Facultad 2

Componente de visualización de las notificaciones del sistema XABAL eXcriba, para la plataforma Android de Alfresco Mobile

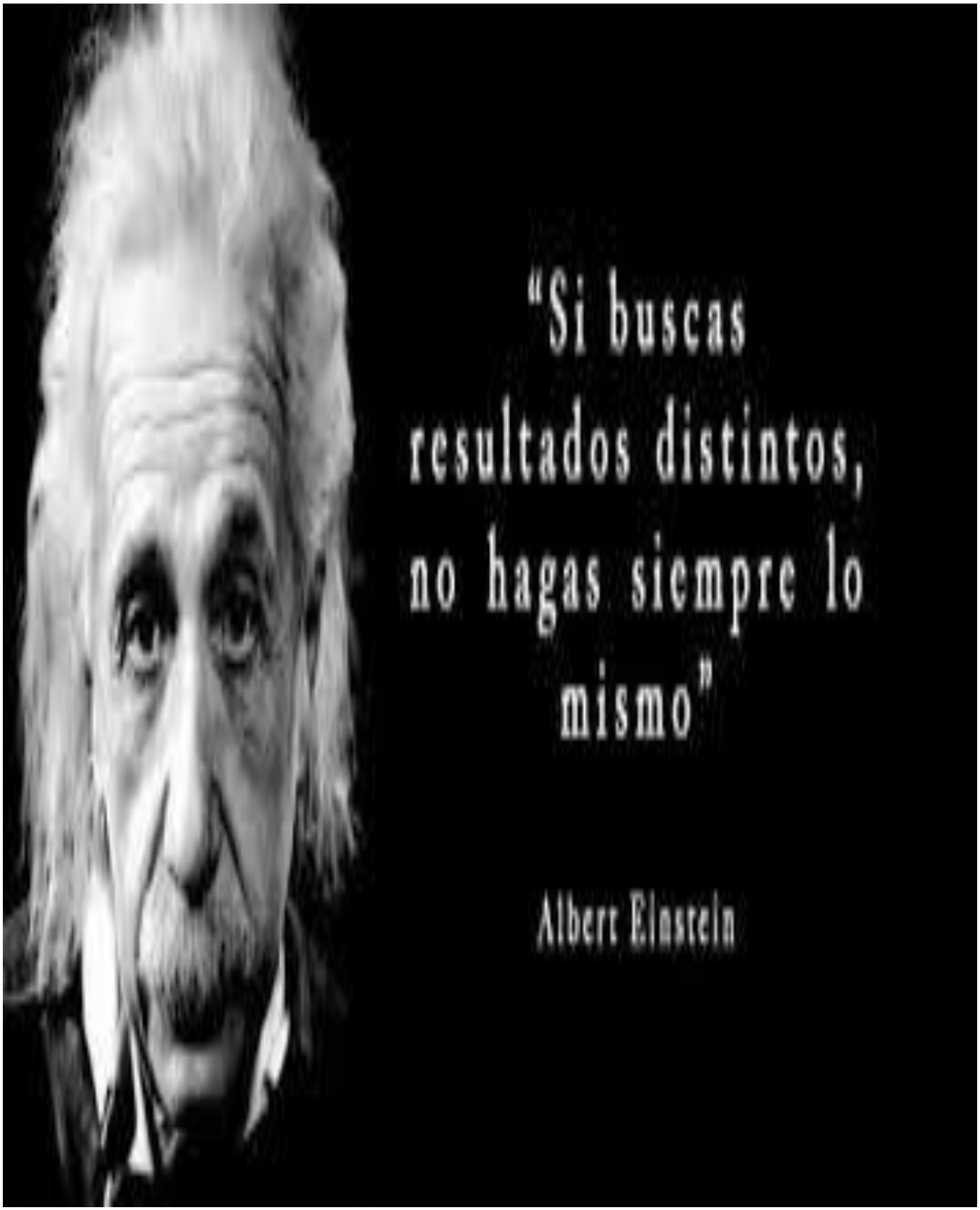
Trabajo de Diploma para optar por el título de
Ingeniero en Ciencias Informáticas

Autor: Miguel Angel Hernández González

Tutores: MSc. Darling Darias Pérez

Ing. Frank Simón Ricardo

La Habana, Cuba, julio 2018


"Si buscas
resultados distintos,
no hagas siempre lo
mismo"
Albert Einstein

DECLARACIÓN DE AUTORÍA

Declaro ser el único autor de la presente tesis que tiene por título “Componente de visualización de las notificaciones del sistema XABAL eXcriba, para la plataforma Android de Alfresco Mobile” y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____del año 2018.

_____

Miguel Angel Hernández González

Firma del Autor

_____ _____

MSc. Darling Darias Pérez

Firma del Tutor

Ing. Frank Simón Ricardo

Firma del Tutor

Datos del contacto

Autor: Miguel Angel Hernández González

Email: mhgonzalez@estudiantes.uci.cu

Universidad de las Ciencias Informáticas, La Habana, Cuba.

Tutores:

Msc. Darling Darias Pérez

Email: ddarias@uci.cu

Universidad de las Ciencias Informáticas, La Habana, Cuba.

Ing. Frank Simón Ricardo

Email: franksm@uci.cu

Universidad de las Ciencias Informáticas, La Habana, Cuba.

Resumen

En la Universidad de las Ciencias Informáticas, de acuerdo a su acometido y misión social, se desarrolla una línea de productos enmarcados dentro del área de la gestión documental, siendo uno de dichos productos el Gestor de documentos administrativos en la web Xabal eXcriba v3.1, sistema de alto impacto enfocado en la gestión de documentos en su fase activa. La aplicación web sobre la que está desarrollada eXcriba en su versión 3.1, actualmente Alfresco Share v4.2f, no está optimizada para su acceso desde dispositivos móviles o en un espectro amplio de tamaños de pantalla. El objetivo de esta investigación fue desarrollar un componente utilizando el SDK de Alfresco para la plataforma móvil Android que posibilita obtener de manera rápida y relativamente sencilla, la configuración de los contenidos sobre los cuales se desea emitir notificaciones, así como los usuarios, o grupos de ellos, a los que se quieran dirigir las mismas. Para ello se utilizó como metodología de modelado UML, como Entorno Integrado de Desarrollo IDE el Android Studio v2.3.3, como lenguajes de programación JAVA y JavaScript, para la plataforma Android y Alfresco respectivamente. Al componente desarrollado se le realizaron las pruebas de software correspondientes haciendo uso del método de caja negra, mediante la técnica de partición equivalente. El grado de importancia de la investigación se deriva del uso de tecnologías y métodos de desarrollo nuevos sobre la plataforma Alfresco, y el impacto directo en la explotación de los dispositivos móbiles para recibir, enviar y emitir notificaciones.

Palabras claves: gestión documental, dispositivos móviles, notificaciones, contenidos, y tecnologías.

Abstract

At the University of Computer Sciences, according to its objective and social mission, you can access a line of products framed within the area of document management, one of the best products being the Manager of administrative documents on the Xabal website. EXcriba v3.1, high impact system focused on document management in its active phase. The web application on which eXcriba is developed in version 3.1, currently Alfresco Share v4.2f, is not optimized for access from mobile devices or in a wide spectrum of screen sizes. The objective of this research was to develop a component using the Alfresco SDK for the Android mobile platform that made it possible to obtain, in a quick and simple way, the option of the users who wanted to receive notifications, as well as groups of users, to whom they want to direct them. To do this, we used as UML modeling methodology, as IDE Development Integrated Environment Android Studio v2.3.3, as programming languages for JavaScript and JAVA, for Android and Alfresco respectively. The software was tested using the black box method, using the equivalent partition technique. The degree of importance of the research is derived from the use of new technologies and development methods on the Alfresco platform, and the direct impact on the exploitation of mobile devices to receive, send and issue notifications

Keywords: document management, mobile devices, notifications, content, and technologies.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Introducción

En la actualidad las Tecnologías de la Información y de las Comunicaciones (TICs) han tomado un papel importante en nuestra sociedad y se utilizan en multitud de esferas como son las empresas, la educación, la medicina, el comercio y el esparcimiento. El avance de dichas tecnologías, han permitido llevar la globalidad al mundo de las comunicaciones y facilitar la interconexión entre las personas e instituciones a nivel mundial. Su uso y explotación por parte de las nuevas empresas ha sido vital para el desarrollo de la informatización del país, brindándose una amplia gama de servicios, aplicaciones y tecnologías. (Belloch, 2018).

El auge de la telefonía móvil ha sido realmente sorprendente. Su evolución ha permitido disminuir en tamaño y peso, su rápido desarrollo ha incorporado funciones adicionales como mensajería instantánea (SMS), agenda, juegos, cámara fotográfica, acceso a Internet, reproducción de video e incluso GPS y reproductor mp3 (Dreig, 2007).

Los dispositivos móviles cuentan con un sistema operativo (SO) que determinan las capacidades multimedia de los mismos, y la forma de éstas de interactuar con el usuario. Existen multitud de opciones entre los que se destacan, Windows Phone, iPhone OS y el sistema móvil de Google, Android. En un estudio realizado por la International Data Corporation (IDC) en agosto de año 2017, como parte del continuo análisis del mercado asociado a las tecnologías móviles, la mayoría de suministros en el período de referencia corresponderán a los teléfonos inteligentes con la plataforma Android cuyas ventas, según IDC, alcanzarán 1.479 millones de unidades para el año 2021 (un 85,5% del mercado). "Los teléfonos inteligentes basados en el sistema operativo Android siguen dominando el mercado mundial, IDC espera que los envíos de teléfonos en esta plataforma crezcan en un 2,3% en 2018" (Sputnik Mundo, 2018).

Android es un sistema operativo de código abierto, se programa principalmente en Java, y fue presentado en junio del 2007 junto la fundación del Open Handset Alliance (un consorcio de 78 compañías de hardware, software y telecomunicaciones) dedicadas al desarrollo de estándares abiertos para dispositivos móviles. Parte de su éxito se debe en gran medida a que posee la mayor tienda en línea de aplicaciones oficiales, que brindan un mayor número de funcionalidades al sistema y son compatibles con una gran variedad de hardware en el mercado (Android, 2017). Es la razón por la cual se apuesta por Android como tecnología para los dispositivos móviles.

Desde la década del 1990 las empresas e instituciones, así como el hombre en el plano personal, se han preocupado por el constante incremento de la información que se maneja a diario en la sociedad sin importar su procedencia o destino. Las entidades de la administración pública son las que fundamentalmente se ven en el centro de este fenómeno que es el aumento indiscriminado de la producción documental, al cual los archiveros y profesionales de la tecnología han dado solución, en

alguna medida, con el desarrollo de sistemas informáticos que permitan un control y tratamiento uniforme de la documentación administrativa.

Si bien ha proliferado la producción documental, ha cobrado también un auge extraordinario el desarrollo de sistemas para la gestión de documentos, sin embargo, la tendencia actual es el desarrollo de sistemas híbridos que gestionen los documentos soportados tanto en papel como en formato electrónico y que su gestión comience desde el propio acto de creación hasta su disposición final o permanente en un archivo histórico. Esta situación ha dado lugar al surgimiento de los llamados Sistemas Integrales de Gestión de Documentos y Archivos que en la actualidad cumplen con estrictos requisitos para otorgar el grado de confiabilidad necesario que requieren los nuevos soportes documentales (EcuRed, 2018).

La Universidad de las Ciencias Informáticas, en concordancia con su misión social de producir servicios y aplicaciones informáticas de alta calidad y profundo impacto en el entorno socio-económico del país, desarrolla una línea de productos enmarcados dentro del área de la gestión documental. Dichos productos, y los servicios asociados a ellos, están dirigidos fundamentalmente a empresas e instituciones tanto nacionales como extranjeras con el objetivo de almacenar, gestionar y auditar documentos electrónicos e imágenes de documentos capturadas a través de un escáner, cubriendo las diferentes etapas del ciclo de vida de un documento. Uno de estos productos es el Gestor de documentos administrativos en la web Xabal eXcriba v3.1, orientado a la gestión de documentos, envío de correos electrónicos, flujos de trabajo y elementos de auditoría. Actualmente, la aplicación web sobre la que está desarrollado el eXcriba (Alfresco Share v4.2f) no está optimizada para su acceso desde dispositivos móviles o en un espectro amplio de tamaños de pantalla. Además se tiene como consideración las dificultades subyacentes para la implementación de sitios “responsive” o con diseño elástico usando las tecnologías heredadas en que está basada la aplicación web Alfresco Share, base de Xabal eXcriba 3.1. Alfresco ha desarrollado, en estrecha colaboración con la comunidad de desarrollo de la versión community, un SDK específicamente enfocado en la agilización del desarrollo de aplicaciones móviles tanto para la plataforma Android como iOS. Este SDK, y la aplicación base que lo acompaña, aunque cuenta con todas las funcionalidades necesarias para su explotación, no contiene las diferentes funcionalidades añadidas por el equipo de desarrollo del centro CIGED en la versión 3.1 de la aplicación eXcriba, especialmente al sistema asociado para recepción, envío y emisión de notificaciones, sistema de alto valor agregado tomando en consideración las facilidades y ventajas de las plataformas móviles.

Teniendo en cuenta la problemática planteada surge el siguiente **problema de la investigación:** ¿Cómo automatizar el proceso de recibir, enviar y emitir notificaciones del sistema XABAL eXcriba para dispositivos móviles?

Luego de planteado el problema de la investigación, se define como **objeto de estudio:** el proceso de notificaciones del sistema XABAL eXcriba**.**

Para dar solución al problema se plantea como **objetivo general:** desarrollar un componente para la recepción, envío y emisión de notificaciones del sistema XABAL eXcriba, utilizando el SDK de Alfresco para dispositivos móviles.

Enmarcado como **campo de acción:** la recepción, envío y emisión de notificaciones del sistema XABAL eXcriba para dispositivos móviles.

Para dar cumplimiento al objetivo general se plantean las siguientes **tareas de investigación:**

1. Diagnóstico de los principales conceptos y definiciones asociados a la gestión documental para la identificación de los procesos a utilizar en la investigación.
2. Análisis de las principales herramientas y lenguajes a utilizar para determinar la que se empleará para el desarrollo de la propuesta de solución.
3. Descripción del proceso de recepción, envío y emisión de notificaciones del sistema XABAL eXcriba v3.1.
4. Diseño del componente con artefactos ingenieriles para la recepción, envío y emisión de notificaciones del sistema XABAL eXcriba v3.1 hacia dispositivos móviles.
5. Desarrollo del componente para el proceso de recepción, envío y emisión de notificaciones del sistema XABAL eXcriba v3.1 hacia dispositivos móviles.
6. Validación del componente para el proceso de recepción, envío y emisión de notificaciones del sistema XABAL eXcriba v3.1 hacia dispositivos móviles con el uso de pruebas de software definidas en la investigación.

Para el desarrollo de la investigación se utilizaron diferentes métodos científicos de investigación que sirvieron para comprender la realidad del estudio de la sociedad y el pensamiento, con el fin de manifestar su esencia y sus relaciones. Dichos métodos se clasifican en teóricos y empíricos, en la realización de la investigación fue necesario el uso de varios métodos para la obtención y extensión del conocimiento, entre los **métodos teóricos** está:

Analítico-Sintético: Este método sirvió para la recopilación de información requerida durante la realización del estudio del estado del arte y para concretar los elementos más importantes relacionados con la recepción, envío y emisión de notificaciones en los sistemas de gestión de documentos. Además del análisis de un conjunto de documentos, herramientas, metodologías y tecnologías que permitió sintetizar el contenido que brinda soporte a la propuesta de la investigación a desarrollar.

Modelación: Fue utilizado en la representación, mediante el uso de diagramas, de las características del sistema a desarrollar, relaciones entre objetos; y las actividades que intervinieron en el proceso de notificaciones del sistema XABAL eXcriba.

Entre los **métodos empíricos** utilizados en la investigación está:

El método observación científica: El cual fue el primero en ser utilizado por los científicos y en la actualidad continúa siendo su instrumento universal. Permite conocer la realidad mediante la percepción directa de entes y procesos, para lo cual debe poseer algunas cualidades que le dan un carácter distintivo. En este caso, el conocimiento e información de las notificaciones del sistema XABAL eXcriba para dispositivos móviles.

Dentro del método empírico observación científica se encuentra:

Método Experimental

Es el más complejo y eficaz de los métodos empíricos. En este método el investigador interviene sobre la recepción, envío y emisión de notificaciones en los sistemas de gestión de documentos, modificándolo directa o indirectamente para crear las condiciones necesarias que permita revelar sus características fundamentales y sus relaciones esenciales, bien sea:

a) Aislando la recepción, envío y emisión de notificaciones en los sistemas de gestión de documentos y las propiedades que estudia, de la influencia de otros factores

b) Reproduciendo la recepción, envío y emisión de notificaciones en los sistemas de gestión de documentos en condiciones controladas

c) Modificando las condiciones bajo las cuales tiene lugar el proceso que se estudia.

La investigación se estructura, como se muestra a continuación:

Capítulo 1. Fundamentación teórica: Se realiza un detallado estudio de los principales conceptos y aspectos asociados a la recepción, envío y emisión de las notificaciones del sistema XABAL eXcriba, así como a los sistemas existentes, tanto internacionales como nacionales. Se realiza un análisis detallado sobre la metodología, herramientas, tecnología y lenguajes de programación para el desarrollo del componente.

Capítulo 2. Características y diseño del componente: Se realiza una descripción detallada de la propuesta de solución definiendo los elementos técnicos de la misma: el modelado de los procesos de negocio. Se representan en un modelo de dominio los principales conceptos que se manejan en el contexto del sistema. Se determinan los requisitos funcionales y no funcionales que debe cumplir el sistema. Se define la arquitectura, el patrón arquitectónico de la aplicación y los patrones de diseño utilizados. Se realiza un diagrama de caso de uso a partir de los requisitos obtenidos y se definen los casos de uso y los actores que se relacionan con cada uno de ellos.

Capítulo 3. Implementación y pruebas del componente: Se describe todo lo relacionado con las pruebas efectuadas al proceso de implementación, para poder ratificar la calidad y eficiencia de acuerdo con las necesidades que requiere el cliente. Se representan los elementos físicos necesarios para un

correcto despliegue de la aplicación, empleando para ello el diagrama de despliegue. Se muestra el diagrama de componentes de la implementación. Se realiza la validación y pruebas de la solución de acuerdo a los requisitos que debe cumplir para garantizar una calidad óptima, utilizando para ello pruebas funcionales.

Capítulo 1: Fundamentación teórica

En el presente capítulo se exponen conceptos para lograr una mejor comprensión sobre los términos tratados en el proceso de recepción, envío y emisión de notificaciones en el sistema XABAL eXcriba para dispositivos móviles y sistemas similares. Se profundizan los aspectos y conceptos más importantes para la investigación. Además de un análisis detallado sobre la metodología de desarrollo de software, las herramientas y tecnologías que complementan el desarrollo e implementación del componente de visualización de las notificaciones del sistema XABAL eXcriba.

1.1 Conceptos fundamentales para el desarrollo de la aplicación

1.1.1 Gestión Documental

El conjunto de normas, técnicas y prácticas usadas para administrar el flujo de documentos de todo tipo en una organización. Permite la recuperación de información desde ellos, determina el tiempo que los documentos deben guardarse, eliminar la que está en desuso y asegurar la conservación indefinida de los documentos más valiosos, aplicando principios de racionalización y economía (Pixelware, 2018).

La gestión documental pone a su disposición la posibilidad de procesar el contenido de sus líneas de negocio durante todo el ciclo de vida, desde su creación, transformación, aprobación, revisión, publicación, búsqueda y recuperación, hasta archivo y destrucción.

La norma ISO (Organización Internacional de Normalización) 15489 se refiere a la gestión documental como el área de gestión responsable de un control eficaz y sistemático de la creación, la recepción, el mantenimiento, el uso y la disposición de documentos de archivo, incluidos los procesos para incorporar y mantener en forma de documentos la información y prueba de las actividades y operaciones de la organización (Aenor, 2016).

1.1.2 Dispositivos Móviles

Un dispositivo móvil es un pequeño dispositivo de computación portátil que generalmente incluye una pantalla y un método de entrada (ya sea táctil o teclado en miniatura). Usualmente, poseen capacidad de conexión a uno o más tipos de redes inalámbricas con un amplio espectro de frecuencias de operación y una amplia capacidad de interacción a través de la pantalla o el teclado. Poseen además sistemas operativos que pueden ejecutar aplicaciones, las que hacen posible que se utilicen como dispositivos para juegos, reproductores multimedia, calculadoras, navegadores y más (Biblioteca multimedia SUD, 2018). En muchas ocasiones pueden ser sincronizados con algún sistema de la computadora para actualizar aplicaciones y datos. Normalmente se asocian al uso individual de una persona, tanto en posesión como en operación, la cual puede adaptarlos a su gusto. (Dispositivos móviles , 2018)

1.1.3 Diseño Adaptativo Elástico

El diseño web responsive o adaptativo es una técnica de diseño web que busca la correcta visualización de una misma página en distintos dispositivos. Hoy en día se accede a sitios web desde todo tipo de dispositivos; ordenador, tablet, smartphone, etc. Cada uno de estos dispositivos muestra el sitio web de una forma diferente, si esto ocurre así estamos ante una Web Responsive, es decir un sitio web capaz de adaptarse al dispositivo en el que se está visualizando. La estructura de la página debe ser flexible, es decir, el diseño debe permitir que los anchos de pantalla sean modificables, por lo tanto, no se puede tener un ancho fijo en tablas o columnas y además se debe permitir que el número de columnas pueda disminuirse en función del tamaño de la pantalla del dispositivo desde el que se acceda. Reduce el tiempo de desarrollo, evita los contenidos duplicados, y aumenta la viralidad de los contenidos ya que permite compartirlos de una forma mucho más rápida y natural, los layouts (contenidos) e imágenes son fluidos, cada vez más, surgiendo la necesidad de que la web se adapte a los diferentes tamaños de pantalla, además cuenta con algunos beneficios como mejorar la experiencia del usuario ya que se adapta la web a las características del dispositivo y ahorra costos de mantenimiento, cuando se incluye contenidos nuevos se realiza una sola actualización para todas las versiones de la web (digival.es, 2018).

Características de un diseño web responsive: (Internacionalweb, 2018)

- ✓ **Adaptar el ancho de la web al ancho del dispositivo:** El diseño pasa de ser un diseño horizontal a un diseño vertical, la web se adapta al ancho del dispositivo, el desplazamiento horizontal desaparece, siendo más cómoda la navegación y la visualización de la web. La fluidez de contenido hace que el usuario no tenga que desplazarse horizontalmente.
- ✓ **Reorganización de los contenidos y elementos de la web:** Los contenidos deben de disponerse de tal forma que se puedan ver correctamente en todas las pantallas. Reajustar el tamaño y la separación entre los contenidos (Textos, imágenes, botones…) según el ancho del dispositivo para que la navegación resulte fácil.
- ✓ **Simplificar:** En una versión móvil no se puede mostrar todo lo que aparece en una pantalla de ordenador, hay que priorizar y seleccionar los elementos que sean necesarios, eliminar objetos gráficos que únicamente son estéticos para poder tener menos carga visual.
- ✓ **Cambio de apariencia de algunos elementos:** El menú, por ejemplo, pasa de ser un menú visible a desplegable, un icono que al pinchar sobre él, se despliegan los elementos del menú para poder acceder a los enlaces. Este cambio tiene por objetivo ahorrar espacio y optimizar, labor de gran importancia en una versión móvil.

Ventajas de un diseño web responsive: (Internacionalweb, 2018)

- ✓ **Aumento de la velocidad de carga:** Como ya hemos dicho antes en un diseño responsive se eliminan los elementos que no son necesarios por tanto al haber menos elementos que descargar, la web se carga más rápido.
- ✓ **No afecta negativamente al optimizador de motor de búsqueda SEO (Search Engine Optimization):** Google ha informado a los webmaster que las webs con un diseño responsive gozarán de una mejor posición de sus resultados de búsqueda en estos dispositivos móviles.
- ✓ **No hay duplicidad en la URL:** Evita el contenido duplicado, únicamente existe una web y esta es la que se adapta a los distintos dispositivos.
- ✓ **Ahorro de tiempo y de costes de mantenimiento:** Debido a que la gestión de un único sitio web es más económico.

1.1.4 Notificaciones

Una notificación es un proceso mediante el cual se le informa a alguien acerca de una determinada circunstancia que le incumbe. Dicho proceso puede darse en una pluralidad de contextos, público, privado, a nivel de una persona física, jurídica, etc. (Definicion, 2018).
Las notificaciones push son las comunicaciones que se envían desde un servidor remoto y que reciben los dispositivos que tienen la aplicación instalada. La inmediatez es lo más destacable en este tipo de notificaciones, ya que no se tiene que actualizar ni ejecutar la aplicación para que llegue la información. Esta comunicación con los usuarios de la aplicación permite tener una relación constante y cercana con ellos. Existen distintos usos que se puede dar a estas notificaciones y muchas ventajas, tanto para la empresa que haya creado la aplicación, como para el usuario final de la misma (kingofapp.es, 2018).

Ventajas de las notificaciones push: (kingofapp.es, 2018)

- ✓ **Llegan a un mayor número de usuarios:** Se pasan mucho tiempo conectados al móvil y, por esa razón es lógico que sea más sencillo llegar al público a través de estos mensajes inmediatos en lugar de con otros canales. Aunque tenga la aplicación cerrada seguirá recibiendo la información que se quiera enviar.
- ✓ **Personalizan el mensaje:** Las notificaciones push favorecen la segmentación y la personalización de la información que se envíe. Dependiendo del objetivo que se tenga o del público al que se quiera dirigir, se podrá incluir palabras o links específicos que faciliten la comunicación. El servidor controla los datos que se hayan recogido previamente para enviar la información adecuada a cada segmento y aumenta el ratio de apertura en un 50%.
- ✓ **Permiten la automatización:** Algunos medios automatizan las notificaciones push utilizando la segmentación, la personalización y la geolocalización. De esta manera se pude automatizar los mensajes que se envíen y personalizar el contenido, el momento del envío y el lugar según el público al que se destine esta comunicación.

- ✓ **Son inmediatas:** El usuario no necesita revisar continuamente la aplicación ni estar utilizándola en ese momento para que le lleguen las notificaciones.
- ✓ **Ahorran batería:** Como no es necesario realizar consultas periódicas ni actualizar la aplicación, las notificaciones push ahorran en recursos y batería del móvil. Aunque pueda parecer una ventaja menos importante, sí la tiene para el consumidor. No tendrá que estar continuamente utilizando su smartphone ni consumir energía.

1.2 Estudio de estado del arte

En este acápite se abordaran algunos sistemas existentes y similares para el proceso de notificaciones, teniéndose en cuenta el aporte o no para el desarrollo de la propuesta de solución.

1.2.1 RabbitMQ

RabbitMQ es un software de negociación de mensajes de código abierto, y entra dentro de la categoría de middleware de mensajería. Implementa el estándar Advanced Message Queuing Protocol (AMQP). El servidor RabbitMQ está escrito en Erlang y utiliza el framework Open Telecom Platform (OTP) para construir sus capacidades de ejecución distribuida y conmutación ante errores. Rabbit Technologies Ltd., la compañía que lo desarrolla, fue adquirida en abril de 2010 por la división SpringSource de VMWare. Es esta la última compañía la que desarrolla y da soporte para RabbitMQ. El código fuente está liberado bajo la licencia Mozilla Public License.
A partir de Mayo del 2013, por una unión de empresas, Pivotal es el nuevo patrocinante del proyecto, con más de 35,000 implementaciones de producción de RabbitMQ en todo el mundo en pequeñas y grandes empresas, RabbitMQ es el intermediario de mensajes de código abierto más popular. RabbitMQ es liviano y fácil de implementar en las instalaciones y en la nube. Es compatible con múltiples protocolos de mensajería. RabbitMQ se puede implementar en configuraciones distribuidas y federadas para cumplir con los requisitos de alta disponibilidad y alta escala. RabbitMQ se ejecuta en muchos sistemas operativos y entornos de nube, y proporciona una amplia gama de herramientas de desarrollo para la mayoría de los idiomas populares (Rabbitmq, 2018).

- ✓ Advanced Message Queuing Protocol (AMQP) _"es un protocolo de estándar abierto en la capa de aplicaciones de un sistema de comunicación. Las características que definen al protocolo AMQP son la orientación a mensajes, encolamiento ("queuing"), enrutamiento (tanto punto-a-punto como publicación-subscripción), exactitud y seguridad".
- ✓ Erlang _"es un lenguaje de programación concurrente y un sistema de ejecución que incluye una máquina virtual (BEAM) y bibliotecas (OTP)".

- ✓ Framework Open Telecom Platform (OTP) _"es un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar".
- ✓ SpringSource de VMWare _"es una filial de EMC Corporation que proporciona software de virtualización disponible para ordenadores compatibles X86".
- ✓ Mozilla Public License _"es una licencia de software libre, de código abierto, detallada, desarrollada y mantenida por la Fundación Mozilla".

1.2.2 Alfresco Mobile

Alfresco Mobile SDK está disponible para iOS y Android. Es una capa de interfaz opcional para el desarrollo de aplicaciones móviles que, combinada con la API Alfresco, hace que el desarrollo en contra de la plataforma Alfresco sea aún más fácil. El Alfresco Mobile SDK proporciona la capacidad de conectarse tanto a servidores locales como a la nube de Alfresco. La API de cliente proporciona una biblioteca que permite la incorporación rápida de acceso a servidores Alfresco en aplicaciones móviles. Android SDK requiere Android 4.0 y superior para ejecutar (Google Play, 2018).

- **Principales operaciones soportadas:**
  -Crear nuevas carpetas.
  -Cargar documentos.
  -Buscar documentos y carpetas.
  -Sincronizar ficheros y carpetas para acceder a ellos sin conexión.
  -Trabajar con sus ficheros en aplicaciones de Microsoft y Adobe.
  -Capturar imágenes, vídeo y audio.
  -Enviar documentos para revisión y aprobación.
  -Ver sus tareas.
  -Hacer comentarios en los documentos.
  -Enviar documentos por correo electrónico.
  -Unirse a sitios y marcarlos como favoritos.
- **Otras funciones prácticas que realiza:**
  -Utiliza el motor de texto a voz y el lector de pantalla de Android.
  -Personaliza la pantalla de inicio con los widgets de Alfresco para acceder rápidamente al contenido esencial o para capturar con un toque fotos y texto.
  -Escanea directamente al repositorio con el escáner inalámbrico Fujitsu ScanSnap.
- **Seguridad:**
  -Protege su aplicación con un PIN.
  -Cifra los datos guardados con el estándar de cifrado avanzado de 128 bits.
  -Conecta a su repositorio de Alfresco usando las credenciales de inicio de sesión de Alfresco.

-Se necesita Alfresco 4.2 o una versión posterior.

-Se encuentra disponible para usuarios de las ediciones Alfresco Standard y Enterprise (en la nube y a nivel local).

1.2.3 AC Display

Es una aplicación que implementa las funcionalidades nativas de Ambient Display a todos los dispositivos Android. La pantalla se activará por cada notificación que se reciba, y se agruparan las notificaciones en categorías según la aplicación que haya emitido la notificación. De este modo, si se tiene el smartphone a la vista se podrá ver de qué aplicación en concreto ha recibido la notificación. Además, esta aplicación también se puede configurar para que la pantalla se encienda automáticamente al coger el móvil cuando está en reposo (mientras lo tenemos apoyado sobre una mesa, por ejemplo). De esta forma, se ahorra tener que desbloquear la pantalla manualmente para consultar todas las notificaciones pendientes por atender (El Androide Libre, 2018).

1.2.4 Conclusiones del estudio del estado del arte

Luego de un estudio realizado sobre las aplicaciones y herramientas que se llevaron a cabo para el desarrollo de dicha aplicación se arriba a las siguientes conclusiones:

Al realizar un análisis de las funcionalidades de cada una de las aplicaciones y una evaluación de la aplicación web XABAL eXcriba 3.1 de acuerdo a métricas de diseño y usabilidad en dispositivos móviles, se evidencia la necesidad de la implementación de una nueva solución para lograr el proceso de recepción, envío y emisión de notificaciones para dispositivos móviles. De esta forma, si bien pudiera ser RabbitMQ la solución perfecta para el desarrollo de la misma, la definición e implementación de las funcionalidades asociadas a la integración Alfresco-RabbitMQ no sería la más óptima ya que Alfresco Mobile tiene un conjunto de funcionalidades asociadas a la plataforma Alfresco la cual solo habría que agregarle el módulo de notificaciones para el desarrollo de dicho componente, razón por la que se considera que el componente de notificaciones del sistema XABAL eXcriba en su versión 3.1 es suficiente y necesario para arribar a la solución propuesta.

1.3 Metodologías de desarrollo de software

En la actualidad existen numerosas propuestas de metodologías que inciden de distintas formas en el proceso de desarrollo del software. Las metodologías de desarrollo son marcos de trabajo que constituyen un conjunto de procedimientos y técnicas usadas para estructurar, planificar y controlar este proceso de desarrollo de software generando la documentación necesaria para el mismo. Estos marcos de trabajo indican qué personas deben participar en el desarrollo de las actividades y qué papel deben de tener. Detallan la información que se debe producir como resultado de una actividad y la información

necesaria para comenzarla. Dentro de las metodologías existen dos grandes grupos, las metodologías tradicionales o pesadas y las ágiles.

Las metodologías tradicionales se centran fundamentalmente en el control y definición de los procesos, tareas y herramientas a utilizar. Requieren de una extensa documentación previendo todo de antemano y pueden ser muy efectivas para proyectos de gran tamaño. Proponen un gran cúmulo de documentación de acuerdo al tamaño del proyecto y como consecuencia del equipo de desarrollo, es recomendable utilizarla en el desarrollo de proyectos medianos y grandes.

1.3.1 Metodología de desarrollo AUP-UCI

Proceso Unificado Ágil (AUP) en su variante UCI, es una versión simplificada al desarrollo del software basado en el Proceso Unificado Rational de IBM (RUP). Este describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software de negocio usando disciplinas y entregables incrementales con el tiempo. AUP se preocupa especialmente de la gestión de riesgos. Propone que aquellos elementos con alto riesgo obtengan prioridad en el proceso de desarrollo y sean abordados en etapas tempranas del mismo.

En la Universidad de la Ciencias Informáticas (UCI) se desarrolló una versión de esta metodología ágil con el objetivo de crear un estándar en el proceso productivo de manera que se adapte al ciclo de vida de los proyectos en la UCI. En esta variación de AUP para la UCI se mantiene la primera fase de inicio, la segunda fase de ejecución en esta variación para la UCI resume las fases restantes de elaboración, construcción y transición. Finalmente se encuentra la fase de cierre donde se analiza los resultados del proyecto y se realizan las actividades formales de cierre de proyecto (Sanchez, 2015).

AUP-UCI propone para el ciclo de vida de los proyectos en la UCI tener siete disciplinas, los flujos de trabajo son los siguientes:

- ✓ **Modelado de negocio:** Disciplina destinada a comprender los procesos de negocio de la organización.
- ✓ **Requisitos:** Disciplina que comprende la administración y gestión de los requisitos funcionales y no funcionales del producto.
- ✓ **Análisis y diseño:** En esta disciplina se modela el sistema y su forma para que pueda soportar todos los requisitos, incluyendo los requisitos no funcionales.
- ✓ **Implementación:** En la implementación, a partir de los resultados del Análisis y Diseño se construye el sistema.
- ✓ **Pruebas interna:** Se verifica el resultado de la implementación probando cada construcción, incluyendo tanto las construcciones internas como intermedias, así como las versiones finales a ser liberadas.
- ✓ **Pruebas de liberación:** Pruebas diseñadas y ejecutadas por una entidad certificadora de la calidad externa.

- ✓ **Pruebas de aceptación:** La prueba final antes del despliegue del sistema. Su objetivo es verificar que el software está listo y que puede ser usado por usuarios.

A partir de que el modelado de negocio propone tres variantes a utilizar en los proyectos (Caso de Uso del Negocio (CUN), Diagrama por Proceso del Negocio (DPN) o Modelo Conceptual (MC)) y existen tres formas de encapsular los requisitos (Caso de Uso del Sistema (CUS), Historias de Usuarios (HU), Descripción de los Requisitos de Proceso (DRP)), surgen cuatro escenarios para modelar el sistema en los proyectos, manteniendo en dos de ellos el MC.
El escenario 1 propone a los proyectos que modelan el negocio con casos de usos del negocio que solo pueden modelar el sistema con casos de usos del sistema. El escenario 2 propone modelar el negocio con un modelo conceptual y el sistema con casos de uso del sistema. El escenario 3 propone modelar el negocio con descripción de proceso de negocio, junto al modelo conceptual y el sistema mediante la descripción de requisitos por proceso. El escenario 4 propone no modelar el negocio y describir el sistema mediante historias de usuario.

Se selecciona para esta investigación el escenario número dos ya que este aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan que no es necesario incluir las responsabilidades de las personas que ejecutan las actividades, de esta forma modelarían exclusivamente los conceptos fundamentales del negocio. Se recomienda este escenario para proyectos donde el objetivo primario es la gestión y presentación de información.

Con la adaptación de AUP que se propone para la actividad productiva de la UCI se logra estandarizar el proceso de desarrollo de software, dando cumplimiento además a las buenas prácticas que define CMMI-DEV v1.3. Se logra hablar un lenguaje común en cuanto a fases, disciplinas, roles y productos de trabajos. Se redujo a uno la cantidad de metodologías que se usaban y de más de 20 roles en total que se definían se redujeron a 11.

Para lograr que el sistema XABAL eXcriba cumpla con todo el proceso de desarrollo, se utilizaron un conjunto de lenguajes, herramientas y tecnologías, los cuales están definidos por el centro y se describen a continuación.

1.4 Lenguajes de desarrollo

1.4.1 Java

Java es un lenguaje de programación orientado a objetos, el cual agrupa en estructuras encapsuladas tanto sus datos como los métodos (o funciones) que manipulan esos datos. Diseñado para crear software altamente fiable. Para ello proporciona numerosas comprobaciones en compilación y en tiempo de ejecución (Horstman Cay S., 2015). El código binario de Java no es un lenguaje de alto nivel, sino un verdadero código máquina de bajo nivel, viable incluso como lenguaje de entrada para un

microprocesador físico. Se determinó utilizar Java 8 para la implementación del componente para dispositivos que operan con Android teniendo en cuenta que es un lenguaje orientado a objetos de gran potencialidad, robustez, seguro y multiplataforma, el equipo de desarrollo tiene conocimientos sobre este lenguaje y posee experiencias, capacidades y habilidades con el desarrollo de aplicaciones en Java. Además porque es el lenguaje que utiliza el centro para el desarrollo de aplicaciones con sistema operativo Android, utilizando el SDK de Alfresco (Java, 2017).

1.4.2 JavaScript

JavaScript es un sencillo lenguaje de programación, que presenta una característica especial: sus programas, llamados comúnmente scripts, se pueden adicionar a las páginas HTML y se ejecutan en el navegador (Mozilla Firefox, Microsoft Internet Explorer,...). Estos scripts normalmente consisten en unas funciones que son llamadas desde el propio HTML cuando algún evento sucede. De ese modo, podemos añadir efectos como que un botón cambie de forma al pasar el ratón por encima, o abrir una ventana nueva al pulsar en un enlace, JavaScript fue desarrollado por Netscape, a partir del lenguaje Java, el cual sigue una filosofía similar la diferencia fundamental es que Java es un lenguaje completo, que puede ser utilizado para crear aplicaciones de todo tipo, mientras que JavaScript sólo "funciona" dentro de una página HTML (Maestros del web, 2018).

1.4.3 Lenguaje de modelado

Para graficar las entidades se emplearán el Lenguaje Unificado de Modelado (UML, por sus siglas en inglés), es un lenguaje utilizado para especificar, visualizar, construir y documentar artefactos de un sistema de software. Está constituido por un conjunto de diagramas y es necesario contar con todos esos diagramas dado que cada uno se dirige a cada tipo de persona implicada en el sistema. Un modelo UML indica qué es lo que supuestamente hará el sistema, mas no cómo lo hará (Visual Paradigm Copyright, 2018). UML proporciona el proceso de diseño de tal forma que los analistas, clientes, desarrolladores y otras personas involucradas en el desarrollo del sistema lo comprendan y convengan con él. Por las características antes mencionadas y además porque el equipo de desarrollo cuenta con experiencia en el desarrollo de este lenguaje se decide emplearlo en su versión 2.1. Para efectuar el modelado de los procesos de negocio se empleó BPMN (Notación para el Modelado de Procesos de Negocio), la cual es una notación gráfica estandarizada para el modelado de procesos de negocio. BPMN sirve como lenguaje común para cerrar la brecha de comunicación que frecuentemente se presenta entre el diseño de los procesos de negocio y su implementación.

1.5 Herramientas y tecnologías de desarrollo

Las herramientas de desarrollo de software (HIS) han desempeñado desde sus inicios un importante

papel en el desarrollo de aplicaciones. Como parte de la ingeniería de software (IS), las HIS han experimentado también continuos cambios, consecuencia del creciente avance tecnológico propio de los últimos años. Este cambio ha sido un factor con especial influencia sobre la IS, así como también sobre otras disciplinas relacionadas, e impulsa una alta tasa de cambio en las HIS. Actualmente éstas son numerosas y apoyan en múltiples formas diferentes dimensiones del desarrollo de software en general (Pérez, 2017).

1.5.1 Herramienta CASE

Las herramientas de Ingeniería de Software Asistida por Computadora (CASE, por sus siglas en inglés) son empleadas con el fin de automatizar los aspectos claves de todo el proceso de desarrollo de software de un sistema, desde su inicio, hasta completar todo el proceso de implementación. Para el desarrollo de la propuesta de solución se empleará Visual Paradigm, una herramienta profesional que soporta el ciclo de vida completo del desarrollo de software: negocio, descripción de los requisitos, análisis y diseño orientados a objetos, implementación, pruebas y despliegue. Las principales características de Visual Paradigm son: (Object Management Group. Unified Modeling Language, 2018)

- Entorno de creación de diagramas UML 2.1.
- Lenguaje estándar, común a todo el equipo de desarrollo.
- Existen varias versiones compatibles con diferentes plataformas, usadas para necesidades específicas del equipo de desarrollo.
- Permite la integración con Eclipse.

Como demuestran las características antes mencionadas la herramienta Visual Paradigm en su versión 8.0 es un software potente utilizándose a nivel mundial. Posibilita la realización satisfactoria de los diferentes diagramas en dependencia del flujo de trabajo de la metodología que se emplea. Por estas razones es la herramienta seleccionada para confeccionar los artefactos relacionados con el proceso del negocio, y el diagrama de despliegue.

1.5.2 Android Studio 2.3.3

Es el entorno de desarrollo integrado oficial para el desarrollo de aplicaciones Android, basado en IntelliJ IDEA, desarrollado por Google y fue anunciado en mayo de 2013. Es un IDE distribuido de forma libre bajo la licencia de Apache License 2.0. Tiene integrado una herramienta llamada Gradle para la construcción y compilación de la aplicación. Contiene plantillas para ayudar a construir aplicaciones con características comunes. Presenta la ventaja de que permite crear dispositivos virtuales para emular las aplicaciones, debido a que cuenta con un paquete de desarrollo de software, SDK por sus siglas en inglés (Uptodown, 2017).

1.5.3 Android SDK

El paquete de desarrollo de software incluye un conjunto de herramientas para ayudar al programador, tales como bibliotecas, un manejador de emuladores, documentación, 10 códigos ejemplos y tutoriales. Es un paquete multiplataforma. Permite la integración con Eclipse usando el Android Development Tools (ADT) y también con Android Studio. Se usa en el desarrollo de la solución debido a que es imprescindible para el entorno de desarrollo integrado seleccionado (Softonic, 2017).

1.5.4 XABAL eXcriba 3.1

Sistema que automatiza los procesos documentales de las organizaciones. Incluye las 3 primeras fases propuestas por la ISO 15489 para el ciclo de vida de un documento: creación, revisión/aprobación y publicación, manteniendo la seguridad de los documentos en formato electrónico. Entre sus principales funcionalidades se destacan el control de versiones, accesos y permisos, automatización de los flujos documentales, gestión de documentos y carpetas, realización de auditorías sobre los documentos electrónicos (EXCRIBA 3.1, 2018).

1.5.5 FreeMaker

Apache FreeMarker ™ es un motor de plantillas: una biblioteca Java para generar texto (páginas web HTML, correos electrónicos, archivos de configuración, código fuente, etc.) basado en plantillas y datos cambiantes. Las plantillas están escritas en el lenguaje de plantillas de FreeMarker (FTL), que es un lenguaje simple y especializado (no un lenguaje de programación completo como PHP). Por lo general, se usa un lenguaje de programación de propósito general (como Java) para preparar los datos (emita consultas de bases de datos, haga cálculos comerciales). Apache FreeMarker muestra los datos preparados usando plantillas. En la plantilla, se enfoca en cómo presentar los datos, y fuera de la plantilla en qué datos presentar (FreeMarker, 2018).

1.5.6 Alfresco

Alfresco es un sistema de administración de contenidos (ECM por sus siglas en inglés) de código fuente libre, desarrollado en Java y basado en estándares abiertos de escala empresarial para sistemas operativos tipo Windows, Unix Solaris y algunas versiones de Linux. Está diseñado para usuarios que requieren un alto grado de modularidad y rendimiento escalable.

Incluye un repositorio de contenidos, un portal web para administrar y usar contenido estándar en portales, una interfaz *Common Internet File System* (CIFS) que provee compatibilidad con los sistemas de archivos tanto de Windows como de Unix, un motor de flujos de trabajos propio, búsquedas de contenido a través de Apache Solr-Lucene, modificación y transformación de documentos usando LibreOffice e ImageMagic, así como todo un conjunto de subsistemas y funcionalidades auxiliares enfocados en la realización de la gestión documental durante las diferentes etapas del ciclo de vida de un contenido. Se distribuye en tres versiones, Community Edition (con licencia LGPL de código y

estándares abiertos), Entreprise Edition (posibilidad de soporte comercial y propietario a escala empresarial), y Cloud Edition (versión de software como servicio de Alfresco).

- **Funciones para entornos de producción:**
  -Alta disponibilidad y personalización con una sencilla administración.
  -Totalmente modular con servicios opcionales, cifrado de contenido, análisis, gestión de medios.

1.5.7 Alfresco Share

Alfresco Share es una plataforma moderna basada en la Web para la colaboración y la gestión social de contenidos. Posee potentes herramientas de equipo que incluyen biblioteca de documentos, blogs, wikis, calendario y un flujo de trabajo sencillo. Posee funcionalidades sociales como, por ejemplo, estado, etiquetas y fuentes de actividad de contenidos permiten que los equipos sean aún más eficaces. Alfresco Share aprovecha el repositorio de Alfresco para proporcionar servicios de contenido.

Características de Alfresco Share:

- ✓ Plataforma totalmente escalable y de código abierto.
- ✓ Utiliza e integra Alfresco como gestor y repositorio de contenidos.
- ✓ Alfresco Share se basa en el framework de Spring Surf que permite una potente integración con Alfresco.
- ✓ Spring surf permite desarrollar aplicaciones web bajo la metodología MVC (modelo-vista-controlador).
- ✓ Tessera extiende las funcionalidades de Alfresco Share siguiendo el mismo framework (Spring Surf) de desarrollo.

1.5.8 Spring Surf

Spring Surf es un marco web con secuencias de comandos que alimenta las capacidades de representación de niveles de presentación de las aplicaciones en el conjunto de aplicaciones de Alfresco. Estos incluyen Alfresco Share, Alfresco Records Management y herramientas de autoría y presentación para Alfresco Web Content Management.
Más específicamente, Surf es una extensión Spring Framework que puede usar para construir nuevas aplicaciones Spring Framework, o puede conectarlo a las aplicaciones Spring Web MVC existentes en su negocio. Surf proporciona una forma de construir interfaces de usuario para sus aplicaciones web utilizando scripts y plantillas del lado del servidor. Sin codificación Java, sin recopilación, sin reinicios constantes del servidor y sin instanciación de objetos de gran tamaño.
Surf sigue un enfoque basado en el contenido. Los scripts y las plantillas son solo archivos simples en el disco. Proporciona un modelo de objeto simple que permite definir páginas, plantillas, componentes y

temas usando codificación XML. La aplicación Spring recoge los nuevos archivos y los procesa a través de scripts y plantillas para producir la vista. Los scripts se escriben usando JavaScript del lado del servidor y Groovy. Las plantillas se escriben usando FreeMarker (Alfresco Comunity5.0, 2017).

1.6 Conclusiones del capitulo

En este capítulo se analizaron las aplicaciones similares a la propuesta de solución, los aportes para los aspectos específicos como la recepción, envío y emisión de notificaciones en el sistema XABAL eXcriba. Se definió la metodología de desarrollo de software AUP-UCI para guiar el desarrollo de la propuesta de solución de la investigación. Se analizaron las herramientas, tecnologías y lenguajes a utilizar para el desarrollo de la aplicación. Seleccionándose como lenguaje de desarrollo Java, ya que el Mobile SDK de Alfresco está desarrollado en Java, como Entorno Integrado de Desarrollo (IDE) Android Studio en su versión 2.3.3, el Visual Paradigm para el modelado de los artefactos y la arquitectura del componente y UML como lenguaje de modelado.

Capítulo 2: Características y diseño del componente

En el presente capítulo se propone la solución para desarrollar un componente para la recepción, envío y emisión de notificaciones, utilizando el SDK de Alfresco para dispositivos móviles, haciendo uso de la metodología AUP-UCI para lograr obtener un producto con buena calidad teniendo en cuenta las buenas prácticas y principios que contribuyen a agilizar el proceso. Además se expone el mapa conceptual para aumentar la comprensión del problema planteado. Se especifican los requisitos funcionales y no funcionales del software, descripción de los actores y diagramas de caso de uso del sistema.

2.1 Descripción de la Propuesta de solución

Para dar solución al problema de la investigación enunciado, se propone la extensión de la aplicación Alfresco Mobile, cuyo código fuente se encuentra disponible en (https://github.com/Alfresco/alfresco-android-app), realizándose las modificaciones necesarias para permitir visualizar y marcar como leídas las notificaciones emitidas desde la aplicación web eXcriba 3.1.

Para mejorar el proceso de emisión de notificaciones sobre los contenidos creados, se propone la implementación de una regla de contenido parametrizada utilizando el API JavaScript de Alfresco, que permita al personal de administración del sistema, de manera rápida y relativamente sencilla, la configuración de los contenidos sobre los cuales se desea emitir notificaciones, así como los usuarios, o grupos de ellos, a los que se quieran dirigir las mismas.

El dispositivo móvil será el encargado de consultar, a través de peticiones a un API REST y usando los componentes del Alfresco Mobile SDK, los servicios web necesarios para mostrar en un *Activity* del sistema operativo Android, un *Fragment* con las notificaciones existentes en el servidor para un usuario previamente autenticado.

De igual manera, la aplicación permitirá eliminar las notificaciones existentes para un usuario, enviando al servidor los datos necesarios para proceder a su eliminación del espacio de notificaciones del sistema XABAL eXcriba usando los servicios web del módulo nativo de notificaciones, brindándose un botón flotante, el cual permitirá al usuario recuperar una notificación que no haya sido deseada eliminar o por error del mismo. El usuario puede desplegar el botón y observar el asunto y el tema de la notificación, además de poder actualizar y verificar si le llego una nueva notificación.

API JavaScript Alfresco:

- ✓ Una API (‘Application Programming Interface’) es un conjunto de reglas (código) y especificaciones que las aplicaciones pueden seguir para comunicarse entre ellas: sirviendo de interfaz entre programas diferentes de la misma manera en que la interfaz de usuario facilita la interacción humano-software.
- ✓ La API de repositorio de JavaScript le permite desarrollar archivos compatibles con JavaScript para acceder, modificar y crear objetos de repositorio, como nodos, aspectos y propiedades.

- ✓ La API de Servicios de JavaScript de Alfresco proporciona una interfaz para servicios básicos a los que se puede acceder desde scripts web.

REST:

- ✓ REST es una interfaz entre sistemas que use HTTP para obtener datos o generar operaciones sobre esos datos en todos los formatos posibles, como XML y JSON.
- ✓ La API REST le permite acceder al contenido en un repositorio en las instalaciones, y en los Servicios de contenido de Alfresco en la nube, desde sus propias aplicaciones. La API es RESTful, lo que significa que cada llamada es una solicitud HTTP, por lo que ni siquiera necesita un lenguaje de programación para probarla. Puede escribir una dirección URL en un navegador web.

Activity:

- ✓ Es cada una de las pantallas o vistas que forman una aplicación. Es decir, que si una aplicación tiene 5 pantallas posee 5 activities.

Fragment:

- ✓ Un Fragment representa un comportamiento o una parte de la interfaz de usuario en una Activity. Se puede combinar múltiples fragmentos en una sola actividad o usar un fragmento en múltiples actividades. Es como una sección modular de una actividad que tiene su ciclo de vida propio, recibe sus propios eventos de entrada y se puede agregar o quitar mientras la actividad se esté ejecutando. Se podría definir como componentes de una actividad.

2.2 Modelo conceptual

Un modelo conceptual es la descripción de cómo se relacionan los conceptos en un problema del mundo real. Su objetivo principal es representar un problema de manera gráfica para un mejor entendimiento del dominio a través de un diagrama de clases (El Conspirador, 2018).

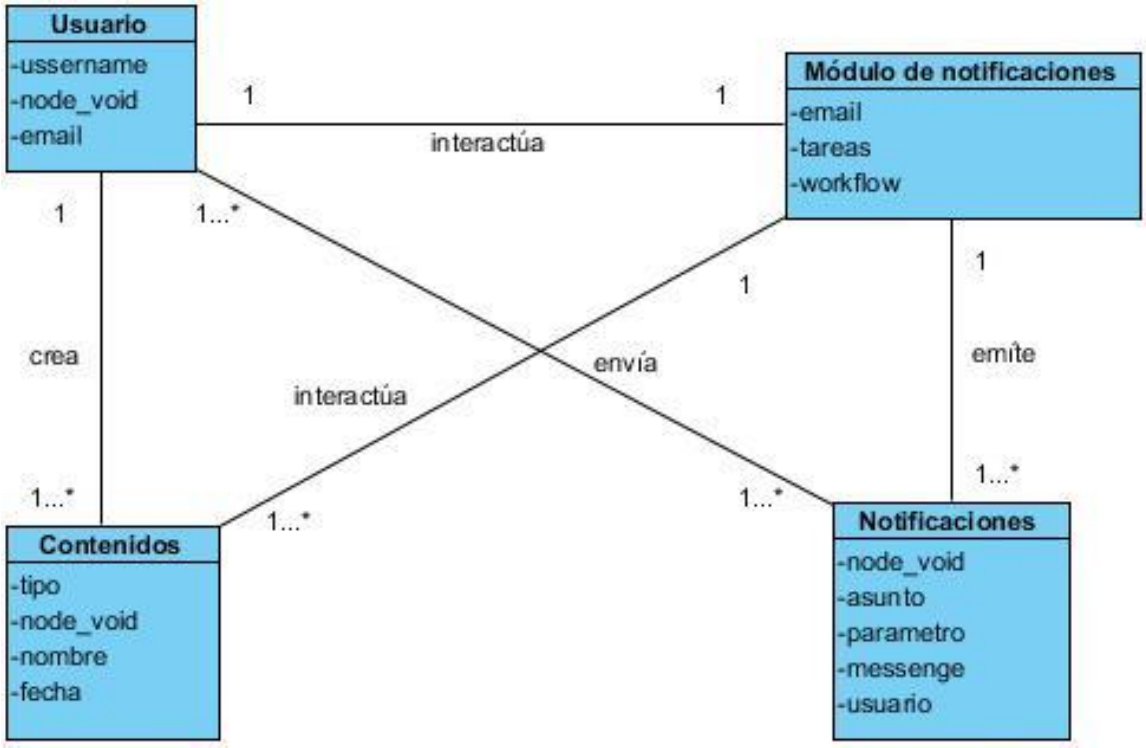


Figura 1: Diagrama del Modelo Conceptual

2.3 Especificación de los requisitos del software

Para hacer referencia al término requisito se utiliza la definición que aparece en el glosario de la IEEE, "… condición o necesidad de un usuario para resolver un problema o alcanzar un objetivo…" Los requisitos determinan el funcionamiento del sistema, ya que ponen en evidencia las restricciones sobre la implementación del mismo. Son un punto clave en el desarrollo de las aplicaciones informáticas y deben especificarse antes de comenzar la construcción del producto.

Pueden ser divididos en varios tipos, entre ellos los denominados requisitos funcionales y no funcionales. Requisitos funcionales: "Los requisitos funcionales de un sistema describen lo que el sistema debe hacer, (…), describen con detalle la función del sistema, así como sus entradas y salidas". Requisitos no funcionales: "Los requisitos no funcionales, son aquellos que no se refieren directamente a las funciones específicas que proporciona el sistema, sino a las propiedades emergentes de éste, como la fiabilidad, el tiempo de respuesta y la capacidad de almacenamiento".

Una buena especificación de requisitos de software ofrece como ventaja un contrato limpio entre clientes y desarrolladores, reduciendo el esfuerzo en el desarrollo y proporcionando un punto de referencia para procesos de verificación y validación. Además, una base para la identificación de posibles mejoras en los procesos analizados.

2.3.1 Requerimientos funcionales

Un requisito funcional define una función del sistema, el cual describe un conjunto de entradas, comportamientos y salidas que un sistema debe cumplir. En algunos casos, los requerimientos funcionales de los sistemas también pueden declarar explícitamente lo que el sistema no debe hacer (SCRIBD, 2018).

Tabla 1: Requisitos funcionales

Nº del requisito	Nombre del requisito	Descripción
RF1	Crear tipo de contenido	Permite al administrador del sistema, crear los contenidos sobre los cuales se desea recibir notificaciones.
RF2	Emitir notificación.	Permite el sistema emitir notificaciones sobre los tipos de contenidos definidos por el administrador del sistema a todos, o un conjunto de los usuarios del sistema.
RF3	Enviar notificación	Permite que el usuario envié una notificación.
RF4	Recibir notificación	Permite que el usuario reciba una notificación del sistema.
RF5	Eliminar notificación.	Permite al usuario eliminar notificaciones.
RF6	Acceder al sistema	Permite a un usuario acceder al sistema desde un dispositivo móvil.
RF7	Confirmar notificación como leída.	Permite al usuario confirmar que haya leído las notificaciones.

2.3.2 Requerimientos no funcionales

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener y que a su vez lo hacen atractivo, rápido y confiable en el proceso de desarrollo y estándares. Los requerimientos no funcionales a menudo se aplican al sistema en su totalidad. Son importantes para que los usuarios puedan valorar las características no funcionales del producto.

Los requerimientos no funcionales, son un tipo de requisito que especifica criterios que pueden usarse para juzgar la operación de un sistema en lugar de sus comportamientos específicos. A continuación, se muestran los requisitos no funcionales definidos para el sistema y que utiliza el centro CIGED.

Tabla 2: Requisitos no funcionales

Nº del requisito	Nombre del requisito	Descripción
RNF1	**Confiabilidad**	1. La información almacenada debe ser confiable en cuanto a su veracidad e integridad desde su recopilación. 2. Verificación sobre acciones irreversibles (mensajes de confirmación).
RNF2	**Seguridad**	1. El usuario debe autenticarse antes de entrar al sistema 2. La información que se maneje en el sistema estará protegida de acceso no autorizado y divulgación, a diferentes permisos de los usuarios que empleen el sistema.

En este acápite se hará una breve reseña del diagrama del caso de uso del sistema haciendo énfasis en la descripción de los actores y el rol que desempeñan cada uno, para un mejor entendimiento del mismo y la descripción de los casos de uso.

2.4 Descripción de los actores

Tabla 3: Descripción de los actores del sistema

Actor	Objetivos
Usuario	El objetivo de este actor es acceder al sistema, recibir notificaciones, enviarlas, eliminarlas, confirmarlas como leídas y crear tipos de contenidos.
Sistema	El objetivo de este actor es emitir notificaciones.

2.4.1 Diagrama de caso de uso del sistema

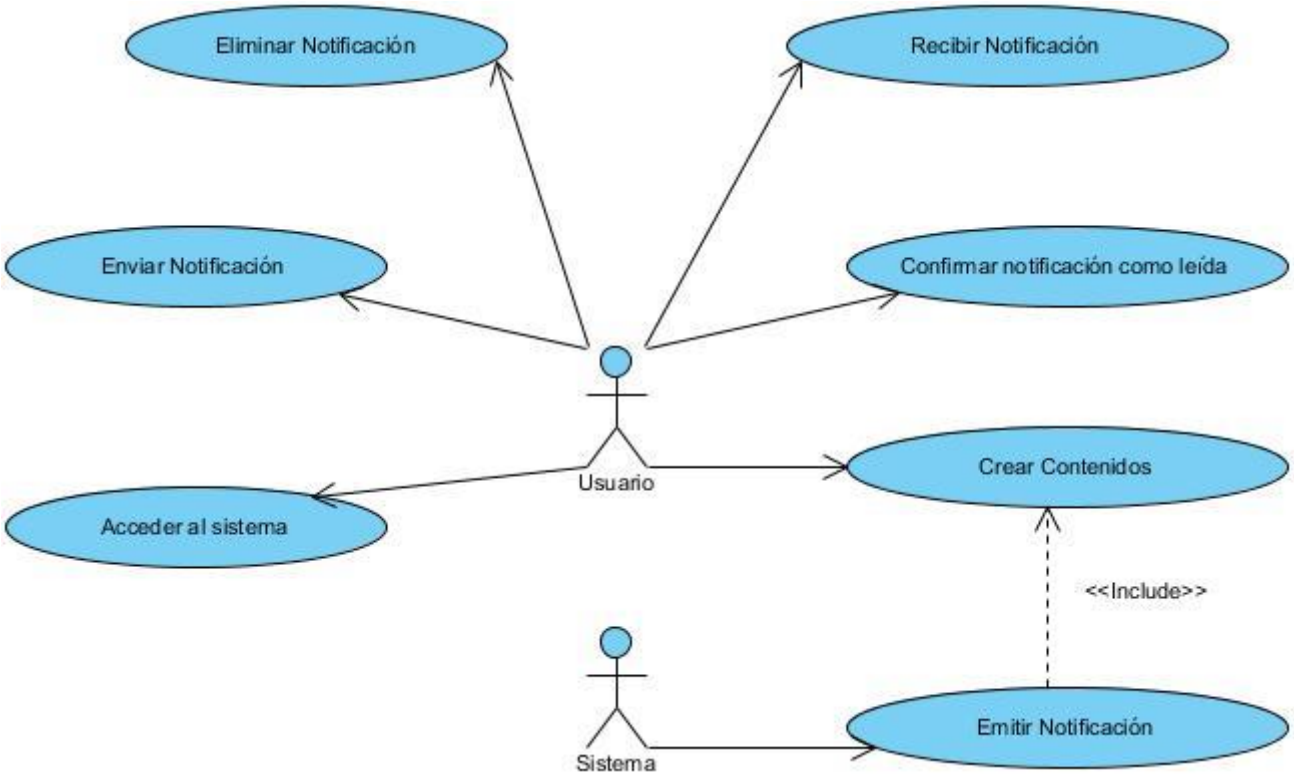


Figura 2: Diagrama de caso de uso del sistema

2.4.2 Descripción de los casos de uso

A continuación se mostrara la descripción de los casos de uso del sistema:

CU-1 Crear tipo de contenido

Tabla 4: Descripción del caso de uso Crear tipo de contenido

Objetivo	Crear tipo de contenido	
Actores	Usuario	
Resumen	El caso de uso comienza cuando el usuario crea el tipo de contenido que desee.	
Complejidad	Media	
Prioridad	Alta	
Precondiciones	Se tiene que haber autenticado previamente.	
Postcondiciones	El sistema notifica que se ha creado un tipo de contenido	
Flujo de eventos		
Flujo básico <Recibir notificación>		
	Actor	**Sistema**
1.	Indica que el usuario ha creado un tipo de contenido.	
1.		El sistema muestra todos los tipos de contenidos sobre los cuales el usuario

			desea recibir esa notificación.
2.			
Relaciones	**CU incluidos**		
	CU extendidos		
Prototipo elemental de interfaz gráfica de usuario			

CU-2 Acceder al Sistema

Tabla 5: Descripción del caso de uso Acceder al sistema

Objetivo	Acceder al sistema	
Actores	Usuario	
Resumen	El caso de uso comienza cuando el usuario inserta los datos de autenticación necesarios.	
Complejidad	Media	
Prioridad	Media	
Precondiciones	El dispositivo móvil del usuario se encuentra conectado a alguna red de datos, y existe un servidor de Alfresco accesible.	
Postcondiciones	El usuario puede acceder exitosamente al sistema.	
Flujo de eventos		
Flujo básico <Recibir notificación>		
	Actor	**Sistema**
2.	Introduce sus datos de autenticación	
3.		Envía los datos a un servidor de Alfresco que comprueba su autenticidad. Si los datos introducidos están correctos, permite visualizar los contenidos. En caso contrario emite un mensaje de error.
4.		
Relaciones	**CU incluidos**	
	CU extendidos	
Prototipo elemental de interfaz gráfica de usuario		

CU-3 Emitir Notificación

Tabla 6: Descripción del caso de uso Emitir Notificación

Objetivo	Emitir Notificación
Actores	Usuario
Resumen	El caso de uso comienza cuando existe alguna tarea para emitir la notificación.
Complejidad	Media

Prioridad	Alta		
Precondiciones	El usuario ha creado algún contenido		
Postcondiciones	Se agrupa la notificación por área.		
Flujo de eventos			
Flujo básico <Recibir notificación>			
	Actor		**Sistema**
3.	Indica que el usuario emitió una notificación		
5.			Muestra en el sistema cuando se emite una notificación
Relaciones	**CU incluidos**		
	CU extendidos		
Prototipo elemental de interfaz gráfica de usuario			

CU-4 Recibir Notificación

Tabla 7: Descripción del caso de uso Recibir Notificación

Objetivo	Recibir Notificación		
Actores	Usuario		
Resumen	El caso de uso comienza cuando el usuario emite una notificación		
Complejidad	Media		
Prioridad	Alta		
Precondiciones	El usuario ha emitido una notificación		
Postcondiciones	El jefe de proyecto confirma como leída la notificación		
Flujo de eventos			
Flujo básico <Recibir notificación>			
	Actor		**Sistema**
4.	Indica que el usuario recibió la notificación		
6.			Muestra al usuario en el móvil cuando recibe la notificación
Relaciones	**CU incluidos**		
	CU extendidos		
Prototipo elemental de interfaz gráfica de usuario			

CU-5 Eliminar Notificación

Tabla 8: Descripción del caso de uso Eliminar Notificación

Objetivo	Eliminar Notificación
Actores	Usuario
Resumen	El caso de uso comienza cuando el usuario está conectado al sistema a través de un servidor, una vez autenticado, procede a eliminar una

	notificación.		
Complejidad	Media		
Prioridad	Alta		
Precondiciones	El usuario está conectado al sistema		
Postcondiciones	El usuario confirma como leída la notificación		
Flujo de eventos			
Flujo básico <Recibir notificación>			
	Actor	**Sistema**	
5.	Indica que el usuario eliminó la notificación		
7.		Muestra al usuario en el dispositivo móvil cuando elimina la notificación	
Relaciones	**CU incluidos**		
	CU extendidos		
Prototipo elemental de interfaz gráfica de usuario			

CU-6 Confirmar notificación como leída

Tabla 9: Descripción del caso de uso Confirmar notificación como leída

Objetivo	Confirmar notificación como leída		
Actores	Usuario		
Resumen	El caso de uso comienza cuando el usuario recibe una notificación		
Complejidad	Media		
Prioridad	Media		
Precondiciones	El usuario ha recibido una notificación		
Postcondiciones	El usuario muestra que ha confirmado la notificación como leída		
Flujo de eventos			
Flujo básico <Confirmar notificación como leído>			
	Actor	**Sistema**	
6.	Indica que el usuario confirma que le llego la notificación		
8.		Muestra en el sistema cuando se confirma la notificación leída	
Relaciones	**CU incluidos**		
	CU extendidos		
Prototipo elemental de interfaz gráfica de usuario			

CU-7 Enviar notificación

Tabla 10: Descripción del caso de uso Enviar Notificación

Objetivo	Enviar Notificación	
Actores	Usuario	
Resumen	El caso de uso comienza cuando el usuario crea una notificación y procede a enviarla.	
Complejidad	Media	
Prioridad	Media	
Precondiciones	El usuario ha creado una notificación	
Postcondiciones	El usuario puede ver la notificación	
Flujo de eventos		
Flujo básico <Confirmar notificación como leído>		
	Actor	**Sistema**
7.	Indica que el usuario ha enviado una notificación	
9.		Muestra en el sistema cuando se envía una notificación
Relaciones	**CU incluidos**	
	CU extendidos	
Prototipo elemental de interfaz gráfica de usuario		

2.5 Interfaces del sistema

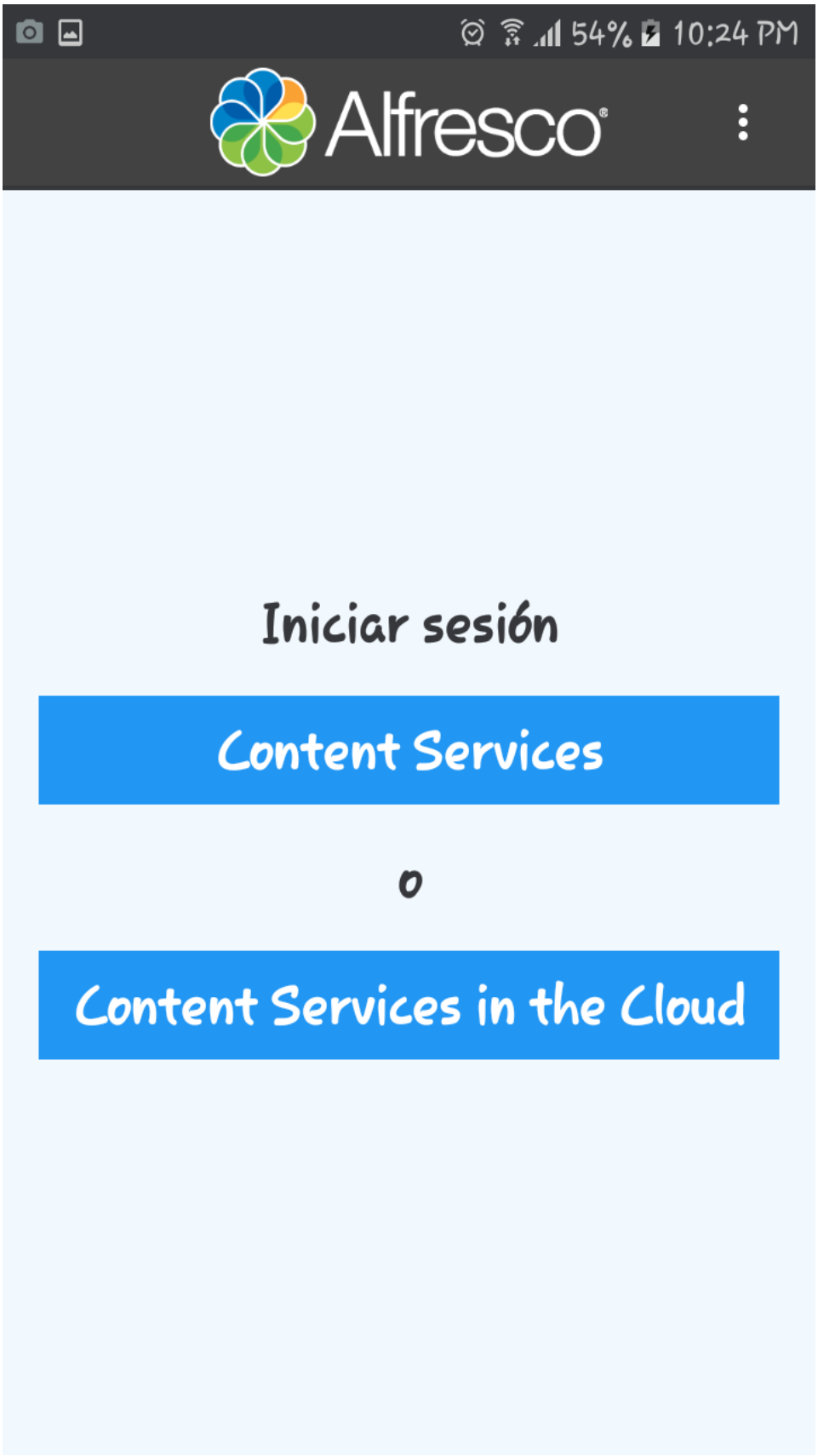


Figura 3: Imagen de interfaz de usuario para acceder al sistema

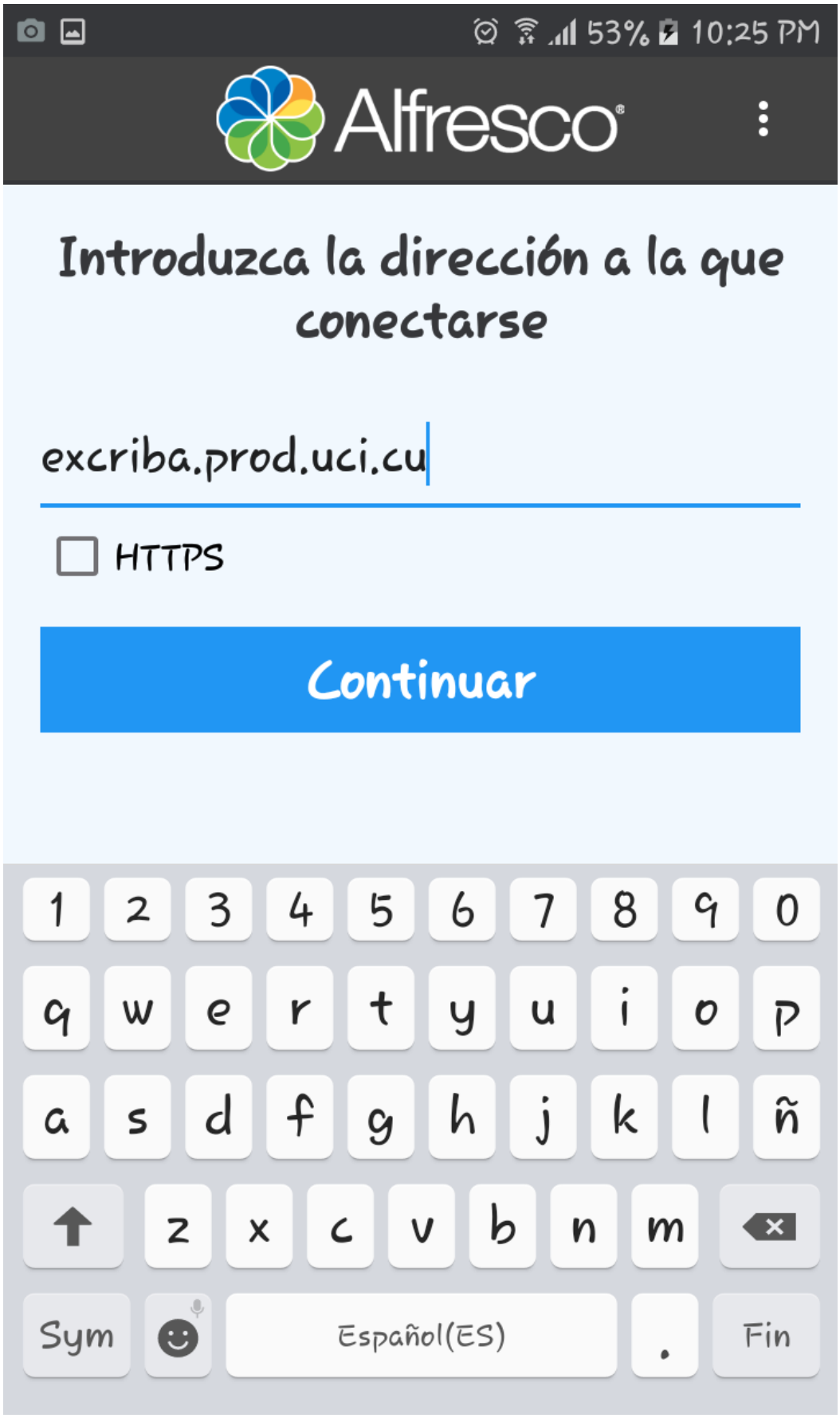


Figura 4: Imagen de interfaz de usuario para conectarse al servidor Tomcat de Alfresco

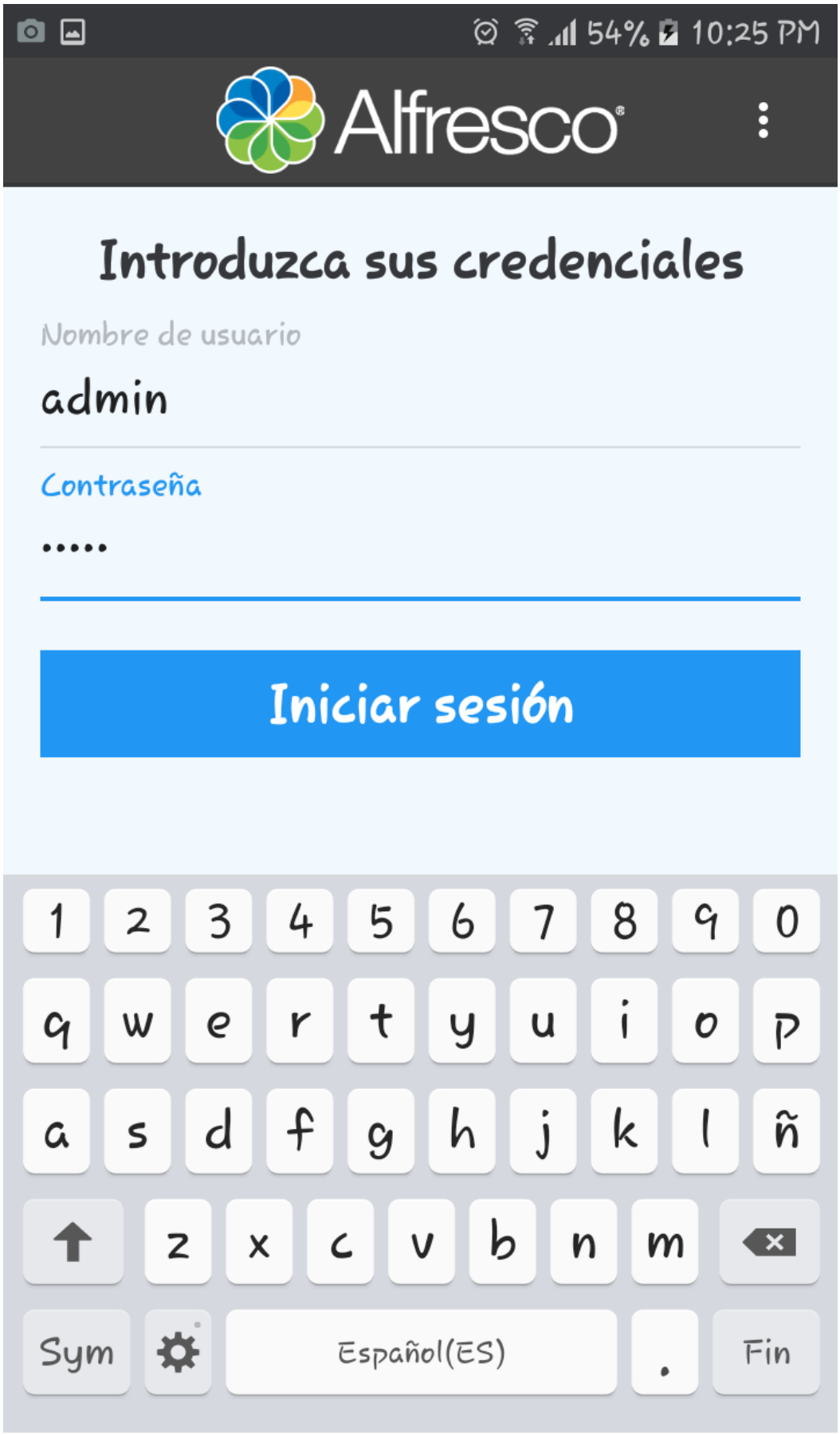


Figura 5: Imagen de interfaz de usuario para iniciar sesión

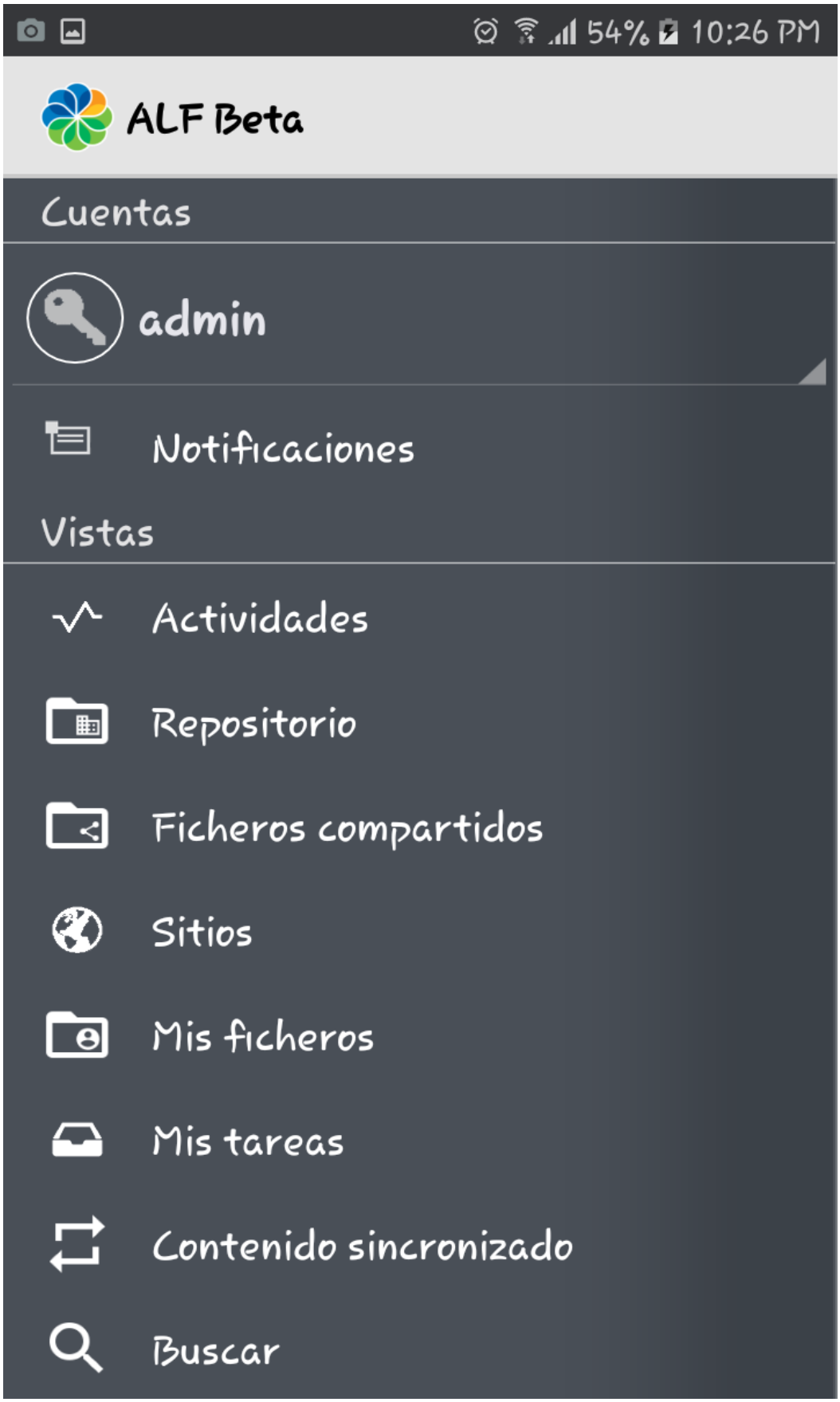


Figura 6: Imagen de interfaz de usuario de la aplicación

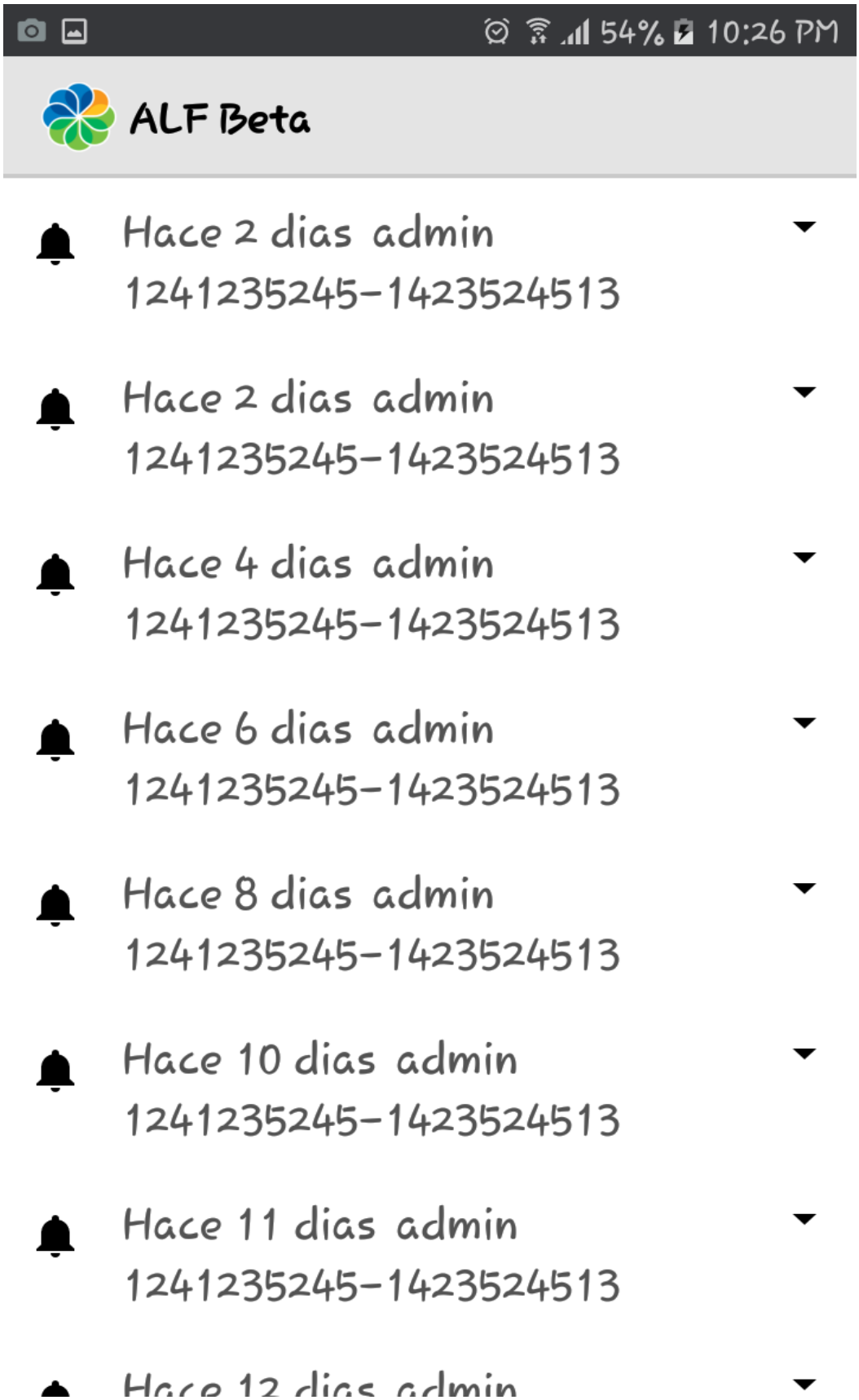


Figura 7: imagen de interfaz de usuario en el espacio de notificaciones

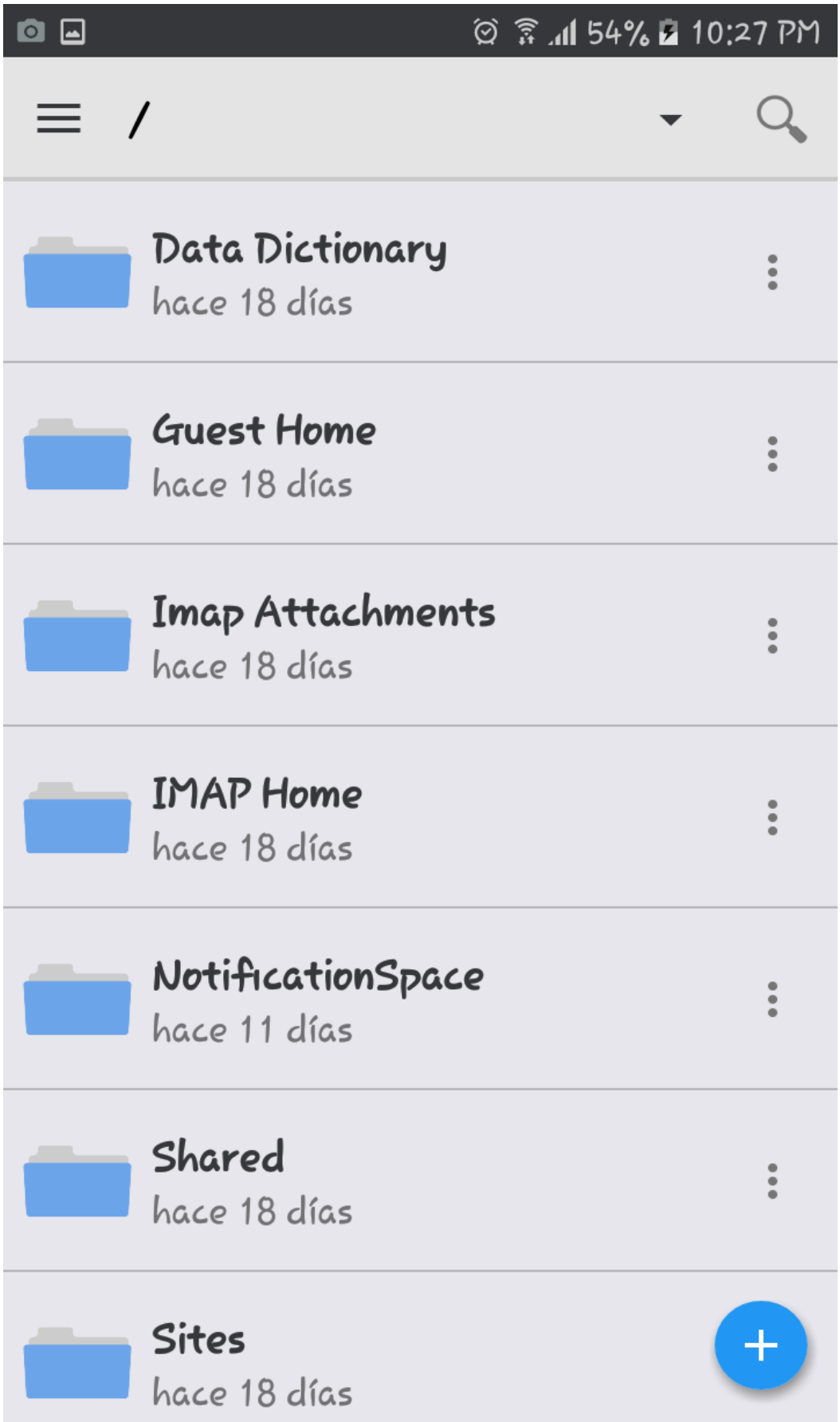


Figura 8: imagen de interfaz de usuario en el espacio del repositorio

2.6 Descripción de la arquitectura

Los patrones arquitectónicos, o patrones de arquitectura, también llamados arquetipos ofrecen soluciones a problemas de arquitectura de software para expresar una estructura de organización. Dan una descripción de los elementos y el tipo de relación que tienen junto con un conjunto de restricciones sobre cómo pueden ser usados (Ingenio DS, 2018). Según la Real Academia Española (RAE), un patrón arquitectónico expresa un esquema de organización estructural esencial para un sistema de software, que consta de subsistemas, sus responsabilidades e interrelaciones.

Para el desarrollo del componente de visualización de las notificaciones del sistema XABAL eXcriba se utiliza el patrón Modelo-Vista-Controlador (MVC). El cual es un paradigma que divide las partes que conforman una aplicación en el Modelo, las Vistas y los Controladores, permitiendo la implementación por separado de cada elemento, garantizando así la actualización y mantenimiento del software de forma sencilla y en un reducido espacio de tiempo.

Se especifica en la siguiente figura un ejemplo de MVC para dicho componente:

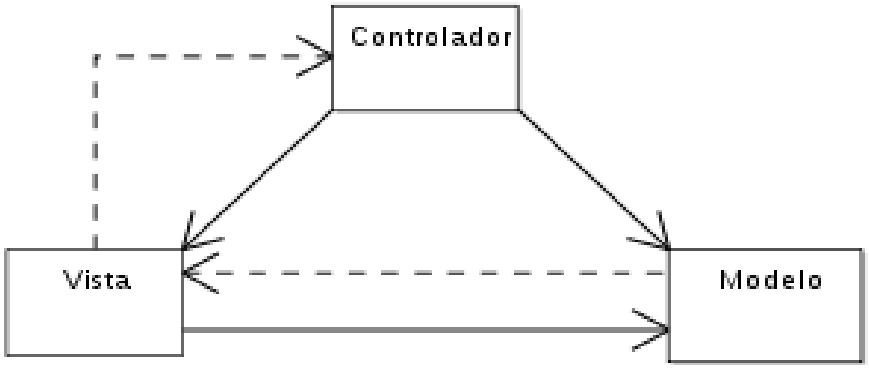


Figura 9: Modelo-Vista-Controlador (MVC)

Nota: las líneas sólidas indican una asociación directa, y las punteadas una indirecta.

En el **modelo** se manejan los datos del sistema y las operaciones asociadas a esos datos. La **vista** define y gestiona cómo se presentan los datos al usuario. El componente **controlador** (componentes del lado del servidor que manejan los requerimientos de HTTP) dirige la interacción del usuario y pasa estas interacciones a Vista y Modelo.

Tanto la vista como el controlador dependen del modelo, el cual no depende de las otras clases. Esta separación permite construir y probar el modelo independientemente de la representación visual.

Cuándo se usa: Se usa cuando existen múltiples formas de ver e interactuar con los datos. También se utiliza al desconocerse los requerimientos futuros para la interacción y presentación.

Ventajas: Permite que los datos cambien de manera independiente de su representación y viceversa. Soporta en diferentes formas la presentación de los mismos datos, y los cambios en una representación se muestran en todos ellos.

Adaptación al cambio. Los requerimientos de interfaz de usuario tienden a cambiar con mayor rapidez que las reglas de negocios. Dado que el modelo no depende de las vistas, agregar nuevas opciones de presentación generalmente no afecta al modelo.

Desventajas: Puede implicar código adicional y complejidad de código cuando el modelo de datos y las interacciones son simples.

2.6.1 Estilo de Arquitectura

La Transferencia de Estado Representacional (en inglés Representational State Transfer) o REST es un estilo de arquitectura software distribuidos como la World Wide Web. El término se originó en el año 2000, en una tesis doctoral sobre la web escrita por Roy Fielding, uno de los principales autores de la especificación del protocolo HTTP y ha pasado a ser ampliamente utilizado por la comunidad de desarrollo (Asier Marqués, 2018).
REST es cualquier interfaz entre sistemas que use HTTP para obtener datos o generar operaciones sobre esos datos en todos los formatos posibles, como XML y JSON. Es una alternativa en auge a otros protocolos estándar de intercambio de datos como SOAP (Simple Object Access Protocol), que disponen de una gran capacidad pero también mucha complejidad. A veces es preferible una solución más sencilla de manipulación de datos como REST**.**

Características de REST

- Protocolo cliente/servidor sin estado: cada petición HTTP contiene toda la información necesaria para ejecutarla, lo que permite que ni cliente ni servidor necesiten recordar ningún estado previo para satisfacerla.
- Las operaciones más importantes relacionadas con los datos en cualquier sistema REST y la especificación HTTP son cuatro: POST (crear), GET (leer y consultar), PUT (editar) y DELETE (eliminar).
- Los objetos en REST siempre se manipulan a partir de la URI**.** Es la URI y ningún otro elemento información para su modificación o borrado, o, por ejemplo, para compartir su ubicación exacta con terceros. El identificador único de cada recurso de ese sistema REST. La URI facilita acceder a la información para su modificación o borrado, o, por ejemplo, para compartir su ubicación exacta con terceros.
- **Interfaz uniforme**: para la transferencia de datos en un sistema REST, este aplica acciones concretas (POST, GET, PUT y DELETE) sobre los recursos, siempre y cuando estén identificados con una URI. Facilitando la existencia de una interfaz uniforme que sistematiza el proceso con la información.
- **Sistema de capas**: arquitectura jerárquica entre los componentes. Cada una de estas capas lleva a cabo una funcionalidad dentro del sistema REST.

La solución propuesta se desarrollará usando una arquitectura cliente – servidor.

2.7 Patrones de diseño

Un patrón de diseño es una estructura de diseño que resuelve un problema de diseño particular dentro un contexto específico y en medio de “fuerzas” que pueden tener un impacto en la manera en que se aplica y utiliza el patrón. Muchos patrones proporcionan guías sobre el modo en el que deberían asignarse las responsabilidades a los objetos. (Pressman, 2014).

2.7.1 Patrones GRASP

GRASP (General Responsibility Assignment Software Patterns). Son patrones generales de software para asignación de responsabilidades y tienen una serie de buenas prácticas de aplicación recomendable en el diseño de software. Es importante destacar que estos patrones están estrechamente relacionados entre sí, pues en la práctica, el nivel de acoplamiento no se puede considerar de manera aislada a otros principios como el Experto o Alta Cohesión. Estos patrones Grasp se dividen en 5 ellos son: Experto, Creador, Controlador, Alta cohesión y Bajo Acoplamiento (AdictosalTrabajo, 2018).

Experto: Asignar una responsabilidad al experto, clase que tiene la información para poder realizarla, y en información es el principio básico de asignación de responsabilidades.

El patrón experto posibilita que se mantenga el encapsulamiento de la información, puesto que los objetos utilizan su propia información para llevar a cabo las tareas. Normalmente, esto conlleva un bajo acoplamiento, lo que da lugar a sistemas más robustos y más fáciles de mantener, además se distribuye el comportamiento entre las clases que contienen la información requerida, por tanto, se estimula las definiciones de clases más cohesivas y ligeras que son más fáciles de comprender y manejar. Se soporta normalmente una alta cohesión. Este patrón se pone de manifiesto en la funcionalidad crear tipo contenido.

Controlador: Asignar la responsabilidad de controlar el flujo de eventos del sistema, a clases específicas. Esto facilita la centralización de actividades (validaciones, seguridad, etc.). El controlador no realiza estas actividades, las delega en otras clases con las que mantiene un modelo de alta cohesión. Un error muy común es asignarle demasiada responsabilidad y alto nivel de acoplamiento con el resto de los componentes del sistema.

Se recomienda dividir los eventos del sistema en el mayor número de controladores para poder aumentar la cohesión y disminuir el acoplamiento. Se manifiesta en el fichero NotificationRepository que se encarga de controlar todas las notificaciones que se realizan en el sistema.

Alta cohesión: La información que almacena una clase debe ser coherente y debe estar relacionada con la clase. Cada elemento de nuestro diseño debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos y auto-identificable.

Su función es asignar una responsabilidad de manera que la cohesión permanezca alta. Un elemento con responsabilidades altamente relacionadas, y que no hace una gran cantidad de trabajo, tiene alta cohesión. Estos elementos pueden ser clases, subsistemas, etcétera.

Posibilitando que se incremente la claridad y facilita la comprensión del diseño, simplifica el mantenimiento y las mejoras, soportando a menudo bajo acoplamiento y el grado bajo de funcionalidades altamente relacionadas incrementa la reutilización porque una clase cohesiva se puede utilizar para un propósito muy específico. Este patrón se evidencia en la creación de las notificaciones.

Bajo acoplamiento: Debe haber pocas dependencias entre las clases. En caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases.

Este patrón permite que en caso de modificarse algún elemento sus cambios no afecten a otro componente, además son fáciles de entender de manera aislada y conveniente para la reutilización. La utilización de este patrón se evidencia cuando se envía una notificación a un usuario determinado, mediante la funcionalidad enviar notificación, donde se hace uso de diferentes servicios.

2.7.2 Los Patrones GOF

Gang of Four –GOF: describen soluciones simples y elegantes a problemas específicos en el diseño de software orientado a objetos" (Gamma et al, 1994)
Eric Braude define los patrones de diseño como "combinaciones de componentes, casi siempre clases y objetos que por experiencia se sabe que resuelven ciertos problemas de diseño comunes" (Braude, 2018).
Fue por los años 1994, que apareció el libro “Design Patterns: Elements of Reusable Object Oriented Sofware” escrito por los ahora famosos Gang of Four (GoF, que en español es la pandilla de los cuatro) formada por Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides. Ellos recopilaron y documentaron 23 patrones de diseño aplicados usualmente por expertos diseñadores de software orientado a objetos. Desde luego que ellos no son los inventores ni los únicos involucrados, pero ese fue luego de la publicación de ese libro que empezó a difundirse con más fuerza la idea de patrones de diseño. Los patrones de diseño el grupo de GoF clasifican en 3 grandes categorías basadas en su propósito: creacionales, estructurales y de comportamiento (Geek the Planet, 2018).

Patrones de creación

Se encargan de la creación de instancias de los objetos. Abstraen la forma en que se crean los objetos, permitiendo tratar las clases a crear de forma genérica, dejando para después la decisión de que clase crear o cómo crearla. Según donde se tome dicha decisión se pueden clasificar los patrones de creación

en: patrones de creación de clases (la decisión se toma en los constructores de las clases y usan la herencia para determinar la creación de las instancias) (Slideshare, 2018). Los patrones de creación son:

- Builder (Constructor virtual): Abstrae el proceso de creación de un objeto complejo, centralizando dicho proceso en un único punto.
- Prototype (Prototipo): Crea nuevos objetos clonándolos de una instancia ya existente.
- Singleton (Instancia única): Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia.

Estos patrones de diseño proporcionan una forma de crear objetos al tiempo que ocultan la lógica de creación, en lugar de crear instancias de objetos directamente utilizando un nuevo operador. Esto le da al programa más flexibilidad para decidir qué objetos necesitan crearse para un caso de uso dado.
El patrón creador que se utilizó en el desarrollo de la aplicación fue:
Singleton: Restringe el número total de instancias de una clase en particular a solo una a la vez. Esto se usa comúnmente cuando se requiere acceso global al objeto en todo el sistema, y cualquier cambio o consulta al objeto debe ser consistente e idéntico.

ClaseServiceGenerator.java
Encargada de crear el retrofit para consumir los servicios en vez de instanciar un nuevo retrofit cada vez que se requiera utilizarlo.

Patrones Estructurales

Son los que plantean las relaciones entre clases, las combinan y forman estructuras mayores. Tratan de conseguir que los cambios en los requisitos de la aplicación no ocasionen cambios en las relaciones entre los objetos. Lo fundamental son las relaciones de uso entre los objetos, y éstas están determinadas por las interfaces que soportan los objetos. Estudian cómo se relacionan los objetos en tiempo de ejecución. Sirven para diseñar las interconexiones entre los objetos (Slideshare, 2018). Los patrones estructurales son:

- Adapter (Adaptador): Permite que las clases trabajen juntas con interfaces incompatibles.
- Bridge (Puente): Desacopla una abstracción de su implementación.
- Composite (Objeto compuesto): Permite tratar objetos compuestos como si de uno simple se tratase.
- Facade (Fachada): Provee de una interfaz unificada simple para acceder a una interfaz o grupo de interfaces de un subsistema.
- Proxy: Mantiene un representante de un objeto.

Estos patrones de diseño se refieren a la composición de clases y objetos. El concepto de herencia se usa para componer interfaces y definir formas de componer objetos para obtener nuevas funcionalidades.

El patrón estructural que se utilizó en el desarrollo de la aplicación fue:

Adaptador: permite que una interfaz, que de lo contrario es incompatible, se adapte para adaptarse a una nueva clase. Normalmente, esto se realiza creando una nueva clase ClassNameAdapter que implementa la interfaz, lo que permite la compatibilidad en todo el sistema.
Clase *NotificationAdapater.java*
Esta clase hereda de PageDListAdapter obteniendo métodos para la ubicación como es el que compara los elementos de la lista y sus atributos para rellenarlos.

Patrones de Comportamiento

Plantea la interacción y cooperación entre las clases. Los patrones de comportamiento estudian las relaciones entre llamadas entre los diferentes objetos, normalmente ligados con la dimensión temporal (Geek the Planet, 2018). Los patrones de comportamiento son:

- Command (Orden): Encapsula una operación en un objeto, permitiendo ejecutar dicha operación sin necesidad de conocer el contenido de la misma.
- Iterator (Iterador): Permite realizar recorridos sobre objetos compuestos independientemente de la implementación de estos.
- Observer (Observador): Define una dependencia de uno-a-muchos entre objetos, de forma que cuando un objeto cambie de estado se notifique y actualicen automáticamente todos los objetos que dependen de él. El patrón Observer se utilizó durante el desarrollo de las diferentes vistas del sistema como se muestra en el siguiente diagrama.
- Strategy (Estrategia): Permite disponer de varios métodos para resolver un problema y elegir cuál utilizar en tiempo de ejecución.

Estos patrones de diseño están específicamente relacionados con la comunicación entre objetos.

El patrón comportamiento que se utilizó en el desarrollo de la aplicación fue:

Observer: crea una dependencia basada en eventos entre los objetos, de modo que una actualización del objeto observado causa que se notifiquen los objetos del observador. Normalmente, esto se encuentra en muchos idiomas que utilizan la funcionalidad asíncrona, que requiere que los eventos se observen y se respondan fuera del orden de ejecución típico.
Clases *MainActivity.java* y *NotificationsViewModel.java*

La clase *MainActivity.java* observa las notificaciones en el ViewModel (*NotificationViewModel.java*) a la espera de algún cambio en los elementos (inserción, modificación o que se elimina de los elementos)

2.8 Modelo de diseño

El modelo de diseño es una abstracción de la implementación del sistema. Se utiliza para concebir y para documentar el diseño del sistema de software. Es un producto de trabajo integral y compuesto que abarca todas las clases de diseño, subsistemas, paquetes, colaboraciones y las relaciones entre ellos, una representación significativa de ingeniería de algo que se va a construir y se centra en cuatro áreas importantes (PRESSMAN S. , 2013).
-Datos
-Arquitectura
-Interfaces
-Componentes Diseño de interfaces

Las cuales proporcionan las representaciones del software que se pueden evaluar en cuanto a calidad.

2.9 Conclusiones del capitulo

En el capítulo se describió la propuesta de solución. Se definieron los artefactos correspondientes a las etapas de análisis y diseño del módulo de notificaciones del sistema XABAL eXcriba, según la metodología AUP-UCI. Se identificaron y describieron los requisitos funcionales y no funcionales del sistema. Se definió como arquitectura de software a emplear la Arquitectura MVC, logrando así una aplicación mucho más organizada y menos compleja a la hora de realizar alguna modificación. Se identificaron además, los patrones de diseño GRASP y GoF a utilizar en el desarrollo de la solución y se diseñó la base de datos.

Capítulo 3: Implementación y pruebas de la solución propuesta

A partir de los resultados obtenidos de las etapas de análisis y diseño, a continuación se describen los elementos requeridos para desarrollar el componente del sistema eXcriba. Se muestran el diagrama de componentes, diagrama de despliegue, diagrama de paquetes y se definen los estándares de codificación utilizados para el desarrollo de la aplicación. Se precisan las pruebas de software a aplicar y se muestran los resultados obtenidos.

3.1 Modelo de implementación

El modelo de implementación representa cómo los elementos del modelo de diseño se implementan en términos de componentes. De igual forma describe cómo se organizan los componentes de acuerdo con los mecanismos de estructuración disponibles en el entorno de implementación y la relación existente entre ellos.

3.1.1 Diagrama de despliegue

Un Diagrama de Despliegue modela la arquitectura en tiempo de ejecución de un sistema. Mostrándose la configuración de los elementos de hardware (nodos) y cómo estos y los artefactos del software se trazan en esos nodos.

A continuación se muestra el diagrama de despliegue del componente de visualización de las notificaciones:

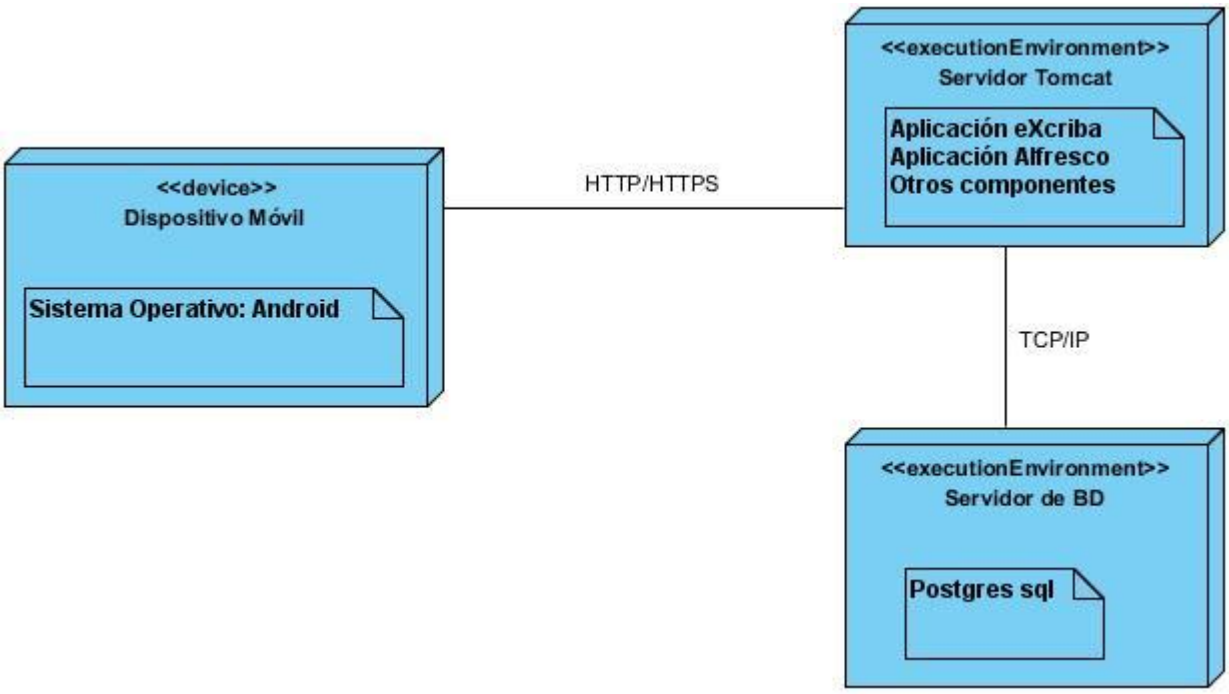


Figura 10: Diagrama de despliegue del sistema de notificaciones

Dispositivo Móvil: representa el hardware donde se ejecutará el sistema. La base de datos que utiliza el mismo es interna y estará representada por el fichero tableModúlo Notificaciones.db, el cual almacenará toda la información con que trabaja la aplicación. El sistema operativo del dispositivo debe ser Android. Contando con los requisitos no funcionales especificados en el acápite 2.4.

Servidor Tomcat: representa el servidor al cual el usuario estará conectado para interactuar a través del dispositivo móvil mediante un API REST de Alfresco y Alfresco Mobile.

Servidor de BD: representa el servidor de base de dato con el cual se va a interactuar para realizar alguna consulta, ya sea eliminar una notificación o verificar si existe alguna almacenada en la base de dato.

3.1.2 Diagrama de componentes

En el diagrama de componentes, se representa un componente como un módulo físico de código o paquete, puede ser relacionado en un diagrama de componentes, el cual muestra varios componentes de un sistema, describiendo sus elementos físicos y sus dependencias.

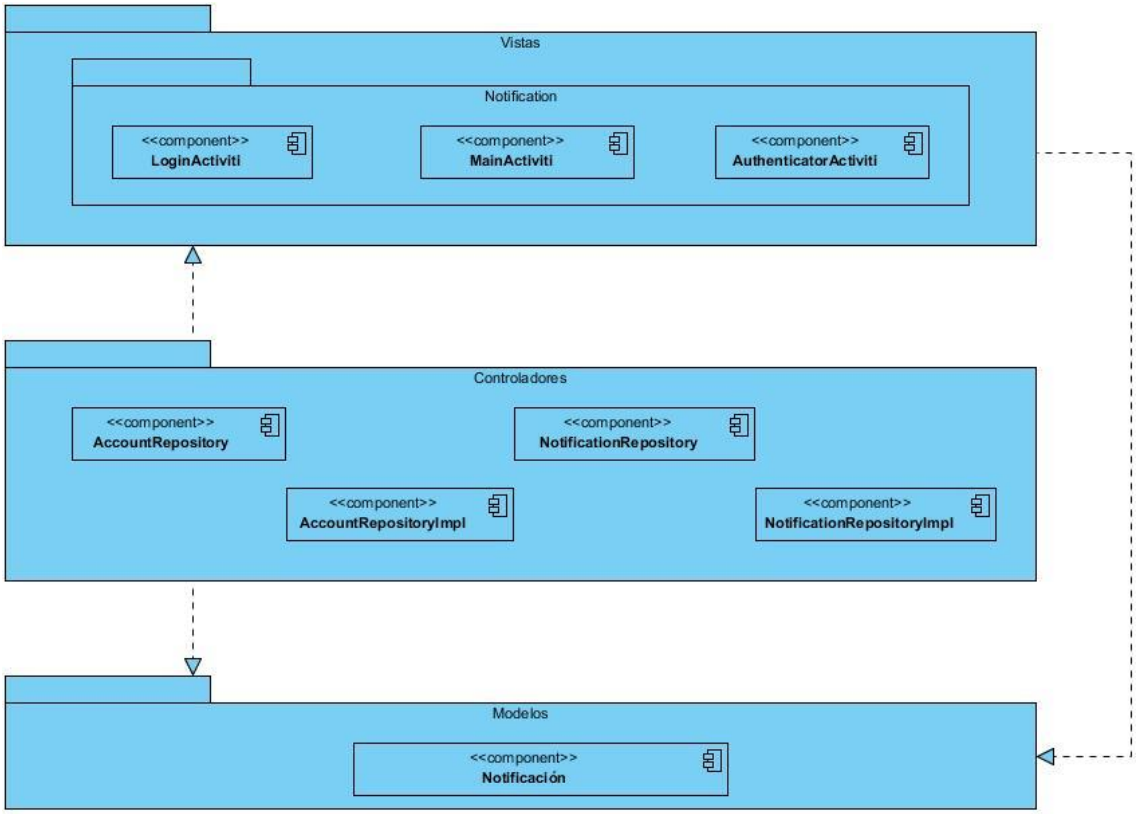


Figura 11: Diagrama de componentes de la aplicación

3.1.3 Diagrama de paquetes

Presentan cómo se organizan los elementos de modelado en paquetes y las dependencias entre ellos, incluyendo importaciones y fusiones de paquetes.

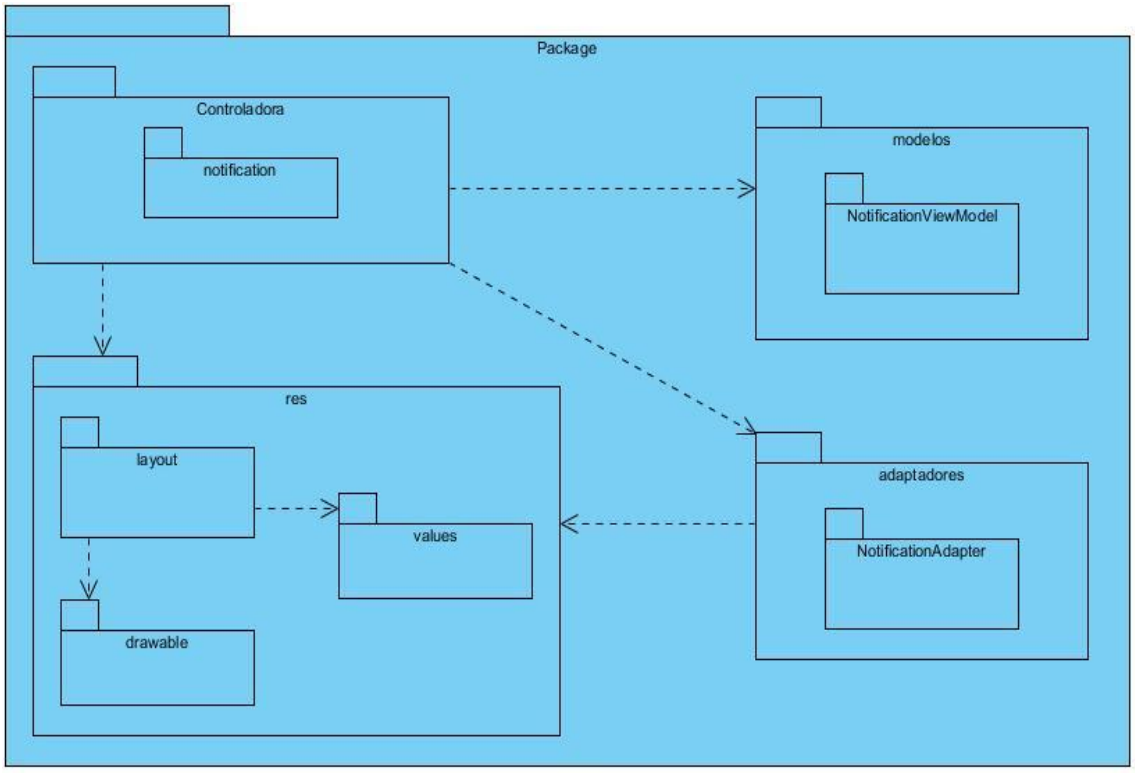


Figura 12: Diagrama de paquetes del modelo de notificaciones

3.2 Estándares de codificación

Los estándares de codificación se definen con vistas a mejorar el entendimiento del código perteneciente al módulo de notificaciones del sistema eXcriba por otros desarrolladores y alcanzar una uniformidad en el mismo. A continuación se listan los elementos pertenecientes al estándar de codificación definido para el desarrollo del módulo de notificaciones del sistema eXcriba:

3.2.1 JavaScript

Declaración de variables

- A la hora de codificar es muy importantes, por eso es necesario tener en cuenta las siguientes premisas:
  - ✓ **Nombre de variables**
- El nombre de todas las variables y métodos comenzarán con letra minúscula y si está compuesto por varias palabras se empleara la notación camelCase.
- Los nombres de las variables deben ser cortos y significativos.

- El nombre de las clases comenzará con mayúsculas y cada salto de palabras debe iniciar con mayúsculas.

Valores de variables

- Dejar espacios entre el nombre y el valor de una variable son muy importantes, puesto que mejoran la legibilidad del código.
- En el contenido siempre se indentará con tabulaciones, nunca utilizando espacios en blanco.

Nombres de funciones

- Los nombres de funciones deben ser escritos con notación camelCase.
  - ✓ **Tipos de funciones**
- Cuando una expresión exceda la línea, se romperá después de una coma o antes de un operador.
- Cada método o función debe estar precedido por un comentario o encabezado con lenguaje descriptivo y claro, donde se enuncie de manera resumida el objetivo y las variables usadas como argumento, a continuación se muestra un ejemplo:

```
/**

*
* @param {*} myUser Alfreso JScript UserNode Object
* @param {*} nodeGroup Alfreso JScript GroupNode Object
* @param {*} rootFolder Root Folder for Documents
*
*/
function getGroup (myUser, nodeGroup, rootFolder) {
}
```

- Los ficheros deben ser nombrados en minúsculas utilizando la forma ‘underscore’.

3.2.2 FreeMaker

Nombres de variables

- Los nombres que se usan deben ser significativos.
- Los nombres deben estar en minúsculas, excepto la primera letra de cada palabra a partir de la segunda.
  - ✓ **Nombres de registros**
- Cuando se lee un registro de una tabla, el nombre del registro, debe empezar por $row y luego tener el nombre de la tabla.

Nombres de programas

- Todo en minúscula excepto la primera letra de cada palabra a partir de la segunda.

✓ **Constantes**

- Se deben evitar constantes numéricas sin mucho significado. Para eso es conveniente definir las constantes en el programa.
- Todos los caracteres deben estar en mayúsculas y las palabras separadas por "_".

Variables globales

- Se debe evitar el uso de variables globales pues pueden ser modificadas erróneamente y pueden causar errores muy difíciles de identificar. Si se usan, para poder identificarlas, deben estar en mayúsculas.

✓ **Corchetes e indentación**

- La indentación es algo que ayuda a darle claridad a un programa y es indispensable que se haga bien. Debe hacerse con "tabs" y no con espacios en blanco.
- Los corchetes de un bloque if, o switch, o for, deben ir en la misma línea de la cláusula.

Claridad de los programas

- Es importante que los programas y rutinas que se escriban sean claros y fáciles de entender.
- Además de brindarse la explicación de qué hace cada programa o función al principio hay que tratar de que las funciones quepan en una sola página y que antes de cada sección se explique qué es o qué se está haciendo (sobre todo, cuando se usan "truquitos ").
- Hay que evitar el uso del if corto, al ser difícil de entender. La única excepción sería cuando se están utilizando campos calculados y se quiere la instrucción en una sola línea.

3.2.3 Java 9

Nombres de ficheros

- Esta sección lista las extensiones más comunes usadas y los nombres de ficheros.

✓ **Extensiones de los ficheros**

- El software lista las extensiones más comunes usadas y los nombres de ficheros.

Tabla 11: Lista de extensiones por tipo de ficheros

Tipo de fichero	**Extensión**
Fuente Java	.java
Bytecode de java	.class

Nombres de ficheros comunes

- Los nombres de ficheros más utilizados incluyen:

Tabla 12: Ficheros más usados

Nombre de fichero	Uso
GNUmakefile	El nombre preferido para ficheros "make". Usamos “gnumake” para construir nuestro software.
README	El nombre preferido para el fichero que resume los contenidos de un directorio particular

Organización de los ficheros

- Un fichero consiste de secciones que deben estar separadas por líneas en blanco y comentarios opcionales que identifican cada sección.
- Los ficheros de más de 2000 líneas son incómodos y deben ser evitados.

Ficheros fuente Java

- Cada fichero fuente Java contiene una única clase o interface pública. Cuando algunas clases o interfaces privadas están asociadas a una clase pública, pueden ponerse en el mismo fichero que la clase pública. La clase o interfaz pública debe ser la primera clase o interface del fichero.
- Los ficheros fuentes Java tienen la siguiente ordenación:
  - Comentarios de comienzo.
  - Sentencias package e import.
  - Declaraciones de clases e interfaces.

Indentación

- Se deben emplear cuatro espacios como unidad de indentación. La construcción exacta de la indentación (espacios en blanco contra tabuladores) no se especifica. Los tabuladores deben ser exactamente cada 8 espacios.

Longitud de la línea

- Evitar las líneas de más de 80 caracteres, al no ser manejadas bien por muchas terminales y herramientas.
  - ✓ **Rompiendo líneas**
- Cuando una expresión no se introduzca en una línea, romperla de acuerdo con estos principios:
  - Romper después de una coma.
  - Romper antes de un operador.
  - Preferir roturas de alto nivel (más a la derecha que el "padre") que de bajo nivel (más a la izquierda que el "padre").
  - Alinear la nueva línea con el comienzo de la expresión al mismo nivel de la línea
  - Si las reglas anteriores llevan a código confuso o a código que se aglutina en el margen derecho, indentar justo 8 espacios en su lugar.

3.3 Verificación funcional

La calidad del software debe implementarse a lo largo de todo el ciclo de vida de un software, debe correr paralela desde la planificación del producto hasta la fase de producción de este como actividad de protección. El aspecto a considerar en el Control de la Calidad del Software es la “Prueba de Software”.

3.3.1 Pruebas de software

Las pruebas de software son una actividad primordial para desarrollar un software de calidad. Aseguran el correcto cumplimiento de la funcionalidad del producto, ayudan a brindar confianza, confirman la fiabilidad del uso y previenen defectos en producción. Además consisten en la dinámica de la verificación del comportamiento de un programa en un conjunto finito de casos de prueba, debidamente seleccionados, por lo general infinitas ejecuciones de dominio, contra la del comportamiento esperado. Son una serie de actividades que se realizan con el propósito de encontrar los posibles fallos de implementación, calidad o usabilidad de un programa; probando el comportamiento del mismo (EcuRed, 2018).

Elementos a tener en cuenta para que una prueba tenga éxito: (Lara, 2015).

- ✓ Estrategia de prueba
- ✓ Niveles de prueba
- ✓ Tipo de prueba
- ✓ Método de prueba
- ✓ Caso de prueba

3.3.2 Estrategia de pruebas

Una estrategia de prueba de software proporciona una guía que describe los pasos que deben realizarse como parte de la prueba, cuando se planean y llevan a cabo dichos pasos, y cuánto esfuerzo, tiempo y recurso requerirán. Por tanto, cualquier estrategia de prueba debe incorporar la planificación, diseño, ejecución, recolección y evaluación de los casos de prueba (PRESSMAN S. , 2013).

Para realizar una correcta estrategia de pruebas se deben tener en cuenta algunos criterios de importancia (RUIZ, 2015.).

- ✓ Describe el enfoque y los objetivos generales de las actividades de prueba.
- ✓ Incluye los niveles de prueba a ser diseccionados, el tipo de prueba a ser ejecutada y los casos de prueba diseñados para lograr los objetivos.
- ✓ Define:
    - Técnicas de pruebas (manual o automática) y herramientas a ser usadas.
    - Criterios de éxitos y culminación de las pruebas.
    - Consideraciones especiales relacionadas con los recursos necesarios para realizar las pruebas.

3.3.3 Niveles de prueba

Las pruebas son realizadas en diferentes niveles de esfuerzo. Estos niveles se distinguen en general por el rol de quien las ejecuta y las técnicas utilizadas.

Niveles de prueba: La prueba es la aplicada para diferentes tipos de objetivos, en diferentes escenarios o niveles de trabajo. Se muestran los siguientes niveles de pruebas.

1. Pruebas unitarias
2. Pruebas de integración
3. Pruebas de sistema
4. Pruebas de aceptación
5. Prueba de desarrollador
6. Prueba independiente
7. Prueba de regresión

Para garantizar la calidad del producto que da respuesta a la solución propuesta, se le decidió realizar pruebas en uno de los niveles anteriormente definidos, en el de pruebas del sistema haciendo uso del método de caja negra, mediante la técnica de partición equivalente. Se enfocan en los requerimientos funcionales del software, es decir, las técnicas de prueba de caja negra le permiten derivar conjuntos de condiciones de entrada que revisarán por completo todos los requerimientos funcionales para un programa.

Es válido aclarar que La metodología AUP-UCI propone 3 disciplinas de pruebas para aplicarle a los proyectos: pruebas internas, pruebas de liberación y pruebas de aceptación.

1- Pruebas internas: Se realizan en el centro y se verifica el resultado de la implementación probando cada construcción. Se desarrollan los casos de pruebas. Se realizaron pruebas de caja negra que permiten obtener conjuntos de condiciones de entrada que ejerciten todos los requisitos funcionales para encontrar varios tipos de errores (interfaz, rendimiento, estructura de datos).Se utilizó la técnica de la partición de equivalencia que permite examinar los valores válidos y no válidos de las entradas existentes en el software.

2- Pruebas de liberación: Pruebas diseñadas y ejecutadas por una entidad certificadora externa antes de ser entregados al cliente para su aceptación. Se realizaron pruebas de caja negra que verifican que el sistema funcione apropiadamente y sin errores.

3- Pruebas de aceptación: Se verifica que el software está listo y que puede ser usado por usuarios finales. Se hizo entrega del módulo al cliente, y este realizo un acta de aceptación planteando haberse cumplido el objetivo del mismo.

3.3.4 Métodos de prueba

Los métodos de prueba definen la estrategia a seguir en función de la verificación y validación del sistema diseñado para descubrir fallos. El método estudiado fue la prueba de caja negra.

Pruebas de caja negra

Las pruebas de caja negra también denominadas pruebas de comportamiento, se centran en los requisitos funcionales del software. O sea, la prueba de caja negra permite obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa Esta prueba intenta encontrar errores de las siguientes categorías: funciones incorrectas o ausentes, errores de interfaz, errores en estructura de datos o en accesos a bases de datos externas, errores de rendimiento y errores de inicialización y terminación (PRESSMAN R. , 2014).

3.3.5 Pruebas de sistema

Las pruebas de sistema se llevan a cabo cuando todo el desarrollo ha sido culminado y se tiene una versión preliminar del sistema que saldrá a producción. Esta etapa de pruebas consiste en probar un sistema integrado con el objeto de comprobar el cumplimiento de requisitos especificados. Las pruebas de sistema se desarrollan utilizando casos de prueba funcionales y no funcionales.

Casos de prueba

Un caso de prueba es una especificación de un caso para probar el sistema, incluyendo qué probar, con qué entradas y resultados y bajo qué condiciones. Su principal objetivo es obtener un conjunto de pruebas que tengan una mayor probabilidad de descubrir los defectos del software. Se utilizó el método de caja negra aplicando la técnica partición de equivalencia.

- ✓ **Partición por equivalencia:**
  - Permite examinar los valores validos e inválidos de las entradas existentes en el software.
  - Descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico.

A continuación se pueden apreciar algunos de los casos de prueba que fueron diseñados para realizar estas pruebas de caja negra al sistema, el resto puede ser consultado en el Anexo.

Tabla 13: Caso de prueba para acceder al sistema

Escenario	Descripción	Nombre	Etiqueta	Comentario	Respuesta del sistema	Flujo central
Loguearse al sistema	Permite a un usuario acceder al sistema desde la plataforma Android.	V	V	V	Envía los datos a un servidor de Alfresco que comprueba su autenticidad. Si los datos introducidos están correctos, permite visualizar los contenidos. En caso contrario emite un mensaje de error.	1-Completar el campo obligatorio Usuario. 2-Completar el campo contraseña. 3-Completar el campo Url. 4-Presionar el botón acceder.
		Usuario	Opcional	Opcional		

Tabla 14: Caso de prueba para eliminar una notificación

Escenario	Descripción	Respuesta del sistema	Flujo central
Eliminar notificación	Permite al usuario eliminar una notificación	El sistema muestra cuando ha sido eliminada una notificación	1-El usuario elimina la notificación.

Tabla 15: Caso de prueba para confirmar una notificación como leída

Escenario	Descripción	Respuesta del sistema	Flujo central
Confirmar notificación como leída	Permite que el usuario confirme que haya leído las notificaciones.	Muestra en el sistema cuando se confirma la notificación leída	1-El usuario lee la notificación.

Tabla 16: Tabla de caso de prueba para recibir una notificación

Escenario	Descripción	Respuesta del sistema	Flujo central
Recibir notificación	Permite que el usuario haya recibido la notificación.	Muestra en el sistema cuando el usuario haya recibido una notificación.	1-El usuario recibe la notificación.

Tabla 17: Caso de prueba para enviar una notificación

Escenario	Descripción	Respuesta del sistema	Flujo central
Enviar notificación	Permite que el usuario haya enviado la notificación.	Muestra en el sistema cuando el usuario haya enviado una notificación.	1-El usuario envía la notificación.

Tabla 18: Caso de prueba para crear tipo de contenido

Escenario	Descripción	Nombre	Tipo	Visibilidad	Descripción	Respuesta del sistema	Flujo central
Crear tipo de contenido	Permite al usuario del sistema, definir los contenidos sobre los cuales se desea recibir notificaciones	V	V	V	V	El sistema muestra todos los tipos de contenidos sobre los cuales el usuario desea recibir esa notificación	1-Definir los formatos de dichos tipos de contenidos. 2-Indicar cual tipo de contenido desea mostrar. 3-Mostrar el contenido deseado.
		Usuario	Espacio de trabajo	Visible	Opcional		

Tabla 19: Caso de prueba para emitir una notificación

Escenario	Descripción	Respuesta del sistema	Flujo central
Emitir notificación en eXcriba	Permite emitir notificaciones sobre los tipos de contenidos definidos por el administrador del sistema a todos, o un conjunto de los usuarios del sistema.	Muestra en el sistema cuando se emite una notificación.	1-Dar clic en el botón crear. 2-Rellenar los metadatos en el formulario del eXcriba. 3-Presionar el botón aceptar.

Se realizaron un total de dos iteraciones de pruebas encontrándose un total de 6 No Conformidades, de ellas dos fueron errores ortográficos y 4 fueron errores del código fuente corrigiéndose los mismos en el momento en que ocurrieron.

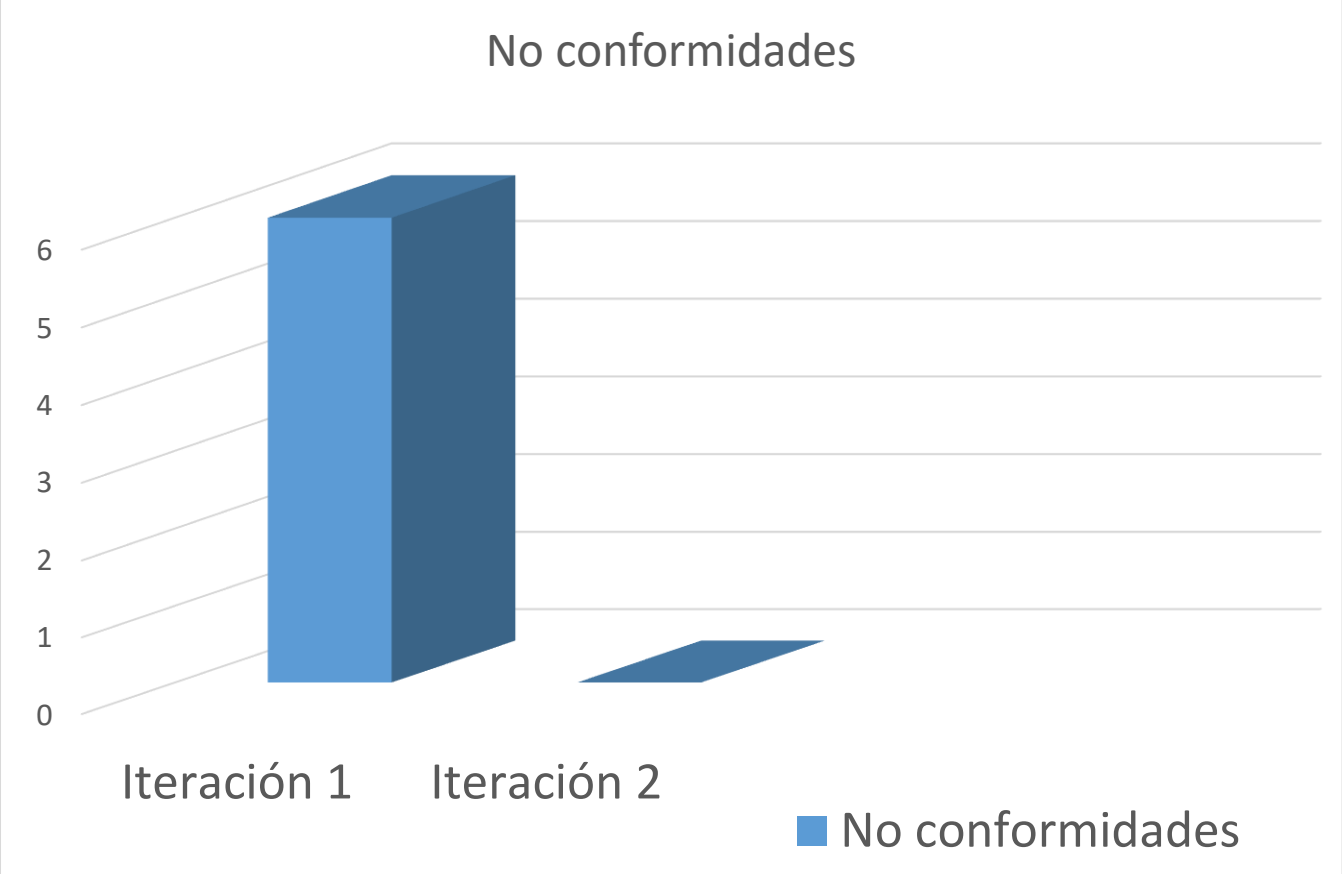


Figura 13: Estadísticas del número de no conformidades encontradas

3.4 Conclusiones del capitulo

En este capítulo fue descrito el proceso de implementación para desarrollar una aplicación utilizando el SDK de Alfresco para la plataforma móvil Android, a través de los artefactos utilizados para llevar a cabo la aplicación; los diagramas de componentes, despliegue y paquetes. Se expuso el estándar de codificación empleado en la escritura del código fuente del sistema. Se presentó la estrategia de prueba seleccionada, que abarcó la realización de las pruebas unitarias y las pruebas al sistema como métrica para erradicar los fallos y medir el correcto funcionamiento y eficiencia del producto.

Conclusiones

Luego de la investigación realizada se concluye que:

- El análisis realizado en el estudio de estado del arte permitió conocer las herramientas existentes que realizan notificaciones como punto de partida de la investigación, sin embargo no resolvían el proceso de recibir, enviar y emitir notificaciones para dispositivos móviles.
- Se evidenció la necesidad de la creación de una solución informática que cumpliera con las especificaciones del cliente, permitiendo utilizar las funcionalidades que brindaron las otras aplicaciones estudiadas se realizó la implementación de la solución propuesta.
- El diseño realizado de la solución propuesta permitió obtener un mejor entendimiento y organización del funcionamiento del sistema, permitiendo una mayor comprensión de las funcionalidades a implementar.
- Se le realizaron las pruebas de caja negra al componente de visualización de notificaciones para dispositivos móviles, permitiendo cumplir con las necesidades y expectativas del cliente en el sistema informático.

Referencias Bibliográficas

40 de fiebre. (6 de junio de 2018). Obtenido de https://www.40defiebre.com/que-es/diseno-responsive/

AdictosalTrabajo. (13 de marzo de 2018). Obtenido de https://www.adictosaltrabajo.com/tutoriales/grasp/

Aenor. (2016). *UNE-ISO. Infomación y documentación. Gestión de documentos. Parte 1: Conceptos y principios.*

AENOR. (2016.). *UNE-ISO 15489-1:2016. Información y documentación. Gestión de documentos. Parte 1: Conceptos y principios.* Obtenido de http://www.aenor.es/aenor/normas/normas/fichanorma.asp?codigo=N0057440.

Alfresco. (5 de febrero de 2018). Obtenido de Gestion de documentos: http://alfresco.com

Alfresco Content Services. (19 de febrero de 2018). Obtenido de https://www.alfresco.com/es/plataforma/servicios-de-contenido-ecm

Android. (diciembre de 2017). Obtenido de http://www.android.com/.

Asier Marqués. (22 de marzo de 2018). Obtenido de http://asiermarques.com/2013/conceptos-sobre-apis-rest/

Belloch, C. (2018). Las tecnologías de la información y comunicación en el aprendizaje., (pág. 9). Valencia.

Biblioteca multimedia SUD. (enero de 2018). Obtenido de https://www.lds.org/media-library/accessing-media-mobile?lang=spa

Braude, E. (13 de marzo de 2018). *Scielo*. Obtenido de https://scielo.conicyt.cl/scielo.php?script=sci_arttext&pid=S0718-07642013000300012

Definicion. (enero de 2018). Obtenido de https://definicion.mx/notificacion/

digival.es. (18 de junio de 2018). Obtenido de https://www.digival.es/blog/que-es-una-responsive-web/

Dispositivos móviles . (20 de junio de 2018). Obtenido de https://revista.seguridad.unam.mx/numero-07/dispositivos-moviles

Dreig, D. R. (2007). *El Caparazón*. Obtenido de http://www.dreig.eu/caparazon/2008/06/01/dispositivos-moviles-aproximacion-resume/

EcuRed. (8 de marzo de 2018). Obtenido de https://www.ecured.cu/Gestor_de_Documentos_Administrativos_eXcriba

EcuRed. (13 de marzo de 2018). Obtenido de https://www.ecured.cu/Patrones_Gof

EcuRed. (11 de mayo de 2018). Obtenido de https://www.ecured.cu/Pruebas_de_software

El Androide Libre. (5 de junio de 2018). Obtenido de https://elandroidelibre.elespanol.com/2016/03/aplicaciones-notificaciones-pantalla-bloqueo.html

El Androide Libre. (5 de junio de 2018). Obtenido de https://elandroidelibre.elespanol.com/2016/03/aplicaciones-notificaciones-pantalla-bloqueo.html

El Conspirador. (9 de marzo de 2018). Obtenido de https://www.elconspirador.com/2013/12/21/que-es-y-para-que-sirve-un-modelo-conceptual/

Erich Gamma, R. H. (13 de marzo de 2018). *Elements of Reusable Software*. Obtenido de http://www.javier8a.com/itc/bd1/articulo.pdf

Español, E. (enero de 2018). *Eel androide libre*. Obtenido de https://elandroidelibre.elespanol.com/2016/03/aplicaciones-notificaciones-pantalla-bloqueo.html

EXCRIBA 3.1. (20 de 1 de 2018). Obtenido de http://www.uci.cu/investigacion-y-desarrollo/productos/xabal/excriba-31

Figueroa, R. y. (2008). Metodologías Tradicionales vs Metodologías Ágiles. *Universidad Técnica Particular de Loja, Escuela de Ciencias en Computación*.

FreeMarker. (7 de marzo de 2018). Obtenido de https://freemarker.apache.org/

(1994). *Gamma et al.*

Geek the Planet. (13 de marzo de 2018). Obtenido de http://geektheplanet.net/5462/patrones-gof.xhtml

Geek the Planet. (15 de marzo de 2018). Obtenido de geektheplanet.net/5462/patrones-gof.xhtml

Google Play. (19 de febrero de 2018). Obtenido de https://play.google.com/store/apps/details?id=org.alfresco.mobile.android.application&hl=es

Google Play. (19 de febrero de 2018). Obtenido de https://play.google.com/store/apps/details?id=com.activiti.android.app

Horstman Cay S., G. C. (2015). Java. *Core Java 2 Volumen I*.

Ingenio DS. (12 de marzo de 2018). Obtenido de https://ingeniods.wordpress.com/2013/09/16/patrones-arquitectonicos/

Internacionalweb. (21 de junio de 2018). Obtenido de https://www.internacionalweb.com/noticias/caracteristicas-y-ventajas-del-diseno-responsive-para-tu-pagina-web

Java. (5 de marzo de 2018). Obtenido de https://www.java.com/es/download/faq/whatis_java.xml

kingofapp.es. (21 de junio de 2018). Obtenido de https://kingofapp.es/blog/notificaciones-push-usos-ventajas-app/

Lara, M. (2015). *Pruebas de software.*

Maestros del web. (6 de 3 de 2018). Obtenido de http://www.maestrosdelweb.com/que-es-javascript/

Object Management Group. Unified Modeling Language. (24 de febrero de 2018). Obtenido de http://www.uml.org/

Ochoa, S. (2005). *Patrones Arquitectonicos.* Madrid.

Olivera, P. (1 de febrero de 2008). *PCWorld de IDG*. Obtenido de http://www.idg.es/pcworldtech/Sistemas-operativos-moviles:-en-busca-de-un-estand/art188636-.htm

Pérez, M. (19 de junio de 2017). *Herramienta de desarrollo de software*. Obtenido de https://www.researchgate.net/publication/228956653_Herramientas_de_Desarrollo_de_Software_Hacia_la_Construccion_de_una_Ontologia

Pixelware. (2018). *pixelware*. Recuperado el enero de 2018, de http://pixelware.com/

Pressman. (2014). 6ta Edición.

PRESSMAN, R. (2014). *Ingeniería del software. Un enfoque práctico.* Obtenido de https://www.tesuva.edu.co/phocadownloadpap/Ingenieria_del_Software._Un_Enfoque_Practico.pdf.

PRESSMAN, S. (2013). *Ingeniería de software. Un enfoque practico.*

Prezi. (12 de marzo de 2018). Obtenido de https://prezi.com/p4vuh3fapd5c/modelo-de-diseno-de-software/

Prezi. (12 de marzo de 2018). Obtenido de https://prezi.com/p4vuh3fapd5c/modelo-de-diseno-de-software/

Prezi. (12 de marzo de 2018). Obtenido de https://prezi.com/p4vuh3fapd5c/modelo-de-diseno-de-software/

Prezi. (12 de marzol de 2018). Obtenido de https://prezi.com/p4vuh3fapd5c/modelo-de-diseno-de-software/

Rabbitmq. (12 de marzo de 2018). Obtenido de https://www.rabbitmq.com/#features

RUIZ, B. y. (2015.). *Módulo Recomendaciones del sistema para repositorios digitales REPXOS 3.0.* Universidad de las Ciencias Informáticas. Obtenido de https://repositorio.uci.cu/jspui/bitstream/123456789/7178/1/TD_07973_15

Sanchez, T. (2015). *Metodología de desarrollo para la Actividad productiva de la UCI.*

Sánchez, T. R. (2015). Metodologia de desarrollo para la Actvidad productiva de la UCI. La Habana: Universidad de las Ciancias Informáticas.

SCRIBD. (9 de marzo de 2018). Obtenido de https://es.scribd.com/doc/37187866/Requerimientos-funcionales-y-no-funcionales

SISCOOP. (9 de marzo de 2018). Obtenido de http://dspace.espoch.edu.ec/bitstream/123456789/188/1/EspecificacionRequerimientosSoftware.pdf

Slideshare. (12 de marzo de 2018). Obtenido de https://es.slideshare.net/Autentia/patrones-de-creacin

Slideshare. (13 de marzo de 2018). Obtenido de https://es.slideshare.net/Autentia/patrones-estructurales-39092993

Softonic. (11 de diciembre de 2017). Obtenido de https://android-sdk.softonic.com/#app-softonic-review

Sputnik Mundo. (30 de mayo de 2018). Obtenido de https://mundo.sputniknews.com/tecnologia/201708291071951100-telefonos-venta-estudio/

Uptodown. (12 de diciembre de 2017). Obtenido de https://android-studio.uptodown.com/windows

Visual Paradigm Copyright. (22 de enero de 2018). Obtenido de UML, BPMN and Software Design Tools for Agile Teams: https://www.visual-paradigm.com/

Bibliografía

Aenor. (2016). *UNE-ISO. Infomación y documentación. Gestión de documentos. Parte 1: Conceptos y principios.*

AENOR. (2016.). *UNE-ISO 15489-1:2016. Información y documentación. Gestión de documentos. Parte 1: Conceptos y principios.* Obtenido de http://www.aenor.es/aenor/normas/normas/fichanorma.asp?codigo=N0057440.

Belloch, C. (2018). Las tecnologías de la información y comunicación en el aprendizaje., (pág. 9). Valencia.

Dispositivos móviles . (20 de junio de 2018). Obtenido de https://revista.seguridad.unam.mx/numero-07/dispositivos-moviles

EXCRIBA 3.1. (20 de 1 de 2018). Obtenido de http://www.uci.cu/investigacion-y-desarrollo/productos/xabal/excriba-31

Figueroa, R. y. (2008). Metodologías Tradicionales vs Metodologías Ágiles. *Universidad Técnica Particular de Loja, Escuela de Ciencias en Computación*.

Horstman Cay S., G. C. (2015). Java. *Core Java 2 Volumen I*.

Ingenio DS. (12 de marzo de 2018). Obtenido de https://ingeniods.wordpress.com/2013/09/16/patrones-arquitectonicos/

Internacionalweb. (21 de junio de 2018). Obtenido de https://www.internacionalweb.com/noticias/caracteristicas-y-ventajas-del-diseno-responsive-para-tu-pagina-web

Java. (5 de marzo de 2018). Obtenido de https://www.java.com/es/download/faq/whatis_java.xml

Lara, M. (2015). *Pruebas de software.*

Object Management Group. Unified Modeling Language. (24 de febrero de 2018). Obtenido de http://www.uml.org/

Ochoa, S. (2005). *Patrones Arquitectonicos.* Madrid.

Olivera, P. (1 de febrero de 2008). *PCWorld de IDG*. Obtenido de http://www.idg.es/pcworldtech/Sistemas-operativos-moviles:-en-busca-de-un-estand/art188636-.htm

Pérez, M. (19 de junio de 2017). *Herramienta de desarrollo de software*. Obtenido de https://www.researchgate.net/publication/228956653_Herramientas_de_Desarrollo_de_Software_Hacia_la_Construccion_de_una_Ontologia

Pixelware. (2018). *pixelware*. Recuperado el enero de 2018, de http://pixelware.com/

Pressman. (2014). 6ta Edición.

PRESSMAN, R. (2014). *Ingeniería del software. Un enfoque práctico.* Obtenido de https://www.tesuva.edu.co/phocadownloadpap/Ingenieria_del_Software._Un_Enfoque_Practico.pdf.

PRESSMAN, S. (2013). *Ingeniería de software. Un enfoque practico.*

RUIZ, B. y. (2015.). *Módulo Recomendaciones del sistema para repositorios digitales REPXOS 3.0.* Universidad de las Ciencias Informáticas. Obtenido de https://repositorio.uci.cu/jspui/bitstream/123456789/7178/1/TD_07973_15

Sanchez, T. (2015). *Metodología de desarrollo para la Actividad productiva de la UCI.*

Sánchez, T. R. (2015). Metodologia de desarrollo para la Actvidad productiva de la UCI. La Habana: Universidad de las Ciancias Informáticas.

Visual Paradigm Copyright. (22 de enero de 2018). Obtenido de UML, BPMN and Software Design Tools for Agile Teams: https://www.visual-paradigm.com/

Rabbitmq. (12 de marzo de 2018). Obtenido de https://www.rabbitmq.com/#features