CICLO FORMATIVO DE GRADO SUPERIOR
DESARROLLO DE APLICACIONES MULTIPLATAFORMA

PROYECTO FIN DE CICLO

Alejandro Andres Huedo del Castillo
Gonzalo Javier Huedo del Castillo

CURSO 2018-19

TÍTULO: Prokom

AUTORES: Alejandro Huedo del Castillo
Gonzalo Huedo del Castillo

TUTOR DEL PROYECTO: Ernesto Ramiro Córdoba

FECHA DE LECTURA: 5 Junio de 2019

En Madrid

Ernesto Ramiro Córdoba

Tutor del PFC

RESUMEN

El proyecto Prokom es un proyecto que estará dirigido a cubrir una necesidad en la empresa de trabajo temporal Prokom.SL, este proyecto se encargará de crear una aplicación para los trabajadores de Prokom y así actualizar su aplicación anterior. Con esta nueva aplicación se les ayudará, ya que, se podrán añadir muchas funciones a la aplicación, como por ejemplo unir una base de datos en la nube de una aplicación Android con la base de datos de SQL en un servidor local.

La aplicación se dividirá en tres partes principalmente, una aplicación de escritorio o de cliente donde las administradoras de Prokom, podrán registrar, modificar y borrar a los trabajadores. La segunda parte, será el servidor, mediante un servidor TCP socket se enviarán los datos al servidor y se guardará en una base de datos de SQL. Y por último una aplicación Android con una base de datos en la nube que se actualizarán en el servidor y se añadirá a la base de datos.

Este proyecto esta principalmente diseñado para que sea más fácil mejorar la aplicación de Prokom, y además añadir las funciones, que sea más fácil a las administradoras registrar un gran número de trabajadores, y ellos mismos puedan rellenar sus datos en una aplicación Android. Las administradoras tendrán el acceso a modificar los perfiles de sus trabajadores registrados si hay algún problema. Además de actualizar la aplicación para simplificar las versiones anteriores.

SUMMARYf

The Prokom's project is made to cover a requirement in the temporary job business Prokom S.L, this project will take care of create an application for the Prokom's workers and actualize their previous application.

With this new application, it will improve their way to do things, many new features could be added, many new features could be added, like for example, connect a data base from Android in the cloud to a local SQL data base.

This application will be divided into three parts, one desktop application where the Prokom's admins, will be able to register, modify and delete the temporary workers. The second part, will be the server, through a TCP socket server, the data will be sent to the server and It will be saved in a SQL data base. And at last an application for Android with a data base in the cloud, that will be actualized in the server and added into the data base.

This application will be divided into three parts, one desktop application where the Prokom's admins, will be able to register, modify and delete the temporary workers. The second part, will be the server, through a TCP socket server, the data will be sent to the server and It will be saved in a SQL data base. And at last an application for Android with a data base in the cloud, that will be actualized in the server and added into the data base.

This project is firstly designed to make easier improve Prokom's application, and add functions, so admins can register a great number of workers, and the workers, themselves add their profiles in the Android application. The admins will be allowed to add, modify or delete, if there is any problem. There will be updates to simplify the previous versions.

AGRADECIMIENTOS

Por parte de familia, ha sido posible gracias a nuestros hermanos, padres, nosotros mismos como hermanos también, apoyándonos mutuamente en todo momento y en los momentos que más flaqueábamos, en los que realmente importaban, estaban allí.

También queremos agradecer en especial a Ernesto, Pilar y Susana, que además de ayudarnos en el propio proyecto, nos han enseñado durante todo el año, no exactamente como hacer lo que estamos haciendo, sino más bien una metodología de aprendizaje y comprensión de la programación y diseño. Gracias a esto, somos capaces de poder gestionar nuevas formas de trabajar.

Los compañeros de clase de este curso nos han dado feedback y ayuda en dudas puntuales, siempre que las hemos tenido.

Muchas gracias a todos los mentados, ellos también son una parte muy importante de este proyecto de final de ciclo.

ÍNDICE

1. INTRODUCCIÓN

Investigación previa sobre la práctica.

En el PFC contactamos con una empresa llamada Prokom.SL, y llegamos al acuerdo de crearle una aplicación de escritorio para actualizar su plataforma informática, ellos nos proporcionaron sus fuentes, logo y paletas de colores.

La aplicación se dividirá en tres partes principalmente.

La aplicación de escritorio donde trabajarán las administradoras donde modificarán, añadirán o eliminarán los datos de los trabajadores.

El servidor que recibirá los objetos de la aplicación del escritorio y este introducirá la información en la base de datos de SQL y también utilizará el
Framework Spring para leer la base de datos en la nube y así poder sincronizarlas.

La aplicación de Android que será utilizada por los trabajadores temporales, donde se registrarán ellos mismos en la base de datos mediante una Tablet.

1.1 OBJETIVOS

El objetivo es innovar la aplicación de la empresa, que actualmente usan Access por una aplicación de escritorio, de fácil uso para el usuario.
Se tratará de un programa Java, implementando una capa visual con Scene Builder para darle un bonito interfaz gráfico. También contará con nuevas funciones, como, por ejemplo, poder filtrar a los usuarios de la base de datos.

Como servidor, utilizaremos un socket TCP para poder transportar la información entre la aplicación del administrador y la base de datos. Se utilizará una base de datos SQL y un sistema de codificación de información AES.

Se añadirá una aplicación Android con dos ventanas, principalmente para registrar la información de los trabajadores que ellos mismo rellenen, a la base de datos. Esta información se almacenará en la una base de datos en la nube
SQL y generará ficheros texto en el servidor, que serán leídos por la aplicación del servidor para registrar y sincronizar las dos bases de datos.

Resumen de los principales avances ya dichos:

-Aplicación Java.
-Implementación del Scene Builder.
-Servidor Socket TCP.
-Base de Datos SQL.
-Implementación método para la codificación de información AES.
-Aplicación Android.
-Base de Datos en la Nube SQL.
-Implementación del Framework Spring y servidor TomCat.

1.2. MOTIVACIÓN

Este proyecto tuvo lugar, por ayudar a una empresa, que requería de una aplicación que les facilitase el trabajo. Para poder realizar un trabajo a la altura,
como equipo debíamos investigar para poder presentar un producto óptimo y elaborado para que pudiese satisfacer las necesidades del cliente.

Nuestra motivación reside en, crear algo nuevo de un ámbito donde no estamos relacionados ni cómodos, tener que documentarnos por nosotros mismos, aprender cosas nuevas y comprobar nuestras capacidades.

Este proyecto hace que el equipo tenga que evolucionar, y así poder hacer cosas que de otra manera no habrían podido ser posibles, como por ejemplo, implementar TomCat con el Framework Spring, así como unir una base de datos en la nube con una base de datos local.

1.3. ANTECEDENTES

El contexto de trabajo se basa en una aplicación de escritorio para facilitar el trabajo de los administradores de la empresa del cliente.

Se realizó de manera en la que los miembros o bien, estaban reunidos para trabajar juntos, o bien, por separado manteniendo el contacto de manera online.

Como se trata de una aplicación diseñada expresamente como el cliente lo requirió, se puede decir, que es una aplicación original y única, así que realmente no sigue directrices de aplicaciones existentes, es una conglomeración de varias funciones, unidas en un único lugar, de esta manera, reduciendo el tiempo empleado para el trabajo, y además hacerlo más eficiente.

Previamente, usaban Access y bases de datos antiguas que hacían el trabajo de forma muy tediosa, teniendo que estar constantemente abriendo archivos Excel, bases de datos, así como el Access.

2. DESARROLLO DEL PROYECTO

2.1.1. Java

Definición:

Java es un lenguaje de programación orientado a objetos, hecho de manera que tenga tan pocas dependencias de implementación como fuere posible para no depender de dependencias externas.

La idea es que los desarrolladores puedan programar y puedan ejecutarlo en cualquier plataforma.

Java desde 2012, es el lenguaje más usado para programar, con especial predilección por las aplicaciones cliente-servidor.

El lenguaje Java fue creado por James Gosling, que trabajaba en Sun Microsystems, que más tarde sería engullida por Oracle en 2010.

La sintaxis se nutrió de C y C++, pero con menos utilidades de bajo nivel.
Las aplicaciones se rigen por clases Java, que pueden ser ejecutadas en cualquier máquina virtual de Java.

Características:

-Orientado a objetos

Está orientado a objetos ("OO"), se refiere a un método de programación y al diseño del lenguaje. Aunque hay muchas interpretaciones para OO, una primera idea es diseñar el software de forma que los distintos tipos de datos que usen estén unidos a sus operaciones. Así, los datos y el código (funciones o métodos) se combinan en entidades llamadas objetos. Un objeto puede verse como un paquete que contiene el "comportamiento" (el código) y el "estado" (datos). El principio es separar aquello que cambia de las cosas que permanecen inalterables. Frecuentemente, cambiar una estructura de datos implica un cambio en el código que opera sobre los mismos, o viceversa. Esta separación en objetos coherentes e independientes ofrece una base más estable para el diseño de un sistema software. El objetivo es hacer que grandes proyectos sean fáciles de gestionar y manejar, mejorando como consecuencia su calidad y reduciendo el número de proyectos fallidos.

Otra de las grandes promesas de la programación orientada a objetos es la creación de entidades más genéricas (objetos) que permitan la reutilización del software entre proyectos, una de las premisas fundamentales de la Ingeniería del Software. Un objeto genérico "cliente", por ejemplo, debería en teoría tener el mismo conjunto de comportamiento en diferentes proyectos, sobre todo cuando estos coinciden en cierta medida, algo que suele suceder en las grandes organizaciones. En este sentido, los objetos podrían verse como piezas reutilizables que pueden emplearse en múltiples proyectos distintos, posibilitando así a la industria del software construir proyectos de envergadura empleando componentes ya existentes y de comprobada calidad, conduciendo esto finalmente a una reducción drástica del tiempo de desarrollo. Podemos usar como ejemplo de objeto el aluminio. Una vez definidos datos (densidad, maleabilidad, etc.), y su "comportamiento" (soldar dos piezas, etc.), el objeto "aluminio" puede ser reutilizado en el campo de la construcción, del automóvil, de la aviación, etc.

La reutilización del software ha experimentado resultados dispares, encontrando dos dificultades principales: el diseño de objetos realmente genéricos es pobremente comprendido, y falta una

metodología para la amplia comunicación de oportunidades de reutilización. Algunas comunidades de "código abierto" (open source) quieren ayudar en este problema dando medios a los desarrolladores para diseminar la información sobre el uso y versatilidad de objetos reutilizables y bibliotecas de objetos.

-Independencia de la plataforma

La segunda característica, la independencia de la plataforma, significa que programas escritos en el lenguaje Java pueden ejecutarse igualmente en cualquier tipo de hardware. Este es el significado de ser capaz de escribir un programa una vez y que pueda ejecutarse en cualquier dispositivo, tal como reza el axioma de Java, "write once, run anywhere".

Para ello, se compila el código fuente escrito en lenguaje Java, para generar un código conocido como "bytecode" (específicamente Java bytecode), instrucciones máquina simplificadas específicas de la plataforma Java. Esta pieza está "a medio camino" entre el código fuente y el código máquina que entiende el dispositivo destino. El bytecode es ejecutado entonces en la máquina virtual (JVM), un programa escrito en código nativo de la plataforma destino (que es el que entiende su hardware), que interpreta y ejecuta el código. Además, se suministran bibliotecas adicionales para acceder a las características de cada dispositivo (como los gráficos, ejecución mediante hebras o threads, la interfaz de red) de forma unificada. Se debe tener presente que, aunque hay una etapa explícita de compilación, el bytecode generado es interpretado o convertido a instrucciones máquina del código nativo por el compilador JIT (Just In Time).

Hay implementaciones del compilador de Java que convierten el código fuente directamente en código objeto nativo, como GCJ. Esto elimina la etapa intermedia donde se genera el bytecode, pero la salida de este tipo de compiladores sólo puede ejecutarse en un tipo de arquitectura.

La licencia sobre Java de Sun insiste en que todas las implementaciones sean "compatibles". Esto dio lugar a una disputa legal entre Microsoft y Sun, cuando este último alegó que la implementación de Microsoft no daba soporte a las interfaces RMI y

JNI además de haber añadido características "dependientes" de su plataforma. Sun demandó a Microsoft y ganó por daños y perjuicios (unos 20 millones de dólares), así como una orden judicial forzando el acatamiento de la licencia de Sun. Como respuesta, Microsoft no ofrece Java con su versión de sistema operativo, y en recientes versiones de Windows, su navegador Internet Explorer no admite la ejecución de applets sin un conector (o plugin) aparte. Sin embargo, Sun y otras fuentes ofrecen versiones gratuitas para distintas versiones de Windows.

Las primeras implementaciones del lenguaje usaban una máquina virtual interpretada para conseguir la portabilidad. Sin embargo, el resultado eran programas que se ejecutaban comparativamente más lentos que aquellos escritos en C o C++. Esto hizo que Java se ganase una reputación de lento en rendimiento. Las implementaciones recientes de la JVM dan lugar a programas que se ejecutan considerablemente más rápido que las versiones antiguas, empleando diversas técnicas, aunque sigue siendo mucho más lentos que otros lenguajes.

La primera de estas técnicas es simplemente compilar directamente en código nativo como hacen los compiladores tradicionales, eliminando la etapa del bytecode. Esto da lugar a un gran rendimiento en la ejecución, pero tapa el camino a la portabilidad. Otra técnica, conocida como compilación JIT (Just In Time, o "compilación al vuelo"), convierte el bytecode a código nativo cuando se ejecuta la aplicación. Otras máquinas virtuales más sofisticadas usan una "recopilación dinámica" en la que la VM es capaz de analizar el comportamiento del programa en ejecución y recompila y optimiza las partes críticas. La recopilación dinámica puede lograr mayor grado de optimización que la compilación tradicional (o estática), ya que puede basar su trabajo en el conocimiento que de primera mano tiene sobre el entorno de ejecución y el conjunto de clases cargadas en memoria. La compilación JIT y la recopilación dinámica permiten a los programas Java aprovechar la velocidad de ejecución del código nativo sin por ello perder la ventaja de la portabilidad en ambos.

La portabilidad es técnicamente difícil de lograr, y el éxito de Java en ese campo ha sido dispar. Aunque es de hecho posible escribir programas para la plataforma Java que actúen de forma correcta en

múltiples plataformas de distinta arquitectura, el gran número de estas con pequeños errores o inconsistencias llevan a que a veces se parodie el eslogan de Sun, "Write once, run anywhere" como "Write once, debug everywhere.
El concepto de independencia de la plataforma de Java cuenta, sin embargo, con un gran éxito en las aplicaciones en el entorno del servidor, como los Servicios Web, los Servlets, los Java Beans, así como en sistemas empotrados basados en OSGi, usando entornos Java empotrados.

El recolector de basura

En Java el problema de fugas de memoria se evita en gran medida gracias a la recolección de basura (o automatic garbage collector). El programador determina cuándo se crean los objetos, y el entorno, en tiempo de ejecución de Java (Java runtime), es el responsable de gestionar el ciclo de vida de los objetos.
El programa, u otros objetos, pueden tener localizado un objeto mediante una referencia a éste. Cuando no quedan referencias a un objeto, el recolector de basura de Java borra el objeto, liberando así la memoria que ocupaba previniendo posibles fugas (ejemplo: un objeto creado y únicamente usado dentro de un método sólo tiene entidad dentro de éste; al salir del método el objeto es eliminado). Aun así, es posible que se produzcan fugas de memoria si el código almacena referencias a objetos que ya no son necesarios; es decir, pueden aún ocurrir, pero en un nivel conceptual superior. En definitiva, el recolector de basura de Java permite una fácil creación y eliminación de objetos y mayor seguridad.

2.1.2. Spring Framework

Spring es un framework para el desarrollo de aplicaciones y contenedor de inversión de control, de código abierto para la plataforma Java.

La primera versión fue escrita por Rod Johnson, quien lo lanzó junto a la publicación de su libro Expert One-on-One J2EE Design and Development (Wrox Press, octubre 2002). El framework fue lanzado inicialmente bajo la licencia Apache 2.0 en junio de 2003. El primer gran lanzamiento fue la versión 1.0, que apareció en marzo de 2004 y fue seguida por otros hitos en septiembre de 2004 y marzo de 2005. La versión 1.2 de Spring Framework obtuvo reconocimientos Jolt Awards y Jax Innovation Awards en 2006, Spring Framework 2.0 fue lanzada en 2006, la versión 2.5 en noviembre de 2007, Spring 3.0 en diciembre de 2009, y Spring 3.1 dos años más tarde. El inicio del desarrollo de la versión 4.0 fue anunciado en enero de 2013. La versión actual es la 5.0.

Si bien las características fundamentales de Spring Framework pueden ser usadas en cualquier aplicación desarrollada en Java, existen variadas extensiones para la construcción de aplicaciones web sobre la plataforma Java EE. A pesar de que no impone ningún modelo de programación en particular, este framework se ha vuelto popular en la comunidad al ser considerado una alternativa, sustituto, e incluso un complemento al modelo EJB (Enterprise JavaBean).

Módulos:

Contenedor de inversión de control: permite la configuración de los componentes de aplicación y la administración del ciclo de vida de los objetos Java, se lleva a cabo principalmente a través de la inyección de dependencias.

Programación orientada a aspectos: habilita la implementación de rutinas transversales.

Acceso a datos: se trabaja con RDBMS en la plataforma java, usando Java Database Connectivity y herramientas de Mapeo objeto relacional con bases de datos NoSQL.

Gestión de transacciones: unifica distintas APIs de gestión y coordina las transacciones para los objetos Java.

Modelo vista controlador: Un framework basado en HTTP y servlets, que provee herramientas para la extensión y personalización de aplicaciones web y servicios web REST.

Framework de acceso remoto: Permite la importación y exportación estilo RPC, de objetos Java a través de redes que soporten RMI, CORBA y protocolos basados en HTTP incluyendo servicios web (SOAP).

Convención sobre Configuración: el módulo Spring Roo ofrece una solución rápida para el desarrollo de aplicaciones basadas en Spring Framework, privilegiando la simplicidad sin perder flexibilidad.

Procesamiento por lotes: un framework para procesamiento de mucho volumen que como características incluye funciones de registro/trazado, manejo de transacciones, estadísticas de procesamiento de tareas, reinicio de tareas, y manejo de recursos.

Autenticación y Autorización: procesos de seguridad configurables que soportan un rango de estándares, protocolos, herramientas y prácticas a través del subproyecto Spring Security (antiguamente Acegi).

Administración Remota: Configuración de visibilidad y gestión de objetos Java para la configuración local o remota vía JMX.

Mensajes: Registro configurable de objetos receptores de mensajes, para el consumo transparente desde la a través de JMS, una mejora del envío de mensajes sobre las API JMS estándar.

Testing: Soporte de clases para desarrollo de unidades de prueba e integración.

El corazón de Spring Framework es su Contenedor de inversión de control (IoC). Su trabajo es instanciar, inicializar y conectar objetos de la aplicación, además de proveer una serie de características adicionales disponibles en Spring a través del tiempo de vida de los objetos.

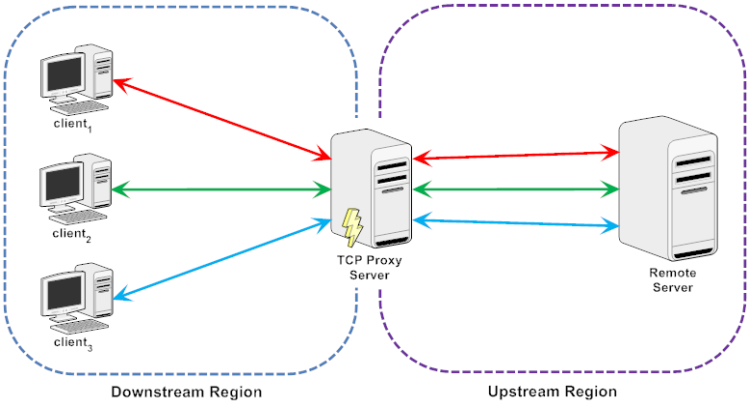
Los objetos creados y gestionados por el Contenedor se denominan objetos gestionados o beans. Estos objetos son objetos tipo POJO. Para realizar su tarea el contenedor necesita información indicando como instanciar y conectar entre sí los beans. A esta información se la llama metadatos de configuración. Hay distintas formas de proporcionar esta información: basándose en XML, basándose en anotaciones o basándose en objetos Java (desde Spring 3.0). El contenedor es independiente del formato de los metadatos de configuración. El usuario puede usar el formato que desee e incluso mezclarlos en la misma aplicación.

Los objetos pueden ser obtenidos por búsqueda de dependencias o por inyección de dependencias. Búsqueda de dependencias es un modelo donde se pide al objeto contenedor un objeto con un nombre específico o de un tipo específico. Inyección de dependencias es un modelo en el que el contenedor pasa objetos por nombre a otros objetos, ya sea a través de métodos constructores, propiedades, o métodos de la fábrica.

En muchos casos cuando se utilizan otras partes del Spring Framework no necesita utilizar el Contenedor, aunque probablemente su uso le permita hacer una aplicación más fácil de configurar y personalizar. El Contenedor de Spring le proporciona un mecanismo consistente para configurar las aplicaciones, y se integra con casi todos los entornos Java, desde aplicaciones de pequeñas a grandes aplicaciones empresariales.

El contenedor se puede convertir en un contenedor EJB 3.0 parcialmente por medio del proyecto Pitchfork. Algunos critican al Spring Framework por no cumplir los estándares. Sin embargo, SpringSource no ve el cumplimiento EJB 3 como un objetivo importante, y afirma que el Spring Framework y el contenedor permiten modelos de programación más potentes. No creas un objeto, sino describes la forma en que deben crearse, definiéndolo en el archivo de configuración de Spring. No llamas a los servicios y componentes, sino dices que servicios y componentes deben ser llamados, definiéndolos en los archivos de configuración de Spring. Esto hace el código fácil de mantener y más fácil de probar mediante la Inyección de Dependencia (IoC).

2.1.3. Server Socket TCP



Protocolo de control de transmisión (en inglés Transmission Control Protocol o TCP) es uno de los protocolos fundamentales en Internet. Fue creado entre los años 1973 y 1974 por Vint Cerf y Robert Kahn.

Muchos programas dentro de una red de datos compuesta por redes de computadoras pueden usar TCP para crear "conexiones" entre sí, a través de las cuales puede enviarse un flujo de datos. El protocolo garantiza que los datos serán entregados en su destino sin errores y en el mismo orden en que se transmitieron. También proporciona un mecanismo para distinguir distintas aplicaciones dentro de una misma máquina, a través del concepto de puerto.

TCP da soporte a muchas de las aplicaciones más populares de Internet (navegadores, intercambio de ficheros, clientes FTP, etc.) y protocolos de aplicación HTTP, SMTP, SSH y FTP.

-Objetivos:

Con el uso de protocolo TCP, las aplicaciones pueden comunicarse en forma segura (gracias al de acuse de recibo -ACK- del protocolo TCP) independientemente de las capas inferiores. Esto significa que los routers (que funcionan en la capa de red) sólo tienen que enviar los datos en forma de segmentos, sin preocuparse con el monitoreo de datos porque esta función la cumple la capa de transporte (o más específicamente el protocolo TCP).

-Funciones de TCP

En la pila de protocolos TCP/IP, TCP es la capa intermedia entre el protocolo de red (IP) y la aplicación. Muchas veces las aplicaciones necesitan que la comunicación a través de la red sea confiable. Para ello se implementa el protocolo TCP que asegura que los datos que emite el cliente sean recibidos por el servidor sin errores y en el

mismo orden que fueron emitidos, a pesar de trabajar con los servicios de la capa IP, la cual no es confiable. Es un protocolo orientado a la conexión, ya que el cliente y el servidor deben anunciarse y aceptar la conexión antes de comenzar a transmitir los datos a ese usuario que debe recibirlos.

-Características del TCP

Permite colocar los segmentos nuevamente en orden cuando vienen del protocolo IP.

Permite el monitoreo del flujo de los datos y así evitar la saturación de la red.

Permite que los datos se formen en segmentos de longitud variada para "entregarlos" al protocolo IP.

Permite multiplexar los datos, es decir, que la información que viene de diferentes fuentes (por ejemplo, aplicaciones) en la misma línea pueda circular simultáneamente.

Por último, permite comenzar y finalizar la comunicación amablemente.

-Formato de los segmentos TCP

En el nivel de transporte, los paquetes de bits que constituyen las unidades de datos de protocolo TCP se llaman "segmentos". El formato de los segmentos TCP se muestra en el esquema segmento TCP.

2.1.4. JavaFX

JavaFX es una familia de productos y tecnologías de Oracle Corporation (inicialmente Sun Microsystems), para la creación de Rich Internet Applications (RIAs), esto es, aplicaciones web que tienen las características y capacidades de aplicaciones de escritorio, incluyendo aplicaciones multimedia interactivas. Las tecnologías incluidas bajo la denominación JavaFX son JavaFX Script y JavaFX Mobile, aunque hay más productos JavaFX planeados.

Las aplicaciones JavaFX pueden ser ejecutadas en una amplia variedad de dispositivos. En su versión (JavaFX 1.3, abril 2010) permite crear aplicaciones de escritorio, para celulares, la Web, TV, consolas de videojuegos, reproductores Blu-ray, entre otras plataformas planeadas. En octubre de 2011 fue lanzada la versión 2.0. Para el desarrollo de aplicaciones JavaFX un lenguaje declarativo, tipado llamado JavaFX Script, además puede integrarse código Java en programas JavaFX. JavaFX es compilado a código Java, por lo que las aplicaciones JavaFX pueden ser ejecutadas en computadores con la máquina virtual de Java instalada (JRE), o celulares corriendo Java ME.

JavaFX fue anunciado en la de desarrolladores JavaOne en mayo de 2007 y liberado en diciembre de 2008.

La intención de Sun Microsystems respecto de JavaFX era competir en el espacio que ya ocupan Flash de Adobe, y Silverlight de Microsoft.

En palabras de James Gosling "La mayoría de los lenguajes de script están orientados a las páginas web; éste está orientado a las interfaces que son altamente animadas".

2.1.5. Scene Builder

JavaFX Scene Builder es un programa el cual te permite hacer un entorno para JavaFX, sin la necesidad de escribir ningún código. Permite posicionar el interfaz gráfico (GUI), mediante el "drag-and-drop", en un "scene" de JavaFX. Según vas construyendo tu "scene", el código "FXML", se va autogenerando.

JavaFX Scene Builder tiene un interfaz fácil e intuitivo que incluso la gente que no sabe programar puede hacer un prototipo de sus aplicaciones que se conectan por GUI a la aplicación lógica.

Sus principales funciones son:

Una interfaz "drag-and-drop" WYSIWYG que te permite crear un layout GUI sin la necesidad de escribir código. Puedes agregar, combinar, y editar los controles de tu layout de JavaFX GUI, usando la librería de controles de GUI y el "content pane".

El código de FXML se genera automáticamente según vas modificando el layout GUI.

El código FXML se almacena en un archivo diferente al de la aplicación lógica y los archivo "style sheet".

La edición y la prevista de las funciones, te permiten visualizar de forma rápida los cambios de layout del GUI, que haces sin necesidad de compilar. Estas funciones ayudan a minimizar el tiempo para poder realizar tu aplicación. También puedes asignar un CSS a tu layout GUI y previsualizar el resultado de este.

Tienes acceso total a la librería de controles de JavaFX GUI. Para poder ver la lista de los componentes de JavaFX 8 GUI, puedes

escribir en el panel de búsqueda FX8. La lista incluye los componentes “TreeTableView”, “DatePicker”, y “SwingNode”.

La habilidad de agregar componentes personalizados a la librería es posible. La librería de los componentes GUI disponibles puede ser extendida por medio de la importación de componentes GUI a través de JAR files, archivos FXML, o agregándolos desde Hierarchy o Content panel.

Hay soporte a 3D, FXML pueden contener objetos 3D, y ser guardados y cargados en la herramienta Scene Builder 2.0. Puedes ver y editar propiedades de objetos 3D usando Inspector panel. Pero por contraparte no puedes crear nuevos objetos 3D con Scene Builder.

Support for Rich Text has been added. A new container, TextFlow, is now available in the Library of GUI components. You can drag multiple text nodes and other types of nodes, into the a TextFlow container. You can also directly manipulate the text nodes to re-arrange them in the container. Inline and property editing features are also available for each text node.

El Kit de JavaFX Scene Builder está provisto con Scene Builder 2.0. El kit es un API que te permite la integración de Scene Builder directamente con la GUI de otra aplicación, como Eclipse.

El soporte CSS permite manejo flexible de la forma en la que se ve la UI de tu aplicación.

Tiene soporte en los sistemas operativos de Windows, Linux y Mac OS X.

2.1.6. Base de datos SQL

Definición:

MySQL es un sistema de gestión de bases de datos relacional desarrollado bajo licencia dual: Licencia pública general/Licencia comercial por Oracle Corporation y está considerada como la base datos de código abierto más popular del mundo, y una de las más populares en general junto a Oracle y Microsoft SQL Server, sobre todo para entornos de desarrollo web.

MySQL fue inicialmente desarrollado por MySQL AB (empresa fundada por David Axmark, Allan Larsson y Michael Widenius). MySQL AB fue adquirida por Sun Microsystems en 2008, y ésta a su vez fue comprada por Oracle Corporation en 2010, la cual ya era dueña desde 2005 de Innobase Oy, empresa finlandesa desarrolladora del motor InnoDB para MySQL.

Al contrario de proyectos como Apache, donde el software es desarrollado por una comunidad pública y los derechos de autor del código están en poder del autor individual, MySQL es patrocinado por una empresa privada, que posee el copyright de la mayor parte del código. Esto es lo que posibilita el esquema de doble licenciamiento anteriormente mencionado. La base de datos se distribuye en varias versiones, una Community, distribuida bajo la Licencia pública general de GNU, versión 2, y varias versiones Enterprise, para aquellas empresas que quieran incorporarlo en productos privativos. Las versiones Enterprise incluyen productos o servicios adicionales tales como herramientas de monitorización y asistencia técnica oficial. En 2009 se creó un fork denominado MariaDB por algunos desarrolladores (incluido algunos desarrolladores originales de MySQL) descontentos con el modelo de desarrollo y el hecho de que una misma empresa controle a la vez los productos MySQL y Oracle Database.

Está desarrollado en su mayor parte en ANSI C y C++. Tradicionalmente se considera uno de los cuatro componentes de la pila de desarrollo LAMP y WAMP.

MySQL es usado por muchos sitios web grandes y populares, como Wikipedia, Google (aunque no para búsquedas), Facebook, Twitter, Flickr, y YouTube.

Aplicaciones:

MySQL es muy utilizado en aplicaciones web, como Joomla, Wordpress, Drupal o phpBB, en plataformas (Linux/Windows-Apache-MySQL-PHP/Perl/Python), y por herramientas de seguimiento de errores como Bugzilla. Su popularidad como aplicación web está muy ligada a PHP, que a menudo aparece en combinación con MySQL.

MySQL es una base de datos muy rápida en la lectura cuando utiliza el motor no transaccional MyISAM, pero puede provocar problemas de integridad en entornos de alta concurrencia en la modificación. En aplicaciones web hay baja concurrencia en la modificación de datos y en cambio el entorno es intensivo en lectura de datos, lo que hace a MySQL ideal para este tipo de aplicaciones. Sea cual sea el entorno en el que va a utilizar MySQL, es importante monitorizar de antemano el rendimiento para detectar y corregir errores tanto de SQL como de programación.

Características:

Inicialmente, MySQL carecía de elementos considerados esenciales en las bases de datos relacionales, tales como integridad referencial y transacciones. A pesar de ello, atrajo a los desarrolladores de páginas web con contenido dinámico, justamente por su simplicidad.

Poco a poco los elementos de los que carecía MySQL están siendo incorporados tanto por desarrollos internos, como por desarrolladores de software libre. Entre las características disponibles en las últimas versiones se puede destacar:

Amplio subconjunto del lenguaje SQL. Algunas extensiones son incluidas igualmente.

Disponibilidad en gran cantidad de plataformas y sistemas.

Posibilidad de selección de mecanismos de almacenamiento que ofrecen diferentes velocidades de operación, soporte físico, capacidad, distribución geográfica, transacciones...

Transacciones y claves foráneas.

Conectividad segura.

Replicación.

Búsqueda e indexación de campos de texto.

MySQL es un sistema de administración relacional de bases de datos. Una base de datos relacional archiva datos en tablas separadas en vez de colocar todos los datos en un gran archivo. Esto permite velocidad y flexibilidad. Las tablas están conectadas por relaciones definidas que hacen posible combinar datos de diferentes tablas sobre pedido.

MySQL es software de fuente abierta. Fuente abierta significa que es posible para cualquier persona usarlo y modificarlo. Cualquier persona puede bajar el código fuente de MySQL y usarlo sin pagar. Cualquier interesado puede estudiar el código fuente y ajustarlo a sus necesidades. MySQL usa el GPL (GNU General Public License) para definir qué puede hacer y qué no puede hacer con el software en diferentes situaciones. Si usted no se ajusta al GPL o requiere introducir código MySQL en aplicaciones comerciales, usted puede comprar una versión comercial licenciada.

2.1.7. Git Hub

GitHub es una plataforma de desarrollo colaborativo de software para alojar proyectos utilizando el sistema de control de versiones Git.

GitHub aloja tu repositorio de código y te brinda herramientas muy útiles para el trabajo en equipo, dentro de un proyecto.

Además de eso, puedes contribuir a mejorar el software de los demás. Para poder alcanzar esta meta, GitHub provee de funcionalidades para hacer un fork y solicitar pulls.

Realizar un fork es simplemente clonar un repositorio ajeno (genera una copia en tu cuenta), para eliminar algún bug o modificar cosas de él. Una vez realizadas tus modificaciones puedes enviar un pull al dueño del proyecto. Éste podrá analizar los cambios que has realizado fácilmente, y si considera interesante tu contribución, adjuntarlo con el repositorio original.

En la actualidad, GitHub es mucho más que un servicio de alojamiento de código. Además de éste, se ofrecen varias herramientas útiles para el trabajo en equipo. Entre ellas, caben destacar:

Una wiki para el mantenimiento de las distintas versiones de las páginas.

Un sistema de seguimiento de problemas que permiten a los miembros de tu equipo detallar un problema con tu software o una sugerencia que deseen hacer.

Una herramienta de revisión de código, donde se pueden añadir anotaciones en cualquier punto de un fichero y debatir sobre determinados cambios realizados en un commit específico.

Un visor de ramas donde se pueden comparar los progresos realizados en las distintas ramas de nuestro repositorio.

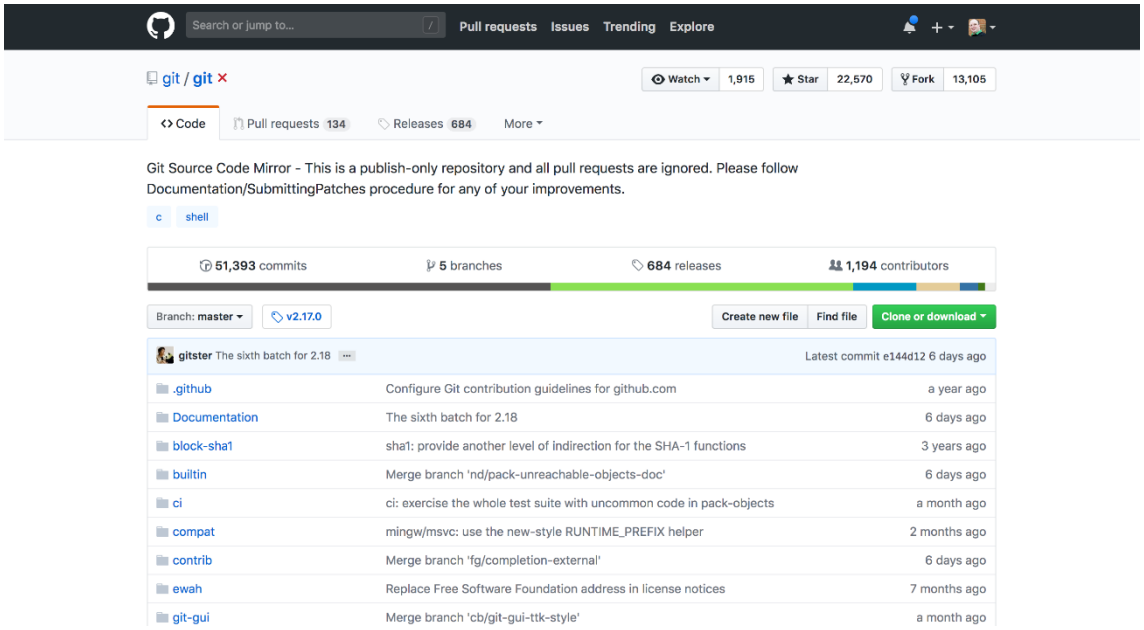
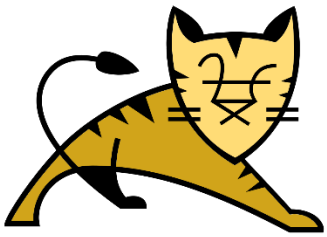


Figura: Página web de Git Hub.

2.1.8 Apache TomCat



Tomcat es un contenedor de servlets que se utiliza en la Referencia oficial de la implementación para Java Servlet y JavaServer Pages (JSP). Las especificaciones Java Servlet y JavaServer Pages son desarrolladas por Sun Microsystems cuyas especificaciones vienen dadas por la JCP (Java Community Process). Apache Tomcat es desarrollado en un entorno abierto y participatorio, bajo la licencia de Apache Software License.

Para simplificar, podríamos decir que Apache Tomcat (o Jakarta Tomcat) es un software desarrollado con Java (con lo cual puede funcionar en cualquier sistema operativo, con su máquina virtual java correspondiente) que sirve como servidor web con soporte de servlets y JSPs.

Tomcat es mantenido y desarrollado por miembros de la Apache Software Foundation y voluntarios independientes. Los usuarios disponen de libre acceso a su código fuente y a su forma binaria en los términos establecidos en la Apache Software License. Las primeras distribuciones de Tomcat fueron las versiones 3.0.x. Las versiones más recientes son las 7.x, que implementan las especificaciones de Servlet 3.0 y de JSP 2.2. A partir de la versión 4.0, Jakarta Tomcat utiliza el contenedor de servlets Catalina.

Tomcat es un servidor web con soporte de servlets y JSPs. Tomcat no es un servidor de aplicaciones, como JBoss o JOnAS. Incluye el compilador Jasper, que compila JSPs convirtiéndolas en servlets. El motor de servlets de Tomcat a menudo se presenta en combinación con el servidor web Apache.

Tomcat puede funcionar como servidor web por sí mismo. En sus inicios existió la percepción de que el uso de Tomcat de forma autónoma era sólo recomendable para entornos de desarrollo y entornos con requisitos mínimos de velocidad y gestión de transacciones. Hoy en día ya no existe esa percepción y Tomcat es usado como servidor web autónomo en entornos con alto nivel de tráfico y alta disponibilidad.

La jerarquía de directorios de instalación de Tomcat incluye:

- bin - arranque, cierre, y otros scripts y ejecutables.
- common - clases comunes que pueden utilizar Catalina y las aplicaciones web.
- conf - ficheros XML y los correspondientes DTD para la configuración de Tomcat.
- logs - logs de Catalina y de las aplicaciones.
- server - clases utilizadas solamente por Catalina.
- shared - clases compartidas por todas las aplicaciones web.
- webapps - directorio que contiene las aplicaciones web.
- work - almacenamiento temporal de ficheros y directorios.

2.2. PLANIFICACIÓN

El equipo está organizado con E-mail, Whatsapp y Git Hub.

En Trello ha sido usado para repartir las tareas que cada miembro del equipo tenía que hacer.

El Whatsapp hace que la comunicación sea mucho más rápida y fluida, por eso mismo ha sido usada esta aplicación de móvil.

Por medio de E-mail, se transmiten nuevos conocimientos, archivos modificados, en general, pequeños avances, que aún no son definitivos y no merecen ser subidos a Git Hub.

El Git Hub es donde ser guarda el proyecto en sí, para que según vaya avanzando, cualquiera de los miembros del proyecto, pueda acceder a la última versión del proyecto de manera más inmediata.

2.3. DESCRIPCIÓN DEL TRABAJO REALIZADO

La descripción estará dividida en dos partes, la parte del entorno gráfico, y la programable.

Entorno gráfico:

Se empezó por decidir el estilo de login se quería hacer, la forma que debía tomar, las fuentes de color y la fuente de escritura.

La fuente de escritura por la que se optó fue la que el propio cliente, así como las fuentes de color, que ya se verán más adelante.

Las fuentes de color usadas han sido las siguientes:

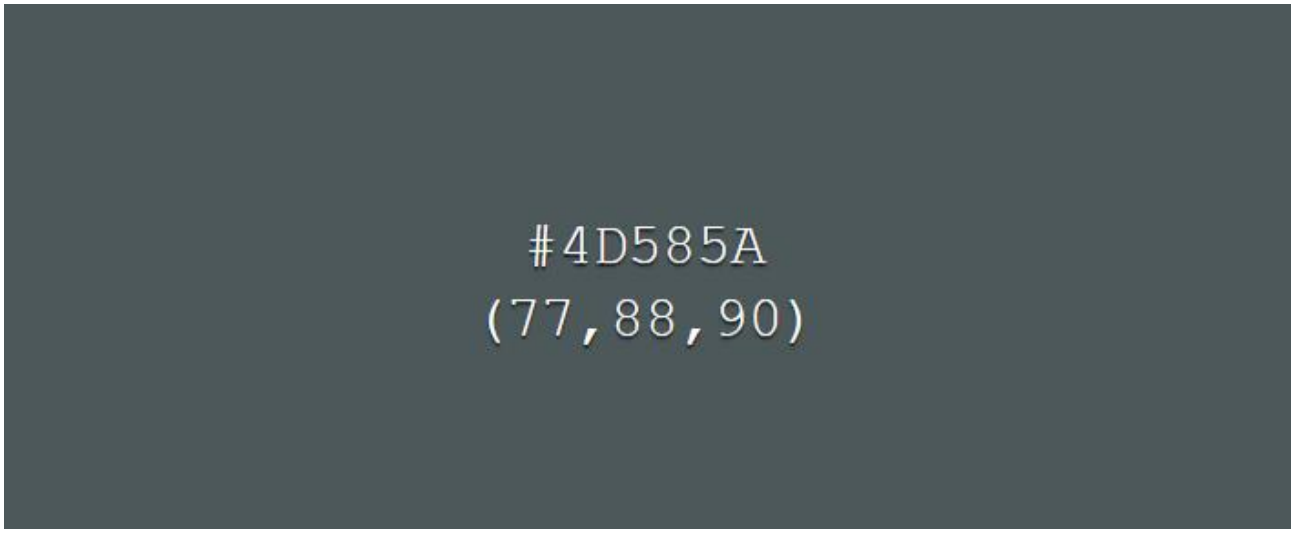


Figura: color primario de la aplicación

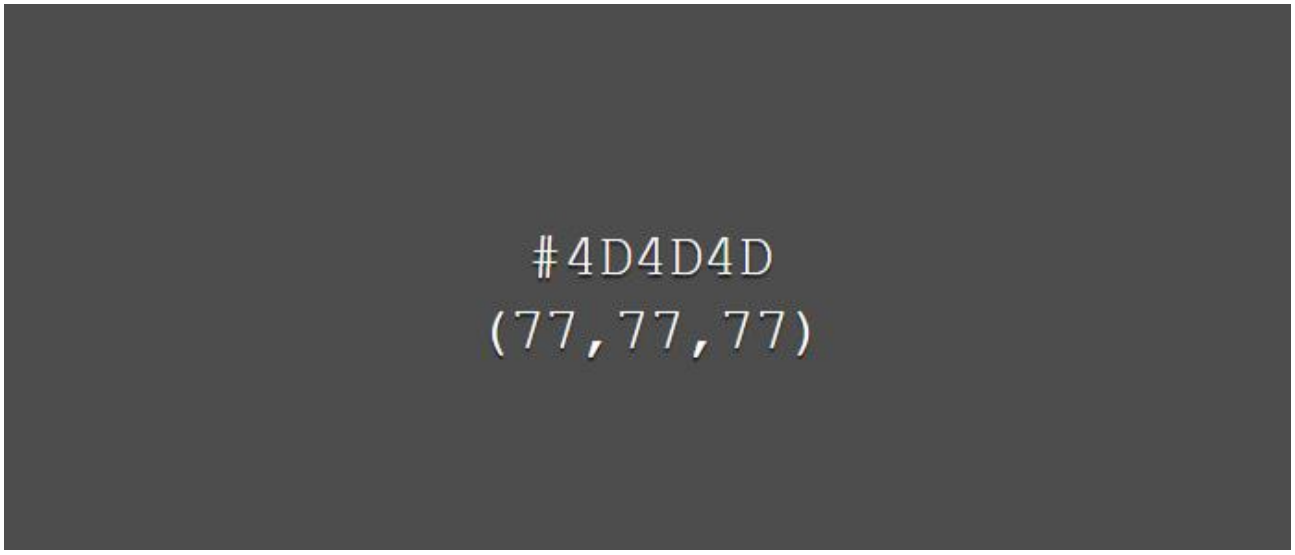


Figura: color usado para los textos de la pestaña Datos y Añadir.

#FFFFFF
(255,255,255)

Figura: color usado para los textos de botones, y texto del login.

Las primeras impresiones de cómo se vería la aplicación fueron las siguientes,
realizadas con la aplicación online "Moqup", posteriormente cambiarían algo,
debido a que crear visualizaciones tiene sus limitaciones, como no poder realizar "Splash Arts" (se refiere a efectos de visualización), o "Fade Transitions" (movimiento de imágenes, de objetos, etc), los cuales deberían ser introducidos directamente en el entorno de trabajo con el Scene Builder. Las siguientes imágenes son las realizadas en el "Moqup".

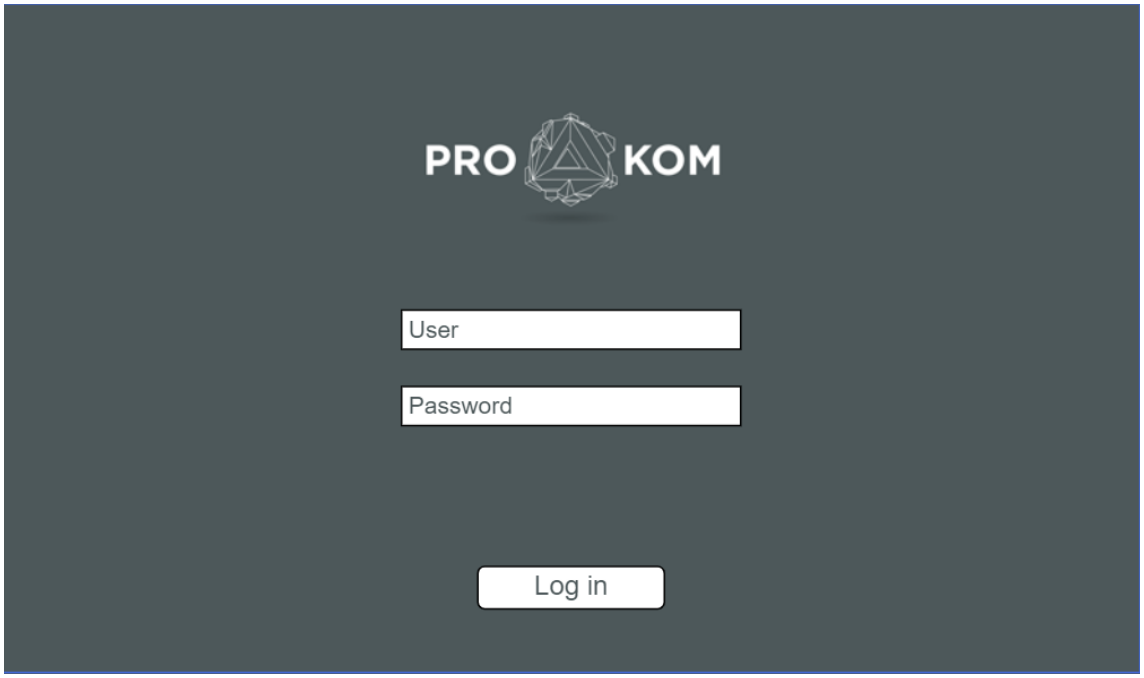


Figura: Imagen del mockup de la pestaña Inicio.

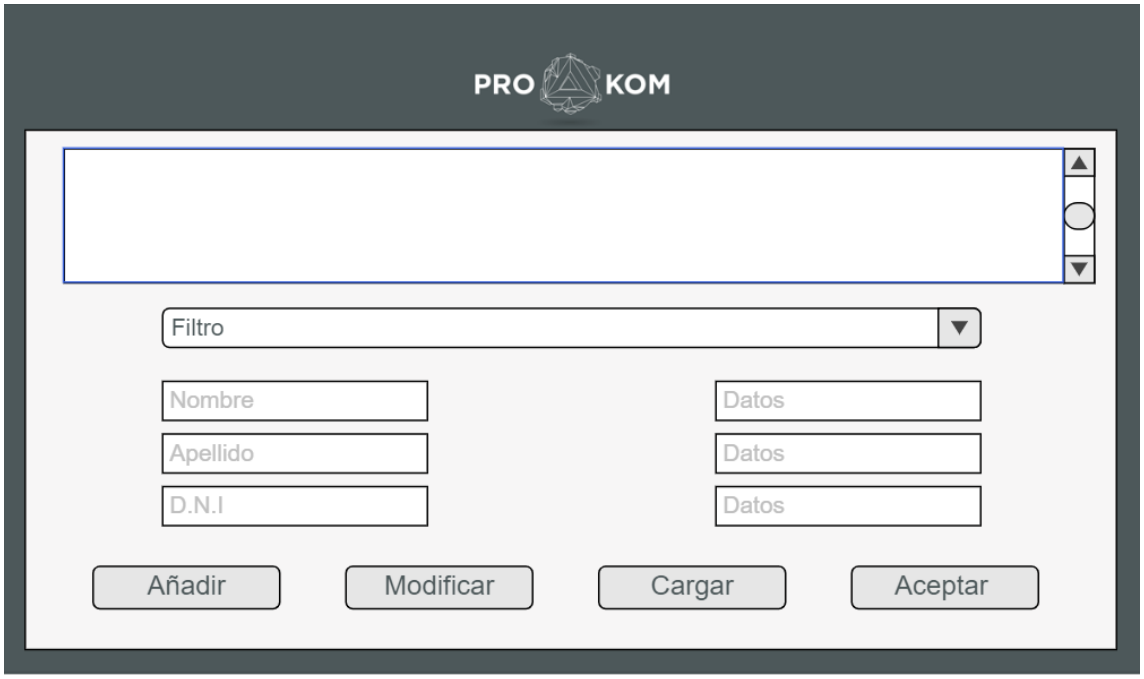
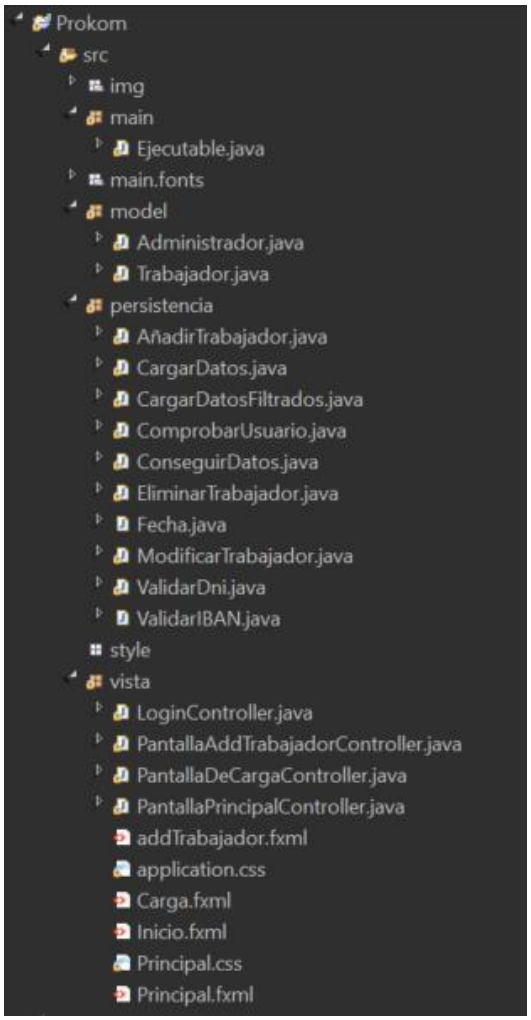


Figura: Imagen del mockup de la pestaña de Datos.

Entorno lógico:

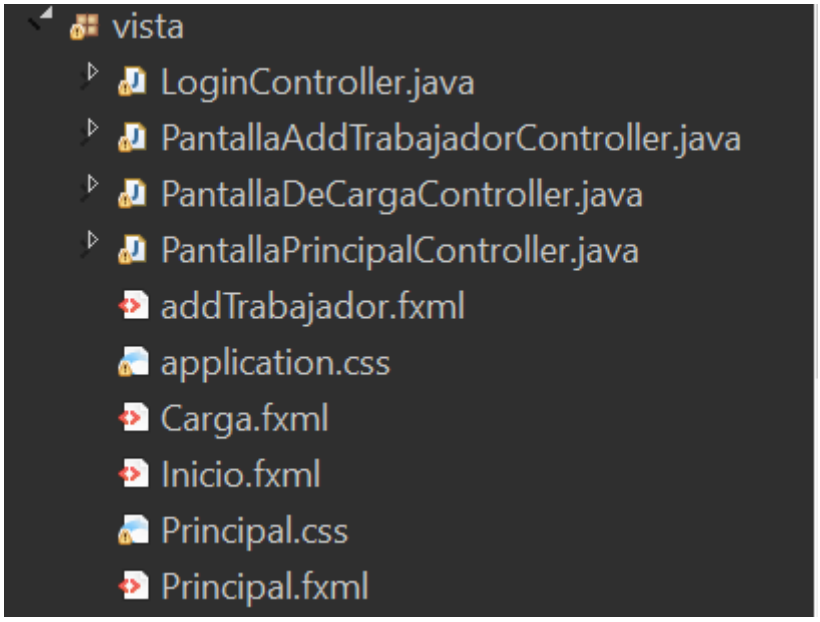
La aplicación se divide principalmente en dos bloques, uno que es la aplicación del cliente y otro que la aplicación el servidor.

La aplicación del Cliente será la que utilizan los empleados.



Se dividirá el contenido de la aplicación en varias carpetas, las imágenes y las fuentes de escritura tendrán sus propias carpetas.

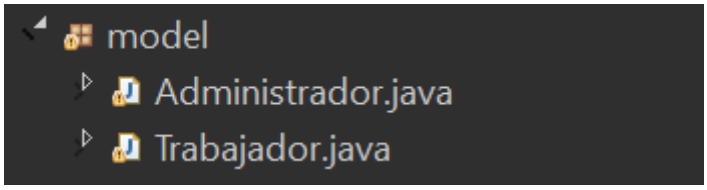
En un principio los CSS también tenían su propia carpeta pero tras ciertos fallos con la interfaz de Scene Builder juntamos los controladores, las vistas FXML y las páginas de estilos CSS.



Hay 4 vistas con sus 4 controladores para gestionar sus métodos, además de tener dos páginas de estilo CSS que trabajarán separadas.

Application.css → Carga.fxml

Principal.css →Principal.fxml, addTrabajador.fxml

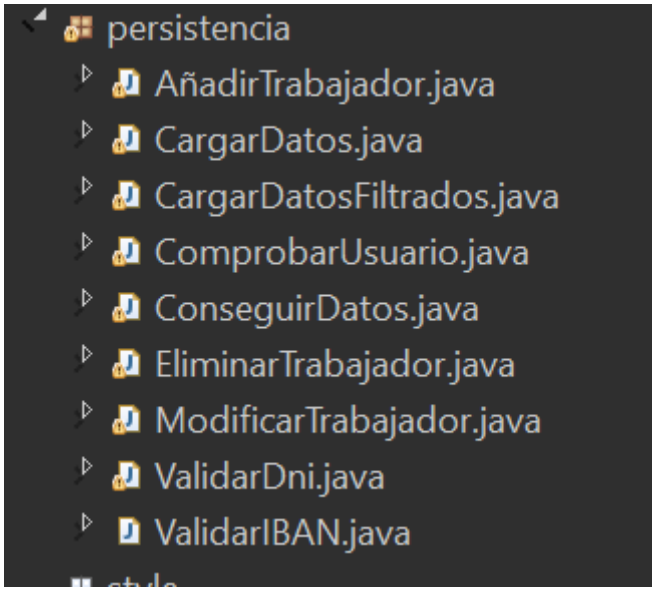


Los Constructores se almacenarán en el paquete model, y su gran distinción es haberle dado otro trato a sus variables para que sean más manejables con el FXJava.

```
public class Administrador implements Serializable{
    private final StringProperty  usuario;
    private final StringProperty  password;

    public Administrador() {
        this(null, null);
    }
    public Administrador(String usuario, String password) {
        super();
        this.usuario = new SimpleStringProperty (usuario);
        this.password = new SimpleStringProperty(password);
    }
```

Lo declaramos como tipo StringProperty para que sea más fácil manejarlo y variamos los gets sets, y contructor originales para poder utilizar esta forma de constructor.

En el paquete de la pesistencia están todos los métodos que se van a utilizar en la aplicación y van a ser llamados normalmente por el controlador.



Los métodos menos importantes son ValidarDni y ValidarIBan ya que son solo para comprobar los campos, y los más importantes son las sentencias que irán asociados a los botones como AñadirTrabajador.

Estos métodos funcionan en conjunto, si por ejemplo si se quisiese añadir un usuario, estos son los caminos que seguirá para crearlo.

Lo primero será conseguir los datos de los campos para poder añadirlos al constructor y enviar el constructor

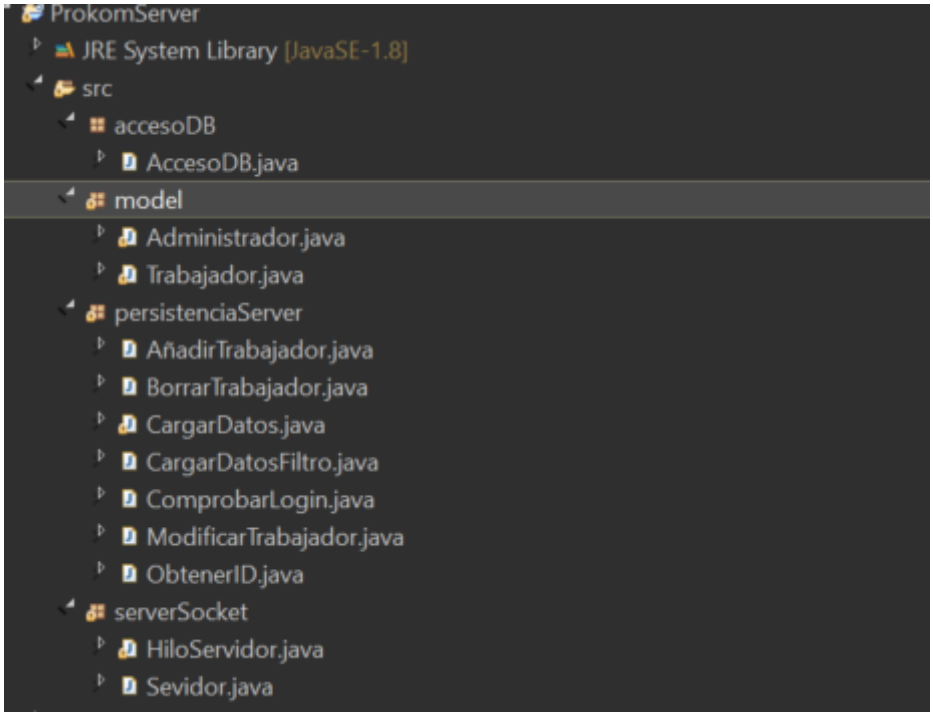
```
public  Trabajador conseguirDatos() {
    Trabajador trabajador = null;

    String dni = dniLabel.getText();

    if (nombreLabel.getText().equals("") || apellido1Label.getText().equals("")
            || apellido2Label.getText().equals("") ||  DomicilioLabel.getText().equals
            || domicilioSocialLabel.getText().equals("")|| nacionalidadLabel.getText()
            || paisLabel.getText().equals("") || MunicipioLabel.getText().equals("")
            || codigoPostalLabel.getText().equals("")|| numeroSeguridadLabel.getText()

        JOptionPane.showMessageDialog(null, "Tienes que rellenas los campos");
    } else {
        ValidarDni Vdni = new ValidarDni(dni);
        if(Vdni.validar() == true) {
                try {
                    trabajador = new Trabajador(dni,nombreLabel.getText()
                            , apellido1Label.getText(),apellido2Label.getText(), Domic
                            , domicilioSocialLabel.getText(), nacionalidadLabel.getTex
                            , Integer.parseInt(codigoPostalLabel.getText()), Integer.p
                } catch (NumberFormatException e) {
                    JOptionPane.showMessageDialog(null, "Error, el codigo postal y el
                }
            }
        }
    return trabajador;
```

Lo siguiente es llamarlo desde el método de Añadir Trabajador, lo normal sería llamarlo y añadirlo a la base de datos pero aquí empieza lo complicado ya que se va a enviar todos estos datos a la base de datos y ya allí se guarde en la base de datos

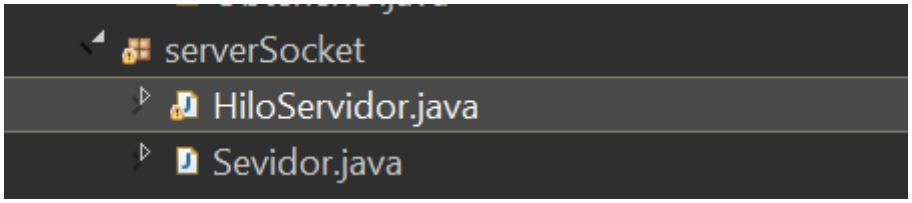
```
ublic class AñadirTrabajador {
    public void AñadirTrabajador() throws ClassNotFoundException, IOException {
        Trabajador trabajador = null;
        ConseguirDatos princi = new ConseguirDatos();
        boolean codIncorrecto;
        int codigo;
        String host = "localhost";
        int Puerto = 60000; // puerto remoto
        System.out.println("PROGRAMA CLIENTE INICIADO...");
        Socket cliente;
        try {
            cliente = new Socket(host, Puerto);
            // CREO FLUJO DE SALIDA AL SERVIDOR

            DataOutputStream flujoSalida = new DataOutputStream(cliente.getOutputStream());

            // CREO FLUJO DE ENTRADA AL SERVIDOR

            ObjectOutputStream outObjeto = new ObjectOutputStream(
                    cliente.getOutputStream());

            codIncorrecto = true;
            codigo = 1;
                trabajador = princi.conseguirDatos();
                outObjeto.writeObject(trabajador);
                flujoSalida.writeInt(codigo);

                outObjeto.close();
                flujoSalida.close();
                cliente.close();

                if (codigo == 0){
                    System.out.println("ERROR, FILTRO SERVIDOR");
                } else {
                    codIncorrecto = false;
                }



        } catch (IOException e1) {
            // TODO Auto-generated catch block
            e1.printStackTrace();
        }

    }
```

Se creara la conexión dándole un puerto y se creara un flujo de salida donde almacenaremos un código que necesitaremos para darle la instrucción en el servidor junto al objeto almacenado y enviarlo al servidor.

El servidor La aplicación que mediante el Server Socket TCP recibirá varios parámetros pero normalmente recibe el código que definirá que tiene que hacer en el servidor y el objeto.



El Servidor también necesita un paquete de model para poder utilizar los constructores, y una de accesoDB que tendrá la clase que nos hará poder conectarnos a la base de datos.



El ServerSocket tendrá las aplicaciones Servidor que permitirá recibir los datos en la aplicación como servidor y además HiloServidor que gestionará todos los datos que llegan al servidor.

Cómo funciona el servidor.

1º Se abre el flujo de entrada para sacar el código y dependiendo del código se le redirige hasta el método indicado y allí haga sus funciones.

```
try {

    flujoEntradaObj = new ObjectInputStream(socket.getInputStream());

    // EL CLIENTE ME ENVÍA UN MENSAJE

    int codigo = flujoEntrada.readInt();
    System.out.println("Recibiendo del Trabajador: \n\t" + codigo);
    if (codigo == 1)
        AddTrabajador(socket, flujoEntradaObj);
    else if (codigo == 2)
        ModificarTrabajador(socket, flujoEntradaObj);
    else if (codigo == 3)
        BorrarTrabajador(socket, flujoEntradaObj);
    else if (codigo == 4)
        CargarTrabajador(socket, flujoSalidaObj);
    else if (codigo == 5)
        CargarTrabajadorFiltrado(socket, flujoEntrada, flujoSalidaObj);
    else if (codigo == 6)
        ComprobarUsuario(socket, flujoEntradaObj, flujoSalida);
```

2º El código al ser el 1 solo se utilizará un método la entrada de un objeto.

```
private static void AddTrabajador(Socket socket, ObjectInputStream flujoEntradaObj) {

    Trabajador trabajador = new Trabajador();


    try {
        ObjectInputStream objEntrada = new ObjectInputStream(socket.getInputStream());
        trabajador = (Trabajador)objEntrada.readObject();
        //trabajador2 = trabajador;
    } catch (ClassNotFoundException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
```

3º Se ejecuta el método AddTrabajador que abrirá el flujoEntradaObj que tendrá el trabajador que nos traen al servidor y lo meterá en un nuevo trabajador.

```
AñadirTrabajador añadirT = new AñadirTrabajador();
añadirT.añadirTrabajador(trabajador);
```

4º Ejecutará los métodos para almacenar a trabajador en la base de datos de SQL.

```
public void añadirTrabajador(Trabajador trabajador) {
    Connection con = null;
    PreparedStatement st = null;

    try {
        con = acceso.conectar();
        System.out.println("Conectado");

        st = con.prepareStatement("insert into trabajador (dni, nombre, apellido1, apellido2, domicilio, "
                + "domiciliosocial, nacionalidad, pais, municipio, codigopostal, numeroseguridad) values (?,?,?,?,?,?,?,?,?,?,?)");
        st.setString(1, trabajador.getDni());
        st.setString(2, trabajador.getNombre());
        st.setString(3, trabajador.getApellido1());
        st.setString(4, trabajador.getApellido2());
        st.setString(5, trabajador.getDomicilio());
        st.setString(6, trabajador.getDomicilioSocial());
        st.setString(7, trabajador.getNacionalidad());
        st.setString(8, trabajador.getPais());
        st.setString(9, trabajador.getMunicipio());
        st.setInt(10, trabajador.getCodigoPostal());
        st.setInt(11, trabajador.getNumeroSeguridad());
        st.execute();

        JOptionPane.showConfirmDialog(null, "Trabajador guardado correctamente", "Mensaje de confirmación",
                JOptionPane.CLOSED_OPTION);
```

Todos estos pasos deben seguir cualquier método para poder almacenarse en la base de datos, hay diferentes métodos que tocarán la base datos como puede ser CargarDatos, que saca todos los trabajadores de la base de datos o modificar, que cambia a un trabajador y por ultimo borrar trabajador, que lo eliminara de la base de datos, todos los métodos envían información a la base de datos menos dos métodos que sacan los datos de la base de datos, que son CargarDatos y CargarDatosfiltrados.

2.4. RESULTADOS Y VALIDACIÓN

El único problema real, fue el Scene Builder, se tiene que planetear si hacerlo en Windows Builder or seguir con Scene Builder.
El tiempo de ejecución es inmediato, lo único que se puede decir que tiene algo de retraso es el propio Splash Art del inicio, pero porque este mismo lo requiere para poder realizar la animación.
Los servidores usados, no tienen ningún problema en principio.
La base de datos se ha realizado por medio de SQLite, se trata de una base de datos realmente sencilla, dado que simplemente por el momento solo se tiene la tabla de trabajadores, y otra en la que se tendrán almacenados los usuarios y contraseñas de los administradores para usar la aplicación. Estas serán dictadas por los programadores, los cuales le crearán perfiles a cada administrador.
La base de datos de los trabajadores consta de los siguientes campos: D.N.I, nombre, primer apellido, segundo apellido, nacionalidad, país, domicilio social, domicilio, código postal, municipio y Nº de S.S.

Las pestañas que serán usadas y con una breve explicación, son las siguientes:

Splash

Figura: En esta imagen se muestra el Splash Screen

No tendrá ningún tipo de funcionalidad, simplemente será una manera bonita y profesional, de iniciar la aplicación.

Inicio

Figura: En esta imagen se puede apreciar la pantalla de login

En la pestaña de Inicio, el usuario podrá logearse con el usuario y contraseña correspondientes, en caso de haber olvidado la contraseña, podría optar a pedir un reenvío de la misma, con un simple click.

Datos

Figura: en la imagen se ve la página principal de la aplicación, donde los administradores podrán añadir, modificar, guardar o inscribir nuevos trabajadores. Hay un filtro, así como, un scroll pane donde estarán cargados la lista de todos los trabajadores.

Añadir

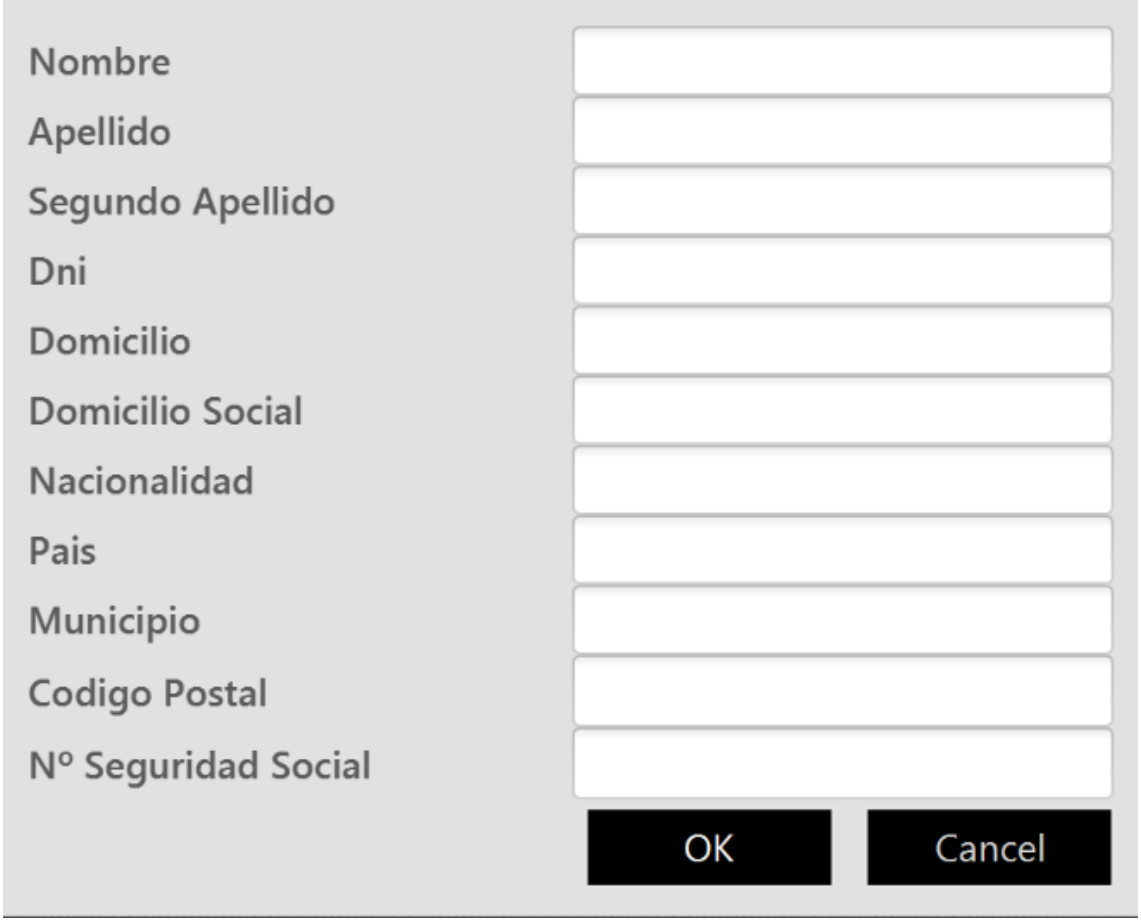


Figura: en la imagen se ve la pantalla para añadir nuevos trabajadores no registrados.

En esta pestaña, el administrador podrá registrar usuario, que no se hayan registrado con la aplicación de móvil, de manera sencilla y rápida.

3. CONCLUSIONES

El proyecto consta de 2 partes principalmente, el entorno gráfico, realizado con JavaFX y Scene Builder Framework, y el entorno lógico que consta de varias partes, la parte de los TPC servers, donde se relacionan las bases de datos entre sí. El modelo de las clases no difiere del tipo MVC, solo que está integrado el Apache TomCat server.
go
Se consiguió realizar la aplicación de escritorio donde los administradores podrán ingresar nuevos trabajadores y tener todo controlado de una manera más sencilla, la de móvil será la próxima parte, que se realizará en un futuro próximo.
Las primeras partes que se establecieron, fue la manera en la que se llevaría a cabo, que herramientas se usarían además de un Mockup.
Luego más tarde se entró en la programación de la aplicación en sí, una vez establecida la metodología del trabajo y como estaría repartido.

Serían necesarias más pruebas para poder demostrar 100% la fiabilidad de la aplicación, y en que ámbitos habría que mejorar.

El Scene Builder retrasó un poco el trabajo, dado que daba muchos problemas de apertura en ciertas ocasiones, y como es usual a la hora de juntar las diferentes partes del proyecto hubo que arreglar muchas cosas.

La imagen de la aplicación respecto al Mockup, no ha variado demasiado, simplemente se aplicaron ciertas animaciones que previamente no pudieron ser hechas en el Mockup, además del Splash con el cual paso exactamente lo mismo.

3.1. Innovación

Supuso un gran cambio respecto a la aplicación que tenían previamente, estaban usando métodos muy antiguos y en desuso, la

nueva aplicación fue un gran salto de calidad, mejorando el trabajo y la facilidad de este.
La fácil accesibilidad a diversas actividades reunidas en una sola aplicación y el fácil uso de esta misma, realmente mejoro la producción.
El equipo no tenía ninguna experiencia previa en este tipo de aplicaciones, si es cierto que se usaron cosas que durante el curso fueron aprendidas, ya sea el Scene Builder, o los TCP servers.

3.2. Trabajo futuro

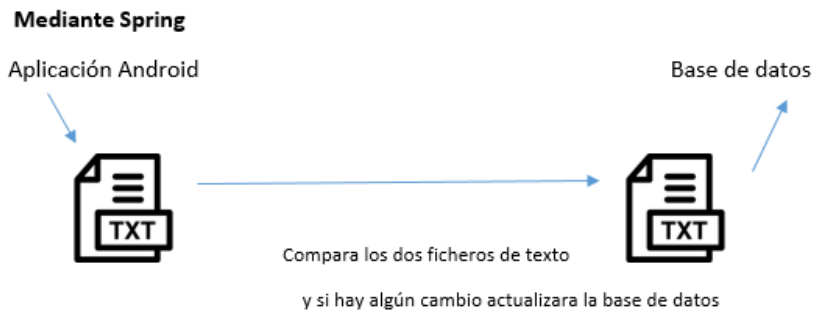
Esta aplicación fue una auténtica prueba de fuego para el equipo, gracias a esto, se sabe cuáles son los límites de cada miembro del equipo y hasta donde se podría llegar.
El proyecto ha sido realizado, pero no se quedará solo ahí, esto solo ha supuesto la mitad del trabajo propuesto.

- ➔ Hay varios Métodos que están ya en la aplicación, pero no se ha podido probar y no son funcionales en este punto de desarrollo.

- ➔ Se creará un método para encriptar los datos que vayan al servidor mediante el conjunto de biblioteca de encryption AES.

- ➔ Se creará la aplicación Android que tendrá un login y un añadir trabajador que tendrá generará un fichero de texto en el servidor.

- ➔ Se ampliará el servidor Socket con el servidor Tomcat y se usará Spring para poder leer los ficheros de texto que genere la aplicación Android y fusione los datos con las de la base de datos SQL.



4. BIBLIOGRAFÍA Y WEBGRAFÍA

https://conociendogithub.readthedocs.io/en/latest/data/introduccion/

https://www.oracle.com/es/index.html

https://www.java.com/

https://www.youtube.com/user/pildorasinformaticas

http://www.ajpdsoft.com/modules.php?name=Encyclopedia&op=content&tid=769