UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e Ingenierías

Software de Control de Documentación

Proyecto Técnico

Miguel Antonio Yoncón Changkuón

Ingeniería de Sistemas

Trabajo de titulación presentado como requisito
para la obtención del título de
Ingeniero en Sistemas

Quito, 22 de diciembre de 2015

UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

COLEGIO CIENCIAS E INGENIERÍAS

HOJA DE CALIFICACIÓN
DE TRABAJO DE TITULACIÓN

Software de Control de Documentación

Miguel Antonio Yoncón Changkuón

Calificación:

Nombre del profesor, Título académico Fausto Pasmay, Ms.C.

Firma del profesor _____

Quito, 22 de diciembre de 2015

Derechos de Autor

Por medio del presente documento certifico que he leído todas las Políticas y Manuales de la Universidad San Francisco de Quito USFQ, incluyendo la Política de Propiedad Intelectual USFQ, y estoy de acuerdo con su contenido, por lo que los derechos de propiedad intelectual del presente trabajo quedan sujetos a lo dispuesto en esas Políticas.

Asimismo, autorizo a la USFQ para que realice la digitalización y publicación de este trabajo en el repositorio virtual, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación Superior.

Firma del estudiante: _____

Nombres y apellidos: Miguel Antonio Yoncón Changkuón

Código: 00018010

Cédula de Identidad: 1713675369

Lugar y fecha: Quito, 22 de diciembre de 2015

RESUMEN

El trabajo realizado consiste en una aplicación de escritorio y para dispositivos móviles que permite realizar un efectivo control de documentación dentro de una empresa. Esta aplicación se encuentra alojada en la nube, por lo cual permite tener portabilidad y escalabilidad para su mayor desarrollo e implementación.

Una de las motivaciones para realizar este trabajo fue el aumentar la eficiencia y reducir tiempos administrativos para el proceso manual implementado en la empresa. De esta manera, la automatización de este proceso permite que se manejen adecuadamente los documentos que describen actividades a ser ejecutadas dentro de un Data Center. Dado a que estos documentos contienen información sensible, la seguridad también es tomada en cuenta para su procesamiento, por esto se mejora la calidad del mismo proceso.

Adicionalmente se logró que el trabajo pueda concentrar toda la información en un repositorio centralizado, por lo cual la misma búsqueda de los documentos se facilita.

A futuro, se busca que el proyecto se implemente por completo dentro de la empresa y que se continúe su uso para futuras ocasiones, mejorando un proceso que en términos administrativos tenía muchos desperdicios.

ABSTRACT

The project developed consists of a desktop and mobile devices application, which allows to perform an effective documentation control inside a company. This application is hosted in the cloud, therefore, it allows portability and scalability for its development and implementation.

One of the reasons that motivated the development of this work was to increase the efficiency and reduce the administrative times in the manual process implemented in the company. In consequence, the automatization of this process allows an adequate handling of the documents that describe activities to be executed inside a Data Center. Having these documents to contain sensitive information, their security is also handled for their processing, as a result, it allowed an improvement in the quality of the process itself.

In addition, all the information was stored in a centralized repository, which eased the document searching.

In the future, it is planned to completely implement the project inside the company and to continue its use for future occasions, improving the process that in administrative terms, had excessive waste.

TABLA DE CONTENIDO

ANTECEDENTES Y JUSTIFICACIÓN

En la actualidad existe un proceso implementado dentro de una empresa, enfocado a aprobar solicitudes de documentos con actividades para ejecutarse dentro de un Centro de Datos. Dado a que las actividades descritas en los documentos son ejecutadas en un área sensible y de acceso restringido y controlado, es necesario contar con un proceso que administre efectivamente dicha documentación y las aprobaciones de las tareas que se solicitan ejecutar. Estas aprobaciones se realizan de manera manual a través de correos, que deben contener toda la información recopilada de los documentos solicitados semanalmente, que finalmente permiten cierto control y ejecución de las tareas, pero no de una manera ágil y ordenada. Como consecuencia, existe tiempo administrativo que se presenta al buscar información pasada y en obtener las aprobaciones de cada uno de los documentos; adicionalmente ocurre el caso de que el proceso no se ejecute de manera ordenada si existiese una solicitud no enviada para las revisiones que se realizan semanalmente.

El trabajo realizado trata de un programa diseñado para controlar documentación digital. El Software está orientado a tener un repositorio centralizado y a reforzar y agilizar el proceso de aprobaciones de documentos. El manejo de esta información, a través de un sistema digital, permite que los tiempos administrativos se reduzcan y se tenga mayor rapidez para evaluar o analizar la información tratada.

OBJETIVOS

Objetivo General

Desarrollar un sistema que permita centralizar la información, reforzar y agilizar un proceso empresarial de aprobación de documentos, logrando una reducción de la carga administrativa de los recursos involucrados y controlando la ejecución ordenada de cada una de las fases en el proceso utilizado.

Objetivos Secundarios

1. Configurar un control de accesos
2. Dividir privilegios de los usuarios del sistema
3. Permitir la creación de nuevos usuarios
4. Permitir la eliminación de usuarios existentes
5. Habilitar la posibilidad de cambiar los permisos en cada usuario
6. Validar los datos requeridos para la carga de un documento
7. Cargar la información en un repositorio centralizado
8. Generar una manera de visualizar los documentos cargados a la base con los detalles principales del mismo
9. Notificar a los aprobadores cuando se carga un nuevo documento
10. Notificar a los usuarios cuando se modifica el estatus de un documento
11. Permitir el acceso de la aplicación desde un dispositivo móvil con las mismas funcionalidades del programa de escritorio.

SOPORTE TEÓRICO

Para este sistema se decidió utilizar una arquitectura de tres capas que permite tener una adecuada división de las tareas principales que se ejecutan, de modo que se pueda cumplir con los objetivos establecidos. En este caso, existen las capas de datos, con un servidor dedicado exclusivamente a todas las transacciones de información que se desplegarán en la capa de presentación; y la capa de negociación que es la encargada de toda la lógica de la programación.

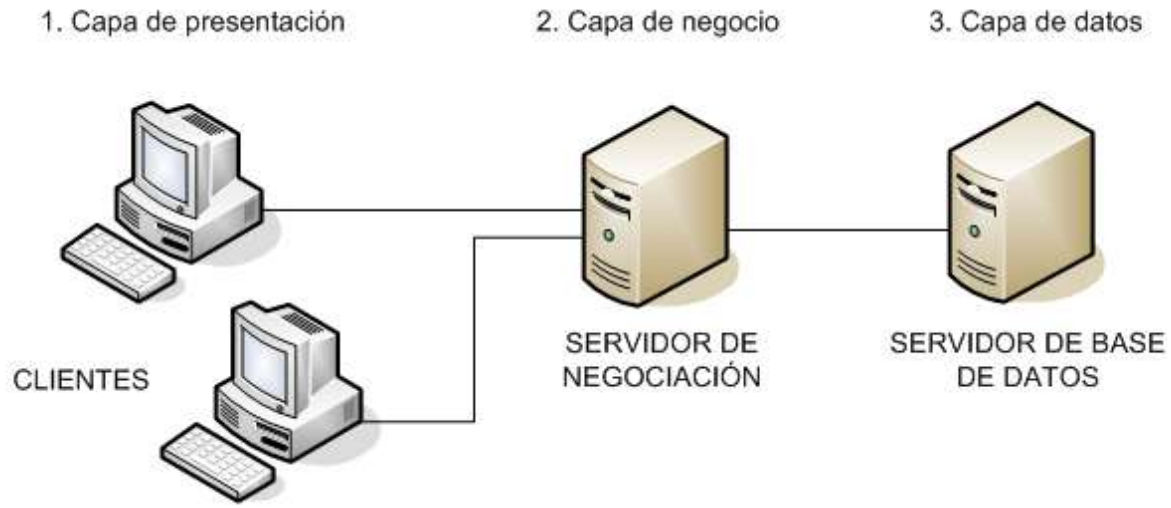

Figura 1: Arquitectura de tres capas

Para la capa de presentación, se utilizó una interface web, de manera que sea de fácil acceso para los usuarios y de rápida implementación. Referente al front-end, se escogió HTML5 como lenguaje para que la programación se oriente al uso en dispositivos móviles, debido al aumento de su popularidad en la actualidad, hecho que se enfoca para conseguir el objetivo 11 de habilitar el acceso desde un dispositivo móvil a la aplicación. La facilidad de utilizar este tipo de interface es que solamente se depende de una conexión de datos y de un

navegador para el acceso de cualquier usuario. Esto facilita a que el ingreso a la aplicación sea accesible fácilmente y que se presente una distribución gráfica intuitiva para cada usuario.

Se utilizó AngularJS como framework de la programación, que a diferencia de utilizar puramente código HTML para la programación web, permite cargar componentes adicionales a la página para generación de contenido. La página utiliza un modelo diseñado con Javascript y utiliza componentes de AngularJS, que se cargan localmente para facilitar el despliegue de controladores que muestran la información que se desea presentar al usuario. Por lo tanto, a más de ser un framework Open-Source con lo que se tiene facilidad de obtener soporte en el desarrollo, permite una implementación rápida y ejecución sencilla de pruebas por la manera en que están diseñados los módulos de código.

AngularJS utiliza un template en HTML que funciona para visualizar la información básica que se presentará al usuario. Adicionalmente al HTML básico, existen elementos que se conocen como directivas, los que permiten adicionar algunos atributos y elementos al código presentado, como botones o secciones que permiten ejecutar ciertas acciones. Estas acciones más los datos que presentamos al usuario y que se cargan a través de diferentes scripts de servicios y controladores, conforman el modelo de la aplicación y con el cual el usuario puede interactuar.

La aplicación funciona dentro de una plataforma de servicios web de Amazon (AWS) y está montada sobre un servidor Linux con componente Apache, utilizando una base de datos relacional MySQL que se comunica con la aplicación a través de PHP para transacciones de datos como se muestra en la imagen inferior.

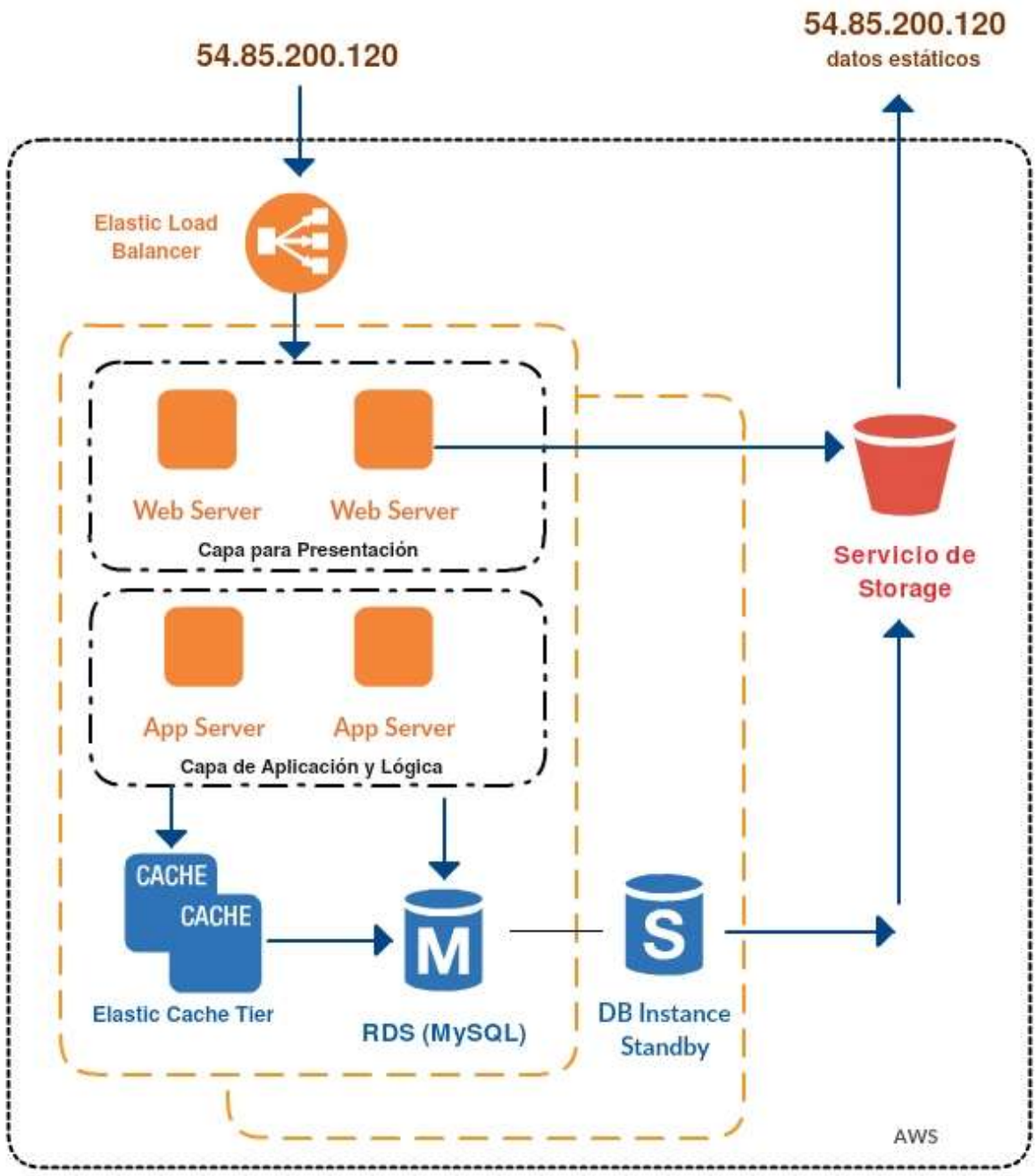

Figura 2: Arquitectura de Aplicación alojada en la nube

Para la configuración de la aplicación dentro de la capa de negociación, se diseñó el sistema dentro de los frameworks mencionados y se incluyó un control de accesos de modo que se limite el acceso a la aplicación y posteriormente se tenga una división clara de las tareas que puede ejecutar cada usuario a través de privilegios. En el proceso establecido de

aprobación de documentos, se tienen 3 tipos de usuarios: un usuario único encargado de consolidar la información y que busca y notifica las aprobaciones de todos los involucrados; un usuario solicitante que presenta el documento ante un comité y es quien necesita ejecutar las tareas descritas en el documento, que pueden involucrar más personal para las acciones requeridas; y 3 aprobadores que validan que la información solicitada esté correcta y que el impacto o riesgo al ejecutar las tareas dentro del Data Center, estén controladas adecuadamente.

Finalmente para la capa de datos, se utilizó una base de datos relacional con MySQL en el servidor, el cual recibe y envía transacciones a través de PHP para el despliegue de la información solicitada dentro del programa. Esto permite que la comunicación con la aplicación sea sencilla y compatible fácilmente con los componentes que se cargan dentro de la página web con Javascript. Para esta base, se crearon 5 tablas con la información que se utiliza para desplegar: usuario, categoría del usuario (para control de privilegios), área (a la cual pertenece el usuario), estatus del documento y documento (key) como se muestra en la imagen inferior. Esto permite cumplir el objetivo 7 de tener un repositorio centralizado para el almacenamiento de la información, debido a que existe un servidor dedicado a los servicios de Base de Datos (RDS) dentro de la plataforma escogida.

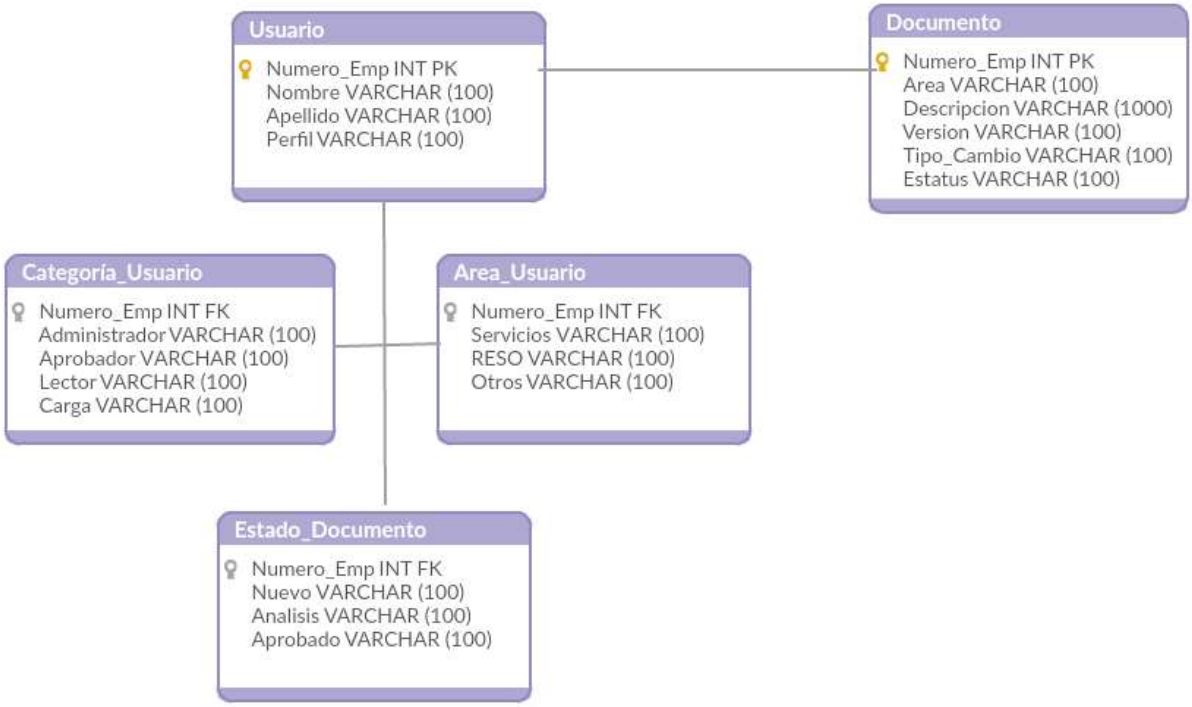

Figura 3: Diagrama de tablas relacionales

DESARROLLO DEL SISTEMA

Utilizando la parte teórica descrita anteriormente, cada vez que se abre una nueva instancia de conexión al sistema, el programa despliega el índice (index.html) con todos los scripts y servicios que permitirán al usuario tener la vista del menú principal para iniciar sesión. Así utilizando las sintaxis "`<script src=...`", se carga la información que se maneja a través de los otros archivos como scripts de carga de componentes, servicios, controladores, etc.

Se definen 4 tipos de usuarios que pueden ejecutar las mismas acciones como en el proceso actual, a través de permisos administrados por un usuario designado. Esto busca controlar el acceso a la aplicación con un login básico de usuario y contraseña (objetivo 1) y que cada uno de los usuarios pueda cumplir las funciones que le corresponde. Los privilegios definidos son de un (1) administrador, un (1) usuario de carga de información, aprobadores y visores de información. Este control permite cumplir el objetivo 2 de la aplicación desarrollada.

Estos privilegios se los controla a través de las cookies del navegador utilizando un script "angular-cookies.js" (ver Anexo A: Código), el cual permite utilizar una función constructora (factory) que inyecta una variable "cookie" dentro de toda la sesión que utilice el usuario en el navegador. Esta variable tiene alcance global y a través del archivo "authentication-service.js" se encarga de validar que las credenciales estén correctamente ingresadas según lo que se tiene registrado en la base de datos de usuarios. Y en el script principal "app.js" se analiza que la sesión se mantenga abierta y permite que se decida cuáles son los permisos que se deben otorgar, así en un lazo que despliega la vista con un controlador que le corresponden al privilegio del usuario, se muestran las opciones correspondientes hasta que ejecute una acción adicional. Por ejemplo, las siguientes líneas muestran la

inyección de los privilegios de un usuario con permisos de visor y la carga de la vista una vez que se ha autenticado:

```
config.$inject = ['$routeProvider', '$locationProvider'];
function config($routeProvider, $locationProvider) {
    $routeProvider
        .when('/ver', {
            controller: 'ViewController',
            templateUrl: 'ver/ver.view.html',
            controllerAs: 'vm'
        })
```

El último permiso de visor sólo permite al usuario leer la información de los documentos cargados en la base de datos y es el que se otorga por defecto cuando cualquier usuario se registra en el programa utilizando el script “register.controller.js”; de esta manera, se permite la creación de usuarios individualmente, sin la necesidad de tener un administrador que deba ejecutar esta tarea, cumpliendo el objetivo 3 del proyecto. Para esto, se utiliza el script “ver.controller.js”

El usuario administrador, tiene la capacidad de modificar los permisos de las cuentas existentes a más de poder cargar información en la base de datos; adicionalmente puede eliminar cuentas que ya no se desee tener, por ejemplo, si se diera el caso de que una persona se retire de la compañía y ya no sean necesarios los accesos de la misma en la aplicación, con lo cual se logra obtener el objetivo 4 y 5 del proyecto. Esto se ejecuta en el controlador “admin.controller.js” con la función “AdminController”, con la cual, se permite realizar cambios de privilegios a los usuarios (UserServce.CambiarCategoriaUsuario), cargar todos los usuarios disponibles (function loadAllUsers()) y eliminar los que se desee (function deleteUser(id)).

El usuario de carga, al igual que el usuario administrador, puede subir documentos de extensiones .xls y .xlsx a la base junto con una validación de campos, de esta manera, es seguro que la información ingresada sea la correcta. La información que se sube en la base se valida de modo en que todos los campos estén llenos y en el formato adecuado, utilizando banderas en cada uno y combo boxes que permiten al usuario seleccionar la opción adecuada, así cuando todos los campos se hayan completado correctamente, se habilita el botón de carga (objetivo 6). Esto se realiza a través de validaciones con expresiones regulares en el código HTML y se muestran mensajes dentro del controlador “subir.controller.js”. En el ejemplo de código, se muestra la función “post” para cargar el documento en la base de datos a través del módulo “upload.php” de carga:

```
$http.post('api/upload.php', fd, {
        withCredentials: true,
        headers: {'Content-Type': undefined },
        transformRequest: angular.identity
    }).success(function(success){
      if(success.success==false)
      {
        $anchorScroll();
        sfiles="";
        FlashService.Error(success.message);
      }
      else{
        FlashService.Success(success.message, true);
        sfiles=success.archivo;
      }
    }
```

El despliegue de la información de documentos cargados, se realiza a través de una tabla con los datos ingresados en los formularios. Esto permite tener una visualización rápida del contenido de cada documento cargado, optimizando el tiempo de procesamiento de cada solicitud (objetivo 8). Ya sea que se cargue un nuevo documento, o que se realice un cambio en el estatus de uno, el programa ejecuta un comando para que se cargue la aplicación por defecto en el dispositivo que se está utilizando. Para cumplir con los objetivos 9 y 10, esta funcionalidad permite enviar una notificación a los aprobadores para agilitar el proceso y hacerlo de manera más automática utilizando una función mailTo() al momento de que se detecta una acción de actualización.

Todas las funciones anteriormente descritas se utilizan de igual manera para la plataforma móvil que se desplegó para el proyecto. En este caso, los controladores que se utilizan funcionan de la misma manera, pero a diferencia del proyecto desplegado del lado de una PC, existen componentes que se cargan localmente en los dispositivos móviles. Dada la capacidad que tienen actualmente los dispositivos, se utilizan elementos como stores, que permiten guardar información localmente para agilizar la carga de información de la aplicación. Luego para la ejecución de algunos scripts, se reutilizan los mismos componentes que se tienen en la aplicación de escritorio gracias a la modularidad que presenta la codificación del framework AngularJS. En el siguiente script, se muestra cómo se cargan las áreas a las que puede pertenecer un usuario que se registrará en el sistema:

```
Ext.define('ProyectoMY.store.Areas', {
    extend: 'Ext.data.Store',
     requires: [
    'Ext.data.proxy.JsonP'
    ],
```

```
    config: {
        autoLoad: true,
        storeId: 'areasstore',
        proxy: {
            type: 'ajax',
            url: 'http://54.85.200.120/api/areas.php',
            reader: {
                type: 'json',
                rootProperty: 'items'
            }
        },
         fields: ['nombre','id']
    }
});
```

Estos paquetes se comprimen posteriormente y son los que se guardarán en el dispositivo móvil. Para esto, es necesario tomar en cuenta que lo que se busca es minimizar el espacio de instalación y que la aplicación pueda ser lo más ligera posible para no afectar el desempeño de la aplicación.

RESULTADOS, CONCLUSIONES Y RECOMENDACIONES

Para la concepción inicial del desarrollo de la aplicación, se optó por una arquitectura diferente, utilizando otra base de datos (exist-DB) en la que algunas funcionalidades venían diseñadas y precargadas para reducir el tiempo de desarrollo de la aplicación en el lado del servidor, lo cual permitía tener una mayor disponibilidad para aumentar el enfoque de desarrollo en las aplicaciones móviles del proyecto. Lamentablemente se pudo determinar que la aplicación escogida inicialmente no tenía un buen manejo de los perfiles de usuarios, en consecuencia, se limitaba mucho el conseguir el objetivo de controlar ordenadamente las aprobaciones de documentos. Por tal razón, se optó por la opción que actualmente se desarrolló y se programó toda la arquitectura descrita anteriormente que de igual manera, permite un acceso móvil y agilita la aprobación de los documentos relacionados con el proceso.

A través del uso del framework AngularJS, se facilitó la programación de la aplicación, a pesar de que se debió programar todas las funcionalidades nuevamente. Este método es que permitió que la aplicación sea ligera y eficiente para la ejecución de cada uno de los objetivos que se buscó. Adicionalmente, aprovechando el tipo de lenguajes similares que utiliza, se facilita la programación en las plataformas móviles y se reduce el tiempo de su implementación y también permitieron mejorar el conocimiento sobre este tipo de dispositivos y su funcionamiento, ya que los conceptos que se utilizan y las consideraciones que se deben tener para este tipo de programación, son distintas a las que se utilizan para aplicaciones de dispositivos de escritorio como el uso de recursos (memoria, gráficos, animaciones) y facilidades de acceso para los usuarios.

Es recomendable que este tipo de proyectos tengan un lenguaje común y sencillo de utilizar, ya que permite reducir considerablemente los tiempos de desarrollo, pruebas e implementación de la aplicación final. En este caso, el uso de lenguajes como Javascript, PHP y HTML, facilitaron la codificación debido a que permiten tener compatibilidad entre las aplicaciones móviles y de escritorio, a diferencia de tener otro despliegue con otra base de datos y funcionalidades como se concibió el proyecto inicialmente. Adicionalmente el soporte que se puede encontrar para este tipo de lenguajes es mayor, y permite solventar inconsistencias de programación de una manera más ágil.

La aplicación generada permitió cumplir con los objetivos iniciales de poder centralizar la información utilizando la base de datos que puede ser accedida desde cualquier dispositivo con datos y un navegador. Adicionalmente se pudo asentar y reforzar el proceso debido a que se utilizan diferentes permisos para los accesos con los usuarios creados, lo cual limita y controla que las actividades ejecutadas, estén dentro de las responsabilidades de cada uno de los usuarios. Con esta implementación se puede agilitar todo el proceso actualmente realizado de forma manual, ya que se reducen cargas administrativas y se utilizan notificaciones para que todo el flujo no se detenga sólo en revisiones semanales de las solicitudes de actividades en el Data Center.

A futuro, se planifica mejorar la versión actual, de manera que se puedan automatizar las notificaciones enviadas, es decir, tomando directamente cada dirección de correo de los aprobadores registrados en la base de datos y enviando el mail de notificación, utilizando un servidor de correos. Por lo tanto, esa carga adicional de tiempo se reduce y se pueden optimizar las tareas de ejecución y revisión de los documentos. En este mismo sentido, se busca que se optimice las imágenes y animaciones utilizadas, para que sea una aplicación más

atrayente para los usuarios, sin que esto afecte considerablemente el desempeño y velocidad de la aplicación, teniendo especial consideración de que esto se pueda acceder desde un dispositivo móvil, lo cual aumenta la agilidad del proceso.

BIBILOGRAFÍA

Perry, D. E.; Wolf, A. L. (1992). "Foundations for the study of software architecture" (PDF). ACM SIGSOFT Software Engineering Notes

Herbst, Nikolas Roman; Samuel Kounev; Ralf Reussner (2012). "Elasticity in Cloud Computing: What It Is, and What It Is Not" (PDF). Proceedings of the 10th International Conference on Autonomic Computing (ICAC 2013), San Jose, CA, June 24–28.

CC BY 3.0 (2013). "What Is Angular?". Retrieved 12 February from https://docs.angularjs.org/guide/introduction

Coman Hamilton. "A sneak peek at the radically new Angular 2.0". Retrieved 2015-10-21.

Amazon S3, Cloud Computing Storage for Files, Images, Videos. Aws.amazon.com (2006-03-01). Retrieved on 2013-08-09.

"What is Cloud Computing by Amazon Web Services | AWS". Aws.amazon.com. Retrieved 2013-07-17

"Comparison of S3QL and other S3 file systems". Retrieved 2012-06-29. https://code.google.com/p/s3ql/wiki/other_s3_filesystems

"Amazon Web Services". AWS Products. Amazon Web Services. Retrieved 23 April 2015. http://aws.amazon.com/products/?nc2=h_l2_p

ANEXOS

Anexo A: Código

Anexo B: Tabla de Módulos vs Objetivos cumplidos

Módulo	Objetivo	Comentario
admin.controller.js admin.view.html	4, 5	Control de usuarios en el sistema
flash.service.js user.service.js user.service.local-storage.js sub.controller.js sub.view.html	2	Control de los privilegios de usuarios existentes
aprobar.controller.js aprobar.view.html	6, 10	Validación y notificación de información ingresada
login.controller.js login.view.html authentication.service.js	1	Control de accesos
register.controller.js register.view.html	3	Registro de nuevos usuarios
subir.controller.js subir.view.html	7, 9	Carga de información a la base y notificación
ver.controller.js ver.view.html	8	Visualización de documentos subidos a la base
app.js index.html	11	Módulos principales del sistema

Anexo C: Manual de Usuario

Para Acceder al sistema, el usuario deberá utilizar un navegador en cualquier computador e ingresar a la dirección http://54.85.200.120.

Posteriormente el usuario debe iniciar sesión en caso de tener credenciales registradas, de lo contrario, deberá registrarse en el sistema, dentro del enlace “Registrarse”.

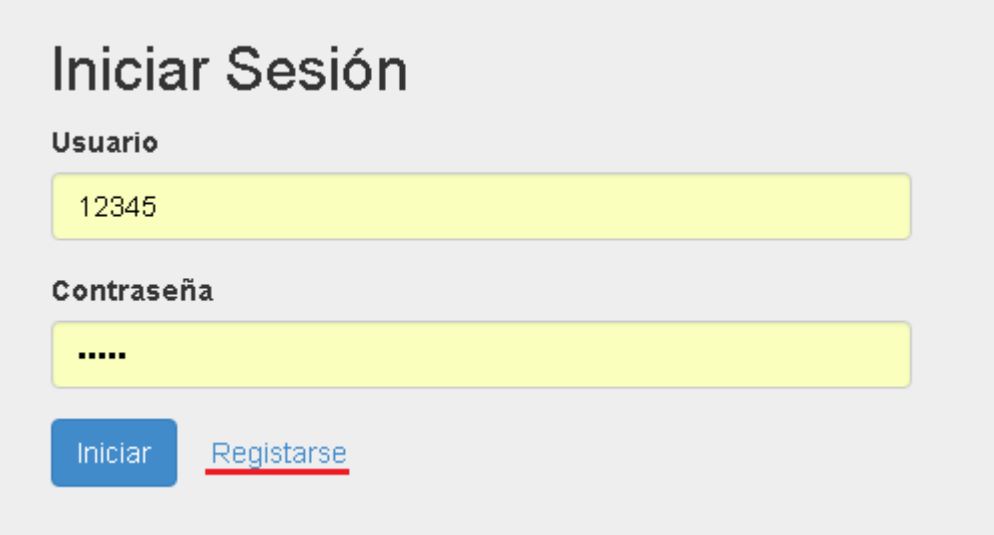

Figura 1C: Pantalla de login

Para el registro en el programa, el usuario deberá completar todos los campos indicados en el formulario (Nombre, Apellido, Área, Número de Empleado y Contraseña). Es importante que el usuario recuerde su número de Empleado, ya que este será a futuro su nombre de usuario. Finalmente deberá hacer click sobre el botón “Registrarse” si ha completado toda la información o en “Cancelar” si desea regresar al menú de inicio.

Una vez que el usuario inicia sesión correctamente, se presentará la pantalla según los permisos que se le haya otorgado. Por defecto, el sistema carga permisos de visor a cualquier usuario que se registra por primera vez. Luego el administrador puede realizar cambios sobre los permisos del resto de usuarios y eliminar usuarios como se muestra en la imagen inferior:

Figura 2C: Listado de usuarios

Para un usuario con permisos solamente de lectura, se presenta una página con todos los documentos cargados en la base de datos y se le permite descargar cada uno de los documentos como se puede apreciar en la imagen inferior.

Figura 3C: Listado de documentos

Para el usuario con permisos de carga de archivos, se le presenta al usuario un formulario que debe completar correctamente para cargar dicha información a la base de datos. Si el formulario contiene errores o información faltante, se impide la carga a la base como se ve en la imagen inferior.

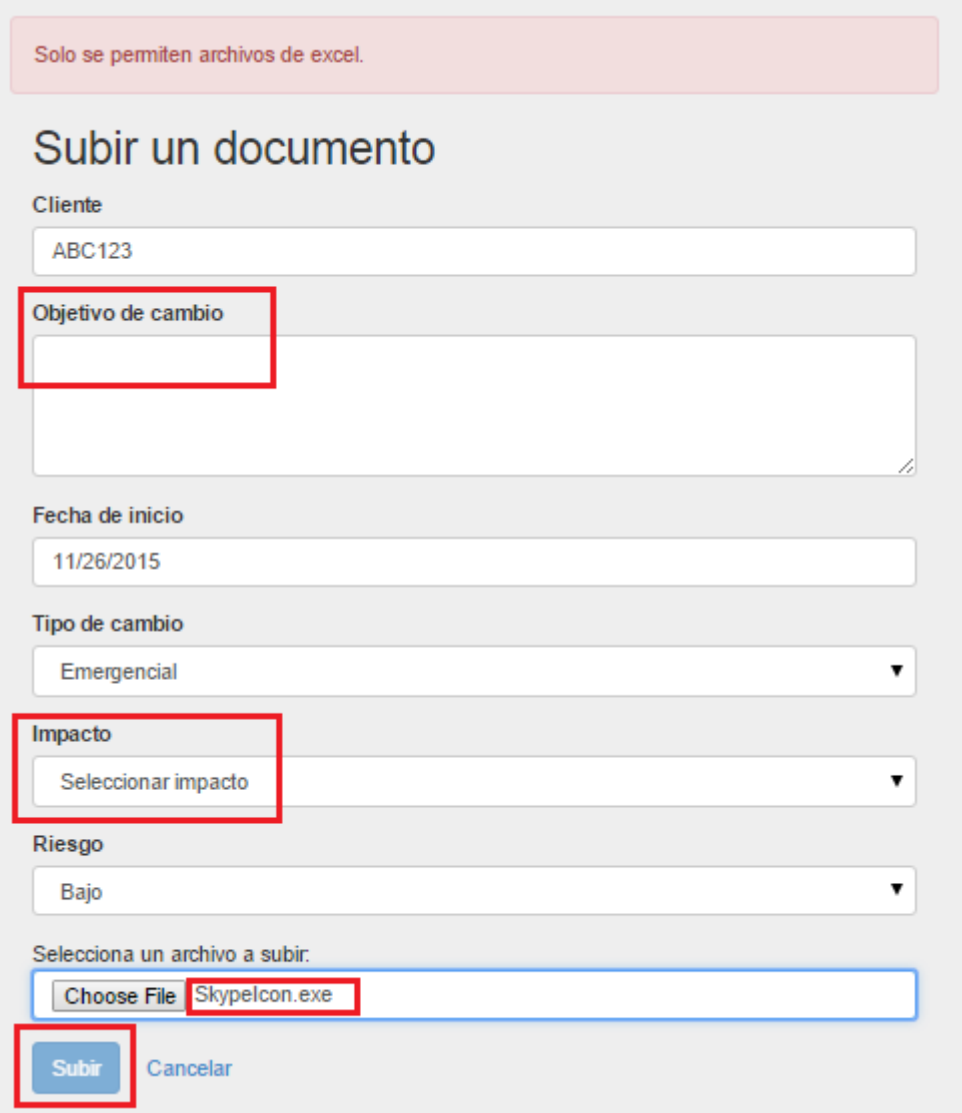

Figura 4C: Carga de documentos con errores

Finalmente, cuando se carga la información de manera acertada, el sistema habilita el botón de “Subir” y se puede ingresar con éxito la misma.

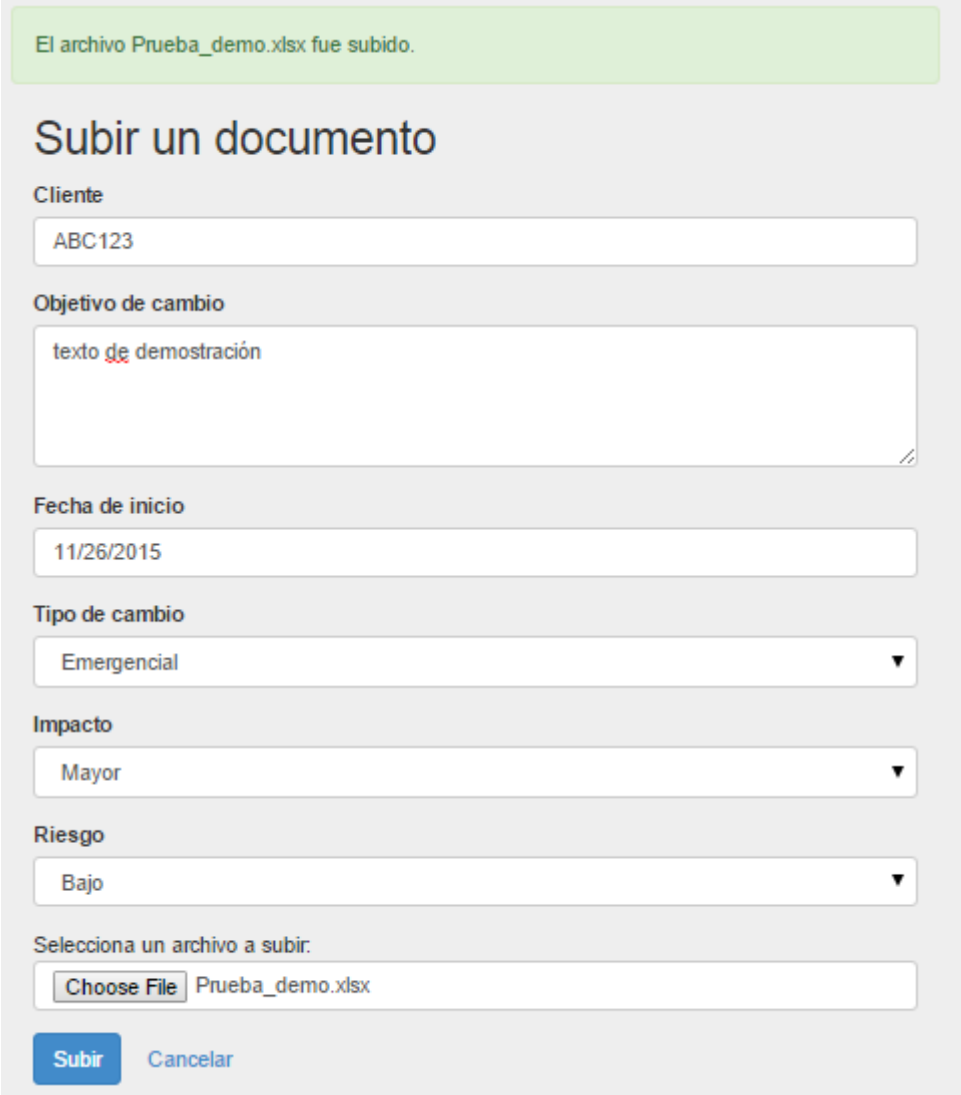

Figura 5C: Carga de documento exitosa

Un usuario que tiene permisos de aprobador puede realizar cambios en el estatus de los documentos, así, cuando se desea aprobar, rechazar o analizar algún cambio, dentro de la tabla con todos los documentos, se puede realizar la actualización como se muestra en la siguiente imagen.

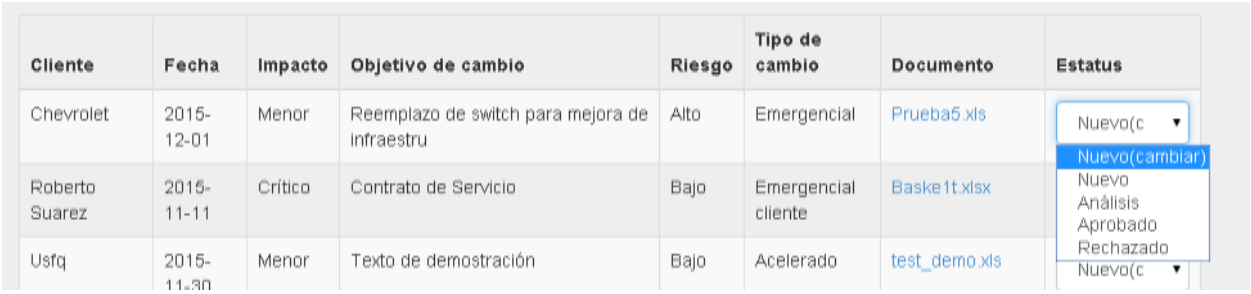

Figura 6C: Cambio de estatus de documentos

Finalmente, el sistema permite enviar notificaciones a los responsables de aprobación de documentos cada vez que se carga un nuevo documento y cuando se hace un cambio en el estatus de aprobación. De esta manera se controla que los aprobadores estén notificados a tiempo de cualquier actualización que ocurra. Estas notificaciones utilizan el cliente de mail configurado en el equipo, por lo tanto, les aparece un envío de mail con un template similar al que sigue:

Figura 7C: Notificación a aprobadores