación se encuentra escrito de antemano en el terminal. El usuario solo debe ejecutar el comando. El usuario debe comenzar con el Mouse relativamente al centro de la pantalla.

Tabla 5: Resultados obtenidos de la prueba N°1.
Fuente: Elaboración Propia.

Aplicación	Duración (segundos)	Distancia (cm)	Cantidad de clics
Aplicación Web	8	100.55	2
	9	104.13	2
	8	91.46	2
	7	102.51	2
	8	96.83	2
Command Center	11	30.39	2
	10	36.96	2
	11	20.46	2
	11	42.62	2
	10	27.68	2

En la tabla 5 se presentan los resultados obtenidos para cada una de las aplicaciones. Se resaltaron los valores más elevados. En primer lugar, es posible apreciar que las distancias recorridas en la aplicación web son mayores, esto se debe a que la interfaz de Command Center tiene dimensiones mucho más pequeñas que la aplicación web.

La duración de las pruebas fueron mayores en la aplicación Command Center, esto se debió principalmente a los extensos tiempos de carga.

Finalmente, se extrae que la realización de esta prueba es más rápida en la aplicación web con tecnologías reactivas, y por lo tanto, presenta una mejora en los tiempos requeridos.

4.2.2. Consultar propiedades del Componente MOUNT

Todo Componente Característico posee propiedades que van siendo actualizadas cada cierto tiempo, dependiendo del grado de actividad del componente.

El propósito de esta prueba es poder acceder de manera rápida a sus propiedades con el fin de que el usuario esté en conocimiento sobre el comportamiento de un Componente Característico de forma instantánea.

A diferencia de la prueba anterior, en esta se utilizó la aplicación Object Explorer, debido a que este tipo de tareas aprovechan el verdadero propósito de la aplicación

Las condiciones para esta prueba fueron exactamente las mismas que las establecidas para la prueba N°1.

Tabla 6: Resultados obtenidos de la prueba N°2.
Fuente: Elaboración Propia.

Aplicación	Duración (segundos)	Distancia (cm)	Cantidad de clics
Aplicación Web	9	94.47	4
	8	89.06	3
	7	71.54	3
	7	80.76	4
	8	73.18	3
Object Explorer	23	103.43	16
	20	88.55	18
	21	109.42	16
	26	109.91	23
	21	105.10	19

En la tabla 6 se muestran los resultados obtenidos. Es fácilmente apreciable que los mayores resultados corresponden a pruebas realizadas en el Object Explorer.

Es interesante prestar atención a las distancias recorridas, en donde se aprecia que las correspondientes al Object Explorer son iguales o más elevadas que las de la aplicación web, aún cuando la interfaz del Object Explorer es más pequeña.

Una de las mayores desventajas del Object Explorer fue que para lograr obtener la información de cada una de las propiedades, era necesario realizar una elevada cantidad de clics. Además, en muchas ocasiones los clics no ejecutaban la acción deseada, aumentando aún más la cantidad de clics.

La ventaja de la aplicación web está en que las propiedades se muestran inmediatamente al acceder al detalle de un Componente Característico, los cuales van siendo actualizados cada una cierta cantidad de segundos.

4.2.3. Consultar el atributo ComponentState del Componente MOUNT

La característica principal del Object Explorer es la ejecución de funciones y la consulta de atributos de un Componente. El propósito de esta prueba es identificar si el prototipo desarrollado es capaz de superar en ciertos factores a la característica principal de una aplicación de ACS.

Las condiciones para esta prueba son las mismas que la utilizadas en pruebas anteriores y tal como se mencionó en el párrafo anterior, la aplicación utilizada fue el Object Explorer.

Tabla 7: Resultados obtenidos de la prueba N°3.
Fuente: Elaboración Propia.

Aplicación	Duración (segundos)	Distancia (cm)	Cantidad de clics
Aplicación Web	10	90.34	4
	9	101.88	4
	9	103.88	4
	9	111.14	4
	8	99.90	4
Object Explorer	9	52.33	5
	7	54.35	5
	8	57.51	6
	7	46.83	4
	9	57.36	7

En la tabla 7 se presentan los resultados obtenidos, en los que se puede apreciar que los tiempos destinados a la tarea están bastante equilibrados, siendo el único punto diferenciador la distancia recorrida por el Mouse. Como se mencionó anteriormente, esto se debe por el gran tamaño de la interfaz y no necesariamente por una falla en el diseño de la aplicación web.

La mayor cantidad de clics lo requiere el Object Explorer, esto es debido a que en ocasiones el clic no ejecutaba las acciones deseada, por lo que no es necesariamente a que el recorrido de la aplicación requiera una mayor cantidad de clics.

4.3. Conclusiones

En base a las pruebas realizadas y sus resultados, es posible elaborar una conclusión parcial, en la que se reconoce que el prototipo desarrollado cumple con las expectativas planteadas al comienzo del desarrollo, estando a lo menos, al mismo nivel que las aplicaciones de ACS en cuanto a usabilidad y efectividad.

Un punto a favor del prototipo es que sigue una línea clara de hacer que el usuario realice el

menor esfuerzo necesario para cumplir con alguna tarea. Esto se traduce en la baja cantidad de clics para realizar estas tareas.

CAPÍTULO 5
CONCLUSIONES

Durante el desarrollo de esta memoria, han sido cada vez más evidentes los problemas y dificultades que presenta el software ACS, demostrando lo intenta evitar el nuevo observatorio CTA.

ACS es software antiguo y desactualizado en ciertas áreas, por lo que su mantención y escalabilidad es costosa. Además de estar ligado a una filosofía distribuida sobre CORBA, su código fuente aún está incompleto y presenta varias falencias. Muchas de sus bibliotecas aún están incompletas, y muchas de sus fallas no están solucionadas. ACS fue presentado aproximadamente hace 18 años, periodo de tiempo en el que no ha tenido cambios significativos en su arquitectura. Esto tiene un efecto directo en la migración de sus componentes gráficos a nuevas tecnologías, dificultando la tarea de realizar dichas migraciones.

Se hace evidente que es necesario hacer una renovación de las tecnologías que ACS utiliza, desde su sistema distribuido y la administración de paquetes, hasta sus interfaces gráficas.

5.1. Estado actual de ACS

El eje central de esta memoria son las interfaces gráficas de las aplicaciones de ACS, las cuales requieren una gran actualización para aprovechar las tecnologías actuales y evitar la obsolescencia, lo que trae consigo una alta tasa de incompatibilidad, carencia de equipo capacitado para mantenimiento, bajo rendimiento y pocas capacidades de escalabilidad. Es debido a esto que los límites de la solución se ven impuestos por las posibles falencias en el código fuente y en la documentación disponible.

5.2. Desarrollo de un Prototipo

El trabajo realizado en esta memoria demuestra que ACS tiene un gran potencial en el uso de tecnologías web. Gracias a los grandes avances en el área, es posible construir plataformas sumamente complejas y ricas en contenido. Una sola plataforma web que reemplace a todas las aplicaciones nativas de ACS es uno de los objetivos que esta memoria propone para el trabajo futuro.

Gran parte de las funcionalidades propuestas para el prototipo funcional fueron posibles llevarse a cabo utilizando solo las herramientas que ACS provee. Sin embargo, lo anterior no quiere decir que las funcionalidades se hayan desarrollado de la manera más adecuada.

Durante el desarrollo del prototipo y la validación de la solución, se experimentó con múltiples tecnologías y bibliotecas. La gran cantidad de investigación requerida para cumplir con los objetivos propuestos ayudó a explorar las diversas herramientas disponibles, permitiendo descartar con mayores argumentos aquellas que no cumplían con las expectativas de la solución. Además, se requirió un gran trabajo e investigación debido a que la documentación de ACSpy no es lo bastante completa. En internet no existen muchos ejemplos de cómo acceder a la información de un Contenedor o Componente. Si bien, ACSpy entrega herramientas para obtener información de estos elementos, no es completamente efectivo. En comparación a herramientas como el Command Center, la información entregada por ACSpy es insuficiente, teniendo que utilizar otras técnicas para acceder a la información relativamente completa. Es por esto que fue necesario implementar características que no estaban resueltas por ACS, como también extraer líneas de código de otras aplicaciones y en diversos lenguajes. Un ejemplo concreto es que para identificar la máquina a la que pertenece un Contenedor, fue necesario crear un nuevo servicio hecho en Java reutilizando código de la aplicación Command Center. Esto demuestra que tanto las bibliotecas de Java y Python no están completas y no entregan una experiencia agradable para que nuevos desarrolladores se dediquen al desarrollo de nuevas herramientas basadas en nuevas tecnologías.

El estado actual del prototipo presenta serios problemas de rendimiento, provocando que la tasa de cuadros por segundo de la aplicación web fuera bastante baja, empeorando la experiencia de usuario. Además, la aplicación web utiliza una gran cantidad de recursos por lo que fue necesario utilizar sistemas más poderosos que un computador portátil para realizar pruebas. Es por esto que queda como trabajo futuro implementar optimizaciones a todos los sistemas de la aplicación web, desde el dibujado de los componentes gráficos hasta los sistemas de eventos.

El prototipo desarrollado cuenta con una gran cantidad de animaciones que permiten al usuario mantenerse orientado frente a todos los cambios gráficos que sufre la interfaz a medida que se producen eventos en el sistema. Dichas animaciones fueron logradas gracias a la biblioteca D3.js, la cual fue dividida en distintos módulos para evitar tener que descargar la biblioteca completa. Se presume que gran parte de los problemas de rendimiento se deben a estas animaciones, por lo tanto, se propone encontrar la forma de disminuir el uso excesivo de estas o incluso eliminarlas por completo, siempre y cuando no empeore la experiencia de usuario. Si por algún motivo no es posible disminuir la cantidad de animaciones, se propone crear una biblioteca optimizada para realizar animaciones en componentes SVG en ReactJS.

Para mantener el estado actual de los datos en la aplicación web, se utilizó una biblioteca llamada Redux, la cual almacena en un lugar centralizado toda la información del sistema, de tal forma que pueda ser accedida por cualquier componente mediante conectores. En la actualidad existen mejores formas de realizar este tipo de tareas. La forma recomendada es utilizando el Context API de ReactJS, el cual requiere mucha menos configuración que Redux, disminuyendo la complejidad del sistema. Sumado a lo anterior, la forma en que la información está siendo guardada no es óptima y se presume que está puede ser otra raíz de los

problemas de rendimiento. Por lo tanto, se propone mejorar el modelo de almacenamiento de la aplicación, con el fin de mejorar el rendimiento.

En cuanto a los servicios de Backend, debido a la poca documentación de ACS, se decidió que para el sistema de eventos se utilizaría el sistema de Logging, escuchando constantemente todos los registros que el sistema ACS generara y reconociendo mediante expresiones regulares los mensajes que serían o no utilizados. Esta forma de obtener eventos no es adecuada, sobre todo porque en ocasiones se generaba un retraso de unos cuantos segundos entre que el evento ocurría y el sistema de Logging generaba el mensaje. Por lo tanto, se propone investigar una forma rediseñar el sistema de eventos para mejorar la confiabilidad del sistema.

5.3. Resultados obtenidos

Encontrar una aplicación adecuada que permitiera medir de manera correcta y precisa las pruebas de validación fue una tarea no trivial. Muchas de las opciones disponibles no cumplían con los requerimientos y otros no ofrecían versiones gratuitas u Open Source. Finalmente, en base a todo el conocimiento adquirido y a todas las horas de trabajo dedicadas al desarrollo del prototipo, fue posible desarrollar una aplicación que permitiera obtener las mediciones necesarias de forma precisa y con resultados verídicos.

La aplicación creada para la validación de la solución fue capaz de registrar los clics, distancia recorrida por el mouse y el tiempo de cada prueba. Contaba con tres elementos principales: cliente para sistemas Linux, servidor con protocolo de WebSockets y aplicación web. Esta última permitía iniciar y detener las pruebas de forma remota, sin interrumpir las tareas que realizaba el usuario a prueba.

Los resultados de las pruebas permitieron determinar que tan efectivo es el prototipo. En muchas ocasiones, el prototipo superaba a la aplicación de ACS, mientras que en otras los resultados eran similares. Estos resultados demostraron que la forma en que el prototipo fue desarrollado, permitió disminuir los tiempos en la realización de tareas sobre el sistema. Por el momento, aún existen funcionalidades que deben ser mejoradas y adaptadas al nuevo paradigma impuesto por el prototipo, en donde los usuarios ven una representación gráfica del sistema en lugar de listas y botones planos. Esta representación gráfica, tal como se mencionó en los primeros capítulos, trata de aprovechar la capacidad preatentiva de las personas, algo que las pruebas fueron capaces de demostrar como logrado.

Por otro lado, la cantidad de datos utilizada en el prototipo solo representa un pequeño porcentaje de los datos que maneja un observatorio en etapa de producción. Es claro que el estado actual del prototipo no sería compatible con tal cantidad de datos, algo que en estas etapas no es verdaderamente relevante, ya que solo se intenta comprobar la aplicabilidad del paradigma más que la utilidad en sí.

En cuanto a los aspectos a mejorar en la etapa de validación, se encuentra la forma en que

el prototipo es ejecutado. Actualmente se compone de 4 módulos que deben ser ejecutados por separado y solo en un sistema con ACS instalado, lo que dificulta mostrar los avances logrados a los interesados. Es por esto que se propone encapsular todos estos módulos en una imagen Docker, con el fin de hacer más fácil la instalación en diversos sistemas operativos.

5.4. Conclusiones Finales

En base a los resultados obtenidos, es posible concluir que el prototipo desarrollado cumple con las expectativas de un prototipo funcional. Iguala o supera alguna de las funcionalidades de las aplicaciones Command Center y Object Explorer, siendo una excelente señal para un tipo de tecnología nunca antes usado en el ecosistema de ACS. Además, logra demostrar que integrar el framework ACS con una interfaz web reactiva es completamente posible gracias a bibliotecas como ACSpy. Lamentablemente, considerando el estado actual de ACS, no se ve viable ni recomendable realizar este tipo de mejora, lo que abre nuevas líneas de investigación sobre las tecnologías de ACS. En primer lugar, se recomienda realizar un estudio aplicado relacionado a la migración de los componentes clave de ACS a nuevas tecnologías que permitan la integración con otras tecnologías de manera fácil, segura y efectiva. De esta forma el abanico de posibilidades crecerá enormemente y la realización de una aplicación web para controlar todos los aspectos de ACS será posible. En la misma línea, ya existen estudios relacionados a la migración de tecnologías de ACS. En particular, un documento que propone una solución para la evaluación y selección de tecnologías de comunicación en sistemas distribuidos para ser aplicados en ACS. [Sanhueza, 2018]

Por otro lado, en cuanto al uso de nuevas tecnologías y a las dificultades encontradas en este estudio aplicado, se propone como trabajo futuro realizar un estudio para actualizar las tecnologías bases de ACS, con el fin de permitir a los desarrolladores crear aplicaciones que no dependan de las tecnologías que ACS ocupe, lo que puede ser logrado exponiendo uno o más servicios mediante APIs. Idealmente, el desarrollo de microservicios permitirá incrementar la escalabilidad del sistema, permitiendo que estos servicios no sean 100 % dependientes entre sí. Además, el utilizar bibliotecas modernas permitirá contar con más personal capacitado para la mantención de los sistemas.

Como es posible apreciar, el prototipo es mejorable desde muchos aspectos. Esto no quiere decir que el desarrollo fue mal ejecutado, si no que fue el adecuado para un ambiente de pruebas y validación. El estado actual de la aplicación no cumple con los estándares de calidad necesarios para ser lanzada a una fase de producción. Aún existen muchos problemas que solucionar y tal como se mencionó en párrafos anteriores, se requiere mucho trabajo aún para que sea 100 % útil para los usuarios del observatorio. Se debe considerar que el hardware utilizado por los posibles usuarios no será el de mayor potencia, y por lo tanto se ve necesario realizar mejoras para que la aplicación pueda ser ejecutada en los sistemas más básicos. En cuanto al estado de los servicios de Backend, se puede concluir que el sistema como tal no requiere mayores cambios, ya que simplemente es una vía de intercambio de mensajes entre ACS y la aplicación web. Lo que si necesita mejoras en el área de Backend es

la conexión con las bibliotecas de ACS, para obtener información confiable y sin errores.

A modo de cierre, teniendo en cuenta la experiencia adquirida, lo desarrollado y los desafíos superados, se concluye que se realizaron grandes avances en materia de actualización de sistemas para los observatorios que decidan utilizar ACS como software principal, particularmente en el área de componentes gráficos. El trabajo realizado es solo uno de los pequeños pasos necesarios para comenzar con la reingeniería para mejorar ACS. Los pasos posteriores consistirán en seguir realizando estudios de factibilidad en las diversas áreas con el fin de tener una gran propuesta de valor para quien decida darle a ACS una gran actualización.

REFERENCIAS BIBLIOGRÁFICAS

[Spr, 2017] (2017). Spring framework documentation. https://docs.spring.io/spring/docs/current/spring-framework-reference/index.html. (Consultado el 02/12/2017).

[Adhikari, 2016] Adhikari, A. (2016). Full stack javascript: Web application development with mean.

[Aswani y de Larquier, 2019] Aswani, J. y de Larquier, S. (2019). How data inspires building a scalable, resilient and secure cloud infrastructure at netflix. https://medium.com/netflix-techblog/how-data-inspires-building-a-scalable-resilient-and-secure-cloud-infrastructure-at- (Consultado el 05/05/2019).

[Banks y Porcello, 2017] Banks, A. y Porcello, E. (2017). Learning React: functional web development with React and Redux. .º'Reilly Media, Inc.".

[Bao y Chen, 2014] Bao, F. y Chen, J. (2014). Visual framework for big data in d3. js. En 2014 IEEE Workshop on Electronics, Computer and Applications, pp. 47–50. IEEE.

[Beilis, 2008] Beilis, A. (2008). Cppcms — high performance c++ web framework. http://cppcms.com/wikipp/en/page/main. (Consultado el 01/12/2017).

[Brennan, 2015] Brennan, K. (2015). The rise of web technology. http://web.archive.org/web/20180606014955/https://www.sencha.com/blog/the-rise-of-web-technology/. (Consultado el 05/05/2019).

[Caproni, 2007] Caproni, A. (2007). Acs alarm system.

[Chiozzi et al., 2006] Chiozzi, G and Caproni, A and Jeram, B and Sommer, H and Wang, V and Plesko, M and Sekoranja, M and Zagar, K and Fugate, DW and Harrington, S and others (2006). Application development using the alma common software. En Advanced Software and Control for Astronomy, volumen 6274, p. 627406. International Society for Optics and Photonics.

[Chiozzi et al., 2005] Chiozzi, Gianluca and Sekoranja, M and Caproni, A and Jeram, B and Sommer, H and Schwarz, J and Cirami, R and Yatagai, H and Avarias, JA and Hoffstadt, AA and others (2005). The alma common software, acs status and developments. Geneva, Switzerland: ICALEPCS.

[Chiozzi et al., 2009] Chiozzi, G., Sommer, H., y Shwarz, J. (2009). Alma common software architecture. Atacama Large Milimiter Array.

[Chiozzi y Šekoranja, 2004] Chiozzi, G. y Šekoranja, M. (2004). Alma common software overview.

[Curry y Grace, 2008] Curry, E. y Grace, P. (2008). Flexible self-management using the model-view-controller pattern. IEEE software, 25(3):84–90.

[Dumon y Deforche, 2008] Dumon, W. y Deforche, K. (2008). Wt: A web toolkit-wt is a freely available library and application server that lets c++ programmers write modern web applications using a familiar c++ gui programming style. Dr Dobb's Journal-Software Tools for the Professional Programmer, pp. 55–59.

[Falco, 2016] Falco, V. (2016). Introducing beast: Http and websockets c++ library.

[Fette y Melnikov, 2011] Fette, I. y Melnikov, A. (2011). The websocket protocol. Technical report.

[Fugate, 2003] Fugate, D. (2003). Alma common software and python.

[Gackenheimer, 2015] Gackenheimer, C. (2015). Introducing flux: An application architecture for react. En Introduction to React, pp. 87–106. Springer.

[Gosling, 1995] Gosling, J. (1995). Java™: An overview. Sun Microsystems.

[Grinberg, 2014] Grinberg, M. (2014). Flask web development: developing web applications with python. .º'Reilly Media, Inc.".

[Ho Ngoc, 2014] Ho Ngoc, H. (2014). Single page web application with restful api and angularjs: Best practices with verto monitor.

[Hofmann, 2017] Hofmann, W. (2017). The cherenkov telescope array: Exploring the very-high-energy sky from eso's paranal site. The Messenger, 168:21–26.

[Holovaty et al., 2007] Holovaty, Adrian and Kaplan-Moss, Jacob and others (2007). The django book. http://www. djangobook. com/en/1.0.

[Islam Naim, 2017] Islam Naim, N. (2017). Reactjs: An open source javascript library for front-end development.

[Jeram et al., 2007] Jeram, b., Chiozzi, G., Plesko, M., Fugate, D., y Roberts, S. (2007). Acs error system.

[Kyriakidis et al., 2016] Kyriakidis, A., Maniatis, K., y You, E. (2016). The majesty of vue.js.

[Liu y Sun, 2012] Liu, Q. y Sun, X. (2012). Research of web real-time communication based on web socket. International Journal of Communications, Network and System Sciences, 5(12):797.

[McDonough, 2011] McDonough, C. (2011). The pyramid web application development framework. Agendaless Consulting.

[McHale, 2007] McHale, C. (2007). Corba explained simply. URL: http://www. CiaranMcHale. com/corba-explained-simply [cit. 2010-3-1].

[Mikowski y Powell, 2013] Mikowski, M. y Powell, J. (2013). Single page web applications: JavaScript end-to-end. Manning Publications Co.

[Panirahi, 2016] Panirahi, K. (2016). Google web toolkit. Tutorialpoint.

[Pisano et al., 2009] Pisano, J., Fugate, D., y Lucero, S. (2009). Acs notification channel module software design & tutorial.

[Rauch, 2013] Rauch, G. (2013). Socket. io-the cross-browser websocket for realtime apps.

[Roberts, 2004] Roberts, S. (2004). Starting out with acs: A 'new alma developers' guide.

[Sanhueza, 2018] Sanhueza, R. (2018). Evaluation of distributed-system technologies for alma common software.

[Stroustrup, 1999] Stroustrup, B. (1999). An overview of the c++ programming language. Handbook of object technology.

[Thorson, 2015] Thorson, P. (2015). Websocket++ 0.6.0 has been released. `https://www.zaphoyd.com/articles/2015-06/websocket-060-has-been-released`. (Consultado el 01/12/2017).

[Treisman, 1985] Treisman, A. (1985). Preattentive processing in vision. Computer vision, graphics, and image processing, 31(2):156–177.

[2007] Van Rossum, Guido and others (2007). Python programming language. En USENIX annual technical conference, volumen 41, p. 36.

[Vojislav et al., 2011] Vojislav, S., Vlajić, S., Milic, M., y Ognjanovic, M. (2011). Guidelines for framework development process. pp. 1–9.

[Žagar y Georgieva, 2007] Žagar, K. y Georgieva, R. (2007). Logging and archiving.