**Universidad Internacional de La Rioja (UNIR)**

**Escuela Superior de Ingeniería y Tecnología**

**Máster en Análisis y Visualización de Datos Masivos (Visual and Analytics Big Data)**

# Sistema de información gerencial de mercado para instituciones financieras ecuatorianas

**Trabajo Fin de Máster**

**Tipo de trabajo:** Desarrollo de Software

**Elaborado por:** León Morocho, Jorge Luis

**Director:** Tejeda Lorente, Álvaro

Ciudad: Cuenca, Ecuador

Fecha: 15 de septiembre de 2019

***Agradecimiento***

Debo agradecer a Dios por ser una guía constante en cada una de las etapas de mi vida, este master universitario necesitó de esa guía espiritual que me fortaleció en cada paso ejecutado.

Agradezco a mi esposa Fernanda y a mi hijo Mateo por toda la paciencia y el sacrificio realizado por parte de ellos para que yo pueda haber cursado este proceso de estudio, el cual requirió de mucho esfuerzo y tiempo no solo mío, también el de ellos.

Agradezco a mis padres Mariana y Jorge, ya que siempre me han inculcado una disciplina de trabajo y esfuerzo para poder cumplir cada meta que me he planteado y que sin sus constantes consejos yo no sería la persona dedicada y luchadora que soy.

Agradecer a mi guía profesional Wendy, que sin su apoyo y conocimientos no hubiera sido posible el planteamiento y desarrollo de este proyecto.

Y por último, agradecer a mi director de TFM Álvaro Tejeda, cuya experiencia y orientación a lo largo de la elaboración de la memoria y el desarrollo del sistema fue clave para el buen termino.

*"Knowledge and skill add up, but the attitude multiplies"*
*"Conocimiento y habilidad suman, pero la actitud multiplica"*
- Victor Küppers

**Resumen**

En Ecuador hace más de 20 años vivió un episodio conocido como el Feriado Bancario, en donde los Bancos congelaron los depósitos de los clientes pasándolos a manos del estado con el fin de apoyar a aquellos que estaban en riesgo de quebrar, todo esto le condujo a la dolarización lo cual afectó a la población ecuatoriana, por esa razón existen entidades públicas que supervisan y controlan a las instituciones financieras.

Los entes de control del Ecuador son la “Superintendencia de Bancos (SB)” y la “Superintendencia de Economía Popular y Solidaria (SEPS)”. Cada institución provee estructuras de datos a estos entes, para que puedan monitorizar la economía nacional, esta información se encuentra libre para que las mismas instituciones e incluso empresas financieras privadas puedan descargárselas en archivos y revisar el comportamiento del mercado nacional.

Con esto en mente, se planteó el TFM de tipo desarrollo de software cuyo objetivo es crear un repositorio centralizado de donde se alimente el Sistema de Información Gerencial, de tal forma que permita a una institución financiera encontrar en un solo lugar información relevante. Se basó en 4 fases que son extracción, almacenamiento, procesamiento y visualización. La información que se obtiene es periódica y de distintas instituciones financieras, se prevé que los datos crezcan. Se aplicó la metodología de desarrollo de software basado en el proceso unificado y con el modelado UML. Se elaboró un diseño de arquitectura master/esclavo para el back-end y front-end, se evaluó los tiempos de recolección de los datos, la aceptación de los usuarios en cuanto a la facilidad y funcionalidad.

El Sistema de información gerencial del mercado financiero ecuatoriano creado resulta una herramienta eficaz para el control de una institución financiera, aportando información que ayude a las gerencias a tomar decisiones dentro del mercado y sean un aporte a la economía nacional, mediante tics que están en auge y que son de libre uso.

Palabras Clave: SIG, institución financiera, SB, SEPS, big data, mercado financiero, tic

**Abstract**

In Ecuador, more than 20 years ago, he experienced an episode known as the *"Feriado Bancario"*, where the banks froze the deposits of the clients by passing them on to the state in order to support those who were at risk of bankruptcy, all of which led him to dollarization which affected the ecuadorian population, for that reason there are public entities that supervise and control financial institutions.

The control entities of Ecuador are the *"Superintendencia de bancos (SB)"* and the *"Superintendencia de economía popular y solidaria (SEPS)"*. Each institution provides data structures to these entities, so that they can monitor the national economy, this information is free so that the same institutions and even private financial companies can download them in archives and review the behavior of the national market.

The final work of master was created as a software development project whose objective is to create a centralized repository of these data to be visualized through a system called Management Information System, in such a way that it allows a financial institution to find relevant information in one place. It was based on 4 phases that are ex-traction, storage, processing and visualization. The information that is obtained from the periodical and from different financial institutions, it is expected that the data will grow. The methodology of software development based on the unified process and UML modeling was applied. A master / slave architecture design was developed for the back-end and front-end, the data collection times were evaluated, the acceptance of the users in terms of ease and functionality.

The Ecuadorian financial market management information system created is an effective tool for the control of a financial institution, providing information that helps managers to make decisions within the market and is a contribution to the economy, through tics that are booming and that are free to use.

Keywords: MIS, financial institution, SB, SEPS, big data, ecuadorian market, tic

# Índice de contenido

# Índice de ilustraciones

# Índice de tablas

# Lista de acrónimos

**ACRONIMOS** SIG SB SEPS TICS HDFS RDD SQL TFM

**SIG** Sistema de Información Gerencial

**SB** Superintendencia de bancos

**SEPS** Superintendencia de Economía Popular y Solidaria

**BCE** Banco Central del Ecuador

**TICS** Tecnologías de la Información

**HDFS** Hadoop Distributed File System

**RDD** Resilient Distributed Dataset

**SQL** Structured Query Language

**TFM** Trabajo de Fin de Máster

**CAF** Banco de Desarrollo de América Latina

**COSEDE** Corporación del Seguro de Depósitos

**TOI** Costo total de propiedad

**ROI** Retorno de Inversión

# 1. Introducción

Toda inversión que sea efectuada por empresas privadas tanto nacionales como internacionales, así como también la misma población de un determinado país, implica un movimiento de capitales gestionado por instituciones financieras públicas y privadas. Este movimiento del dinero en un país hace que las instituciones financieras cumplan un papel importante en la economía y como tal, se deba tener un control de como este se mueve a lo largo del tiempo, pudiendo de esta manera predecir comportamientos futuros de cara a cualquier eventualidad que pudiera suceder en base a distintas variables internas como externas al país.

En el Ecuador este control que ejerce el gobierno es realizado a través de sus instituciones públicas, estas se encargan de solicitar información a todas las instituciones financieras para mantener un conjunto de datos centralizado. La información se encuentra alojada en las bases de datos que mantiene el gobierno nacional, el problema que se tiene es en el acceso de estos datos, ya que las instituciones públicas no disponen de servicios que permitan mediante sistemas informáticos acceder a esa información, si estas pudieran revisar constantemente la información de como esta el comportamiento del mercado, basándose en datos que todas las instituciones aportan entonces, podrán marcar un flujo de trabajo que beneficie a todas y por ende este beneficio es bueno para el país.

El trabajo de extraer estos datos es la parte tediosa y compleja en algunos casos, ya que los entes de control las exponen en sus sitios web, además indicar que por cada cambio de gobierno estos tienden a reestructurar sus páginas web, entonces descargarlas de por sí requiere conocer la ubicación o el enlace que como indico varía, además que esta no está en un formato estandarizado, en algunos son archivos de excel, en otras es página web, en otras son archivos de texto, además mencionar que existen distintos sitios web de donde se puede revisar esta información, entonces se entiende el problema que tiene una institución financiera si desea revisar estos datos.

Surge una razón de facilitar un proyecto que apoye a las instituciones financieras la revisión de estos datos, es decir, facilitarles la consulta y en un solo lugar, para ellos es enormemente beneficioso por que se evitan tener que construir procesos informáticos internos e invertir económicamente en algunos casos. Pero no solo ayudarles en la extracción de los datos es el objetivo, sino el de apoyarles en sus análisis económicos por medio de visualizaciones o reportes, entonces sus tiempos de análisis se reducen enormemente y únicamente se dedican a tomar decisiones en base a la información que el sistema les provee.

Como se menciona en el párrafo anterior, los datos que las instituciones proveen a los entes de control son el interés del trabajo de fin de máster que se ha planteado, y que se detalla a lo largo de este documento, partiendo de un marco conceptual que brindará el sustento necesario que guiará al desarrollo del proyecto. En el documento se tiene detallado las 4 fases importantes en las que se basa el proyecto y son la extracción, almacenamiento, procesamiento y visualización de la información, estas son la base del sistema a desarrollar ya que se tienen que elaborar una serie de componentes por cada fase.

El planteamiento esta orientado a big data ya que la información a capturar con el paso del tiempo crecerá y será de mucha ayuda para procesos de ciencia de datos, como es el caso de predicciones y de machine learning, si bien el alcance del proyecto no abarca estos temas, si son importantes mencionarlos porque con un repositorio de datos históricos es posible llevarlos a cabo.

## 1.1. Justificación

Como vimos en la introducción, el comportamiento de las instituciones financieras privadas son de suma importancia para el estado ya que de cierta forma se puede ver el comportamiento del mercado en base al comportamiento de estas, por lo que son un elemento necesario para el financiamiento de nuevos proyectos de inversión, así como para el crecimiento de la capacidad instalada de las actividades productivas del país (Asobanca, 2014).

La forma en que estas apoyan al crecimiento del país vienen dadas por los créditos y los depósitos que tanto personas naturales y jurídicas realizan, precisamente estos son uno de los puntos que se analizan en los reportes macroeconómicos, por lo que es importante que las instituciones puedan tener un control de estos indicadores, así como la necesidad de que el gobierno pueda tener un control sobre las instituciones, ya que a la larga son los recursos monetarios de la ciudadanía con la que estas negocian y generan utilidades.

Del anterior párrafo destacamos dos puntos importantes, depósitos (pasivos) y créditos (activos), en otras palabras lo que se capta se coloca, manteniendo un margen de estas captaciones como un colchón de seguro frente a posibles riesgos que se puedan presentar. Estos pasivos y activos son declarados mensualmente por las instituciones financieras a las entidades de control mencionadas anteriormente y es lo que las mismas instituciones necesitan analizar para poder determinar el comportamiento del mercado financiero ecuatoriano. Las instituciones financieras no solicitan datos directamente entre ellas, estas lo hacen por medio

de los sitios que los entes de control disponen, esta es la una de las razones que justifican el planteamiento del TFM.

Existen empresas particulares que facilitan a las instituciones financieras este proceso pero de forma manual, es decir acceder al sitio web, bajarse el archivo cuando se pueda, revisarlo y tomar los datos de interés, pero este proceso para ellos es demorado, por lo que en ocasiones tienden a evitar descargarse datos precisamente por el trabajo que esto requiere, tenemos algunos casos de lo que realizan estas empresas:

- El repositorio de información se basa en archivos de excel.
- Se realizan análisis sobre estos documentos y se transfieren a un sistema web.
- En ocasiones, existen repositorio de datos que se van limitando conforme aumenta el volumen de información.
- No permiten almacenar información histórica que podría emplearse para realizar una inteligencia de negocios analítica.
- Los dashboards hacen uso de librerías en donde se deben adaptar los datos.
- Las instituciones financieras se encuentran segmentadas y separadas en distintas fuentes por lo que no se puede realizar un análisis adecuado, por ejemplo entre Bancos y Cooperativas de Ahorro y Crédito.

Como vemos la problemática actual que se tiene es en la consulta de la información, por un lado se plantea disponer un solo lugar de almacenamiento (repositorio) de estos datos; la otra parte es la construcción de un sistema que brinde un análisis claro y fiable del comportamiento del mercado ecuatoriano, solventando la dispersión de los datos y la complejidad en los análisis.

El beneficio que obtiene una institución financiera con este proyecto es alta puesto que reducen sus tiempos de análisis y toma de decisiones, también el manejar datos en una estructura estandarizada facilita a generar reportes de una forma rápida, clara y consistente.

Resumiendo, se trata de construir un repositorio que abarque tecnologías *big data*, habilitando un crecimiento horizontal y buscando una reducción en los tiempos de análisis y toma de decisiones, a través de un sistema de información visual donde permita una interacción directa del usuario.

## 1.2. Planteamiento del trabajo

Como se mencionó en la sección anterior, existen distintas soluciones que permiten a una institución financiera poder llevar a cabo análisis de su situación actual, pero estas en cierto modo se encuentran limitadas. Para ello, me puse en contacto con una empresa ecuatoriana que tiene varios años de trayectoria y experiencia en el país, cuyo nicho de mercado son los bancos y cooperativas, ellos brindan asesoría en temas relacionados a finanzas, para ello, cuentan con un Sistema de Información Gerencial que por cuestiones de tiempo no han logrado optimizar quedándose relegado en temas de almacenamiento y procesamiento de información, así como en la visualización de los datos.

En base a la reunión que se mantuvo con la gerencia de la empresa, se llegó a un acuerdo de apoyo mutuo en donde ellos brindaron la asesoría técnica en temas orientados a finanzas y en como se podría presentar la información al usuario final y de mi lado adaptar sus necesidades haciendo uso de las temáticas aprendidas en el "Máster de Análisis y Visualización de Datos Masivos", las cuáles en base a mi análisis se adaptaron a la perfección y permitieron que pueda trabajar sobre esto en mi proyecto de fin de máster, pero delimitando el alcance del sistema, enfocándome únicamente en los siguientes aspectos:

- Visualización de la tendencia de las captaciones de las instituciones financieras.
- Visualización de la tendencia de las colocaciones de las instituciones financieras.
- Autenticación y Autorización de usuarios.
- Visualización de variaciones en las captaciones y colocaciones.
- Comparaciones de una institución con otras.

En base a esto, el Sistema de Información requirió un planteamiento enfocado a cuatro fases básicas que son:

1. Extracción
2. Almacenamiento
3. Procesamiento
4. Visualización

Estos puntos se encuentran detallados en la sección del estado del arte 4. Para concluir, el planteamiento del TFM consistió en la extracción de la información de los entes de control de forma automática y su posterior almacenamiento en un repositorio central, siguiendo previamente un proceso de transformación de los datos, además, al tener esta información en el repositorio se pudo realizar procesos de análisis y consolidación de la información para que fueran consultados por el Sistema de Información Gerencial a través de API's. Es importante recalcar que el Sistema de Información Gerencial es un cliente que consume del repositorio, pero se lo puede usar para cualquier otros proceso como análisis predictivos, machine learning, por esa razón la necesidad de hacer uso de herramientas big data, vale aclarar que estos no forman parte del proyecto TFM pero puede ser considerados para trabajos futuros.

## 1.3. Estructura de la memoria

El presente trabajo está integrado por cuatro apartados, el capítulo I presenta una introducción, el cual, contiene la justificación, planteamiento y detalla la organización de los contenidos que conforman el documento presente.

En el capítulo II, se definen los fundamentos teóricos del desarrollo del proyecto, el estado del arte que incluye una investigación sobre temas del sistema financiero ecuatoriano, conceptos que engloban a un SIG, detalle de las instituciones financieras que operan en el país, las fuentes de información con las que se trabajó, la aplicación del big data en instituciones financieras y las herramientas empleadas en la construcción del proyecto.

En el capítulo III, se definen los objetivos generales y específicos del proyecto planteado, así como la metodología de trabajo, y una mayor explicación en cuanto a las fases que rigen el flujo de procesos que van desde la extracción hasta la presentación de los datos en el SIG.

En el capítulo IV, se muestran los requisitos para la propuesta de la aplicación a diseñar, se describe la solución propuesta y se plantean los distintos patrones que van desde el punto de vista de diseño hasta la concepción de la arquitectura del sistema tanto para el almacenamiento como para la visualización. Dentro de este apartado se realiza la validación de la herramienta en términos de usabilidad y accesibilidad, basándose en encuestas que serán entregadas a los usuarios y a los administradores de la aplicación.

En el capítulo V se tiene la contribución de la solución es decir se explica siguiendo el enfoque del Proceso Unificado de Desarrollo la ejecución de las actividades de desarrollo y su

implementación.

En el capítulo VI, VII, VIII y IX se plantean las conclusiones obtenidas luego de haber desarrollado el TFM, cuales serían las posibles aplicaciones de trabajo futuro, así como también la bibliografía empleada, y los anexos que servirán de apoyo para lectura de la memoria.

# 2. Estado del arte

## 2.1. Introducción

En este capítulo, se presenta en detalle los distintos temas que permitieron iniciar en el estudio del diseño y construcción del Sistema de Información Gerencial para la toma de decisiones enfocado al mercado financiero ecuatoriano, por lo que se tiene una revisión detalla de como se encuentra actualmente el sistema financiero ecuatoriano y dentro de esta sección especificar las instituciones financieras que operan a nivel nacional y de las cuales se recopiló información, se entiende el funcionamiento y la razón de ser de un sistema para la toma de decisiones, así como sus ventajas y desventajas y el apoyo que brinda a las gerencias en su día a día, como punto clave se recalca la influencia del big data dentro de las instituciones financieras esto con el objetivo de entender cuando se trabaja con big data y cuando no, y para finalizar, una breve inducción de las herramientas usadas en el proyecto.

## 2.2. Sistemas Financieros

### 2.2.1. Inicios de la banca

Al inicio las civilizaciones antiguas usaban el trueque como métodos de comercialización, tornándose inapropiados ya que al realizar intercambios como metales preciosos, era necesario que los sitios en donde se receptaban ofrezcan garantías adecuadas basándose por ejemplo en el peso y calidad, esto genero costos implícitos por transacción, si a esto se le suma los costos de transporte y además de los riesgos asociados como el hurto durante el transporte o almacenamiento, poco a poco se vio la necesidad de sustituir esta forma de trabajo (Le Roy Miller, 1992).

Bajo este concepto, nacen los primeros depositantes (cliente/usuario) y los custodios

(receptores de los bienes). Estos últimos empiezan a cobrar una comisión para preservar, almacenar y cambiar los bienes de los depositantes, entregándoles un recibo firmados por ellos y con este, los depositantes podían solicitar el cambio respectivo, estos se denominaron pagares del oferente (Le Roy Miller, 1992).

Con el paso del tiempo, el público adquirió mas confianza en estos pagares, por su lado el custodio noto que no necesitaban tener en sus bóvedas todos los bienes (metales, monedas), permitiéndoles constituir una reserva de contingencia evitando tener en sus bóvedas toda la cantidad de monedas o metales preciosos, de esta manera reducían sus costos y evitaban cargarlos a sus clientes, así fue como los custodios se convirtieron en banqueros, prestaban su actividad de resguardo sin costo para el cliente a cambio de la actividad de prestamistas, formándose un apoyo mutuo entre ambas partes (Le Roy Miller, 1992).

### 2.2.2. Desarrollo financiero y desarrollo económico

Es importante tener claro como el sistema financiero es clave para el desarrollo económico de un determinado país. Rueda (2013) indica que debe existir una relación entre ambos ya que los servicios prestados por el sistema financiero son importantes para el crecimiento a largo plazo, esto genera nuevos proyectos como canalización de ahorros, diversificación del riesgo y reducción de costos de transacción, esto ha sido un constante debate entre economistas (Zuleta & Carvajal, 1997).

Para probar la hipótesis del impacto del desarrollo financiero respecto al crecimiento económico, tenemos el analizado por CAF (2011), en donde afirman: "(...) existe una correlación positiva entre la intermediación financiera como proporción del PIB y el nivel de actividad económica, basándose en datos tomados en un período de tiempo desde 1860 a 1963 de distintos países (...) junto con otros trabajos se argumentó que las funciones desempeñadas por el sistema financiero afectan al crecimiento estacionario al tener influencia en los niveles de formación de capital, alterando la tasa de ahorro y reasignando el crédito entre las tecnologías que producen capital Goldsmith, (1969)." demostrando así la importancia del desarrollo del sistema financiero en las economías a largo plazo.

### 2.2.3. Sistema Financiero Ecuatoriano

En el Ecuador predomina un sistema de banca universal, en el que imperan los bancos comerciales en donde las instituciones financieras privadas se clasifican en Bancos, Sociedades Financieras, Mutualistas, Cooperativas, y por otro lado se encuentra el Sistema Financiero Público (Rueda, 2013).

Teniendo presente que el Sistema Financiero Ecuatoriano cambio por el año 2000 del Sucre al Dólar, esta ha venido a fortalecerse, además de tener una confianza robusta por parte de la población. En esta sección se revisa la evolución de las captaciones y colocaciones de las entidades financieras con corte a abril-2019 con respecto a diciembre-2018.

A continuación se muestran las principales cuentas del sistema financiero.

| Bancos Privados | dic-17 | ene-18 | nov-18 | dic-18 | Var. Mensual % | Var. Mensual Abs. | Var. Anual % | Var. Anual Abs. |
|---|---|---|---|---|---|---|---|---|
| Número de Instituciones | 24 | 24 | 24 | 24 | 0,0% | 0 | 0,0% | 0 |
| Activos | 38.975 | 38.524 | 39.879 | 40.984 | 2,8% | 1.105 | 5,2% | 2.009 |
| Pasivos | 34.757 | 34.272 | 35.317 | 36.372 | 3,0% | 1.055 | 4,6% | 1.615 |
| Patrimonio | 4.218 | 4.253 | 4.561 | 4.612 | 1,1% | 50 | 9,3% | 394 |
| Utilidad Neta | 396 | 39 | 504 | 554 | 9,8% | 49 | 39,9% | 158 |
| Ingresos | 4.055 | 380 | 4.090 | 4.515 | 10,4% | 425 | 11,4% | 460 |
| Gastos | 3.659 | 340 | 3.586 | 3.961 | 10,5% | 375 | 8,3% | 302 |
| Cartera Bruta | 24.601 | 24.677 | 27.170 | 27.325 | 0,6% | 155 | 11,1% | 2.725 |
| Por Vencer | 23.873 | 23.889 | 26.327 | 26.609 | 1,1% | 281 | 11,5% | 2.736 |
| Improductiva | 728 | 788 | 842 | 717 | -14,9% | -126 | -1,5% | -11 |
| Provisiones | 1.706 | 1.749 | 1.844 | 1.775 | -3,8% | -70 | 4,0% | 69 |
| Total Depósitos | 28.566 | 28.244 | 28.510 | 29.172 | 2,3% | 662 | 2,1% | 606 |
| Depósitos Monetarios | 10.468 | 10.352 | 9.917 | 10.142 | 2,3% | 225 | -3,1% | -325 |
| Depósitos de Ahorro | 8.659 | 8.297 | 8.064 | 8.642 | 7,2% | 578 | -0,2% | -17 |
| Depósitos a Plazo | 9.440 | 9.595 | 10.529 | 10.387 | -1,3% | -141 | 10,0% | 948 |

Figure 1: Principales Cuentas Sistema Financiero: Bancos (en millones de USD)
*Fuente.* Asobanca (2019)

| Cooperativas | nov-17 | dic-17 | oct-18 | nov-18 | Var. Mensual % | Var. Mensual Abs. | Var. Anual % | Var. Anual Abs. |
|---|---|---|---|---|---|---|---|---|
| Número de Instituciones | 64 | 64 | 71 | 71 | 0,0% | 0 | 10,9% | 7 |
| Activos | 9.536 | 9.700 | 11.182 | 11.266 | 0,8% | 84 | 18,1% | 1.730 |
| Pasivos | 8.152 | 8.302 | 9.545 | 9.609 | 0,7% | 64 | 17,9% | 1.457 |
| Patrimonio | 1.384 | 1.398 | 1.637 | 1.658 | 1,2% | 20 | 19,8% | 274 |
| Utilidad Neta | 97 | 99 | 137 | 149 | 9,2% | 13 | 54,2% | 52 |
| Ingresos | 1.029 | 1.137 | 1.168 | 1.293 | 10,7% | 125 | 25,7% | 264 |
| Gastos | 932 | 1.038 | 1.031 | 1.144 | 10,9% | 112 | 22,7% | 212 |
| Cartera Bruta | 6.612 | 6.717 | 8.470 | 8.615 | 1,7% | 145 | 30,3% | 2.003 |
| Por Vencer | 6.272 | 6.410 | 8.154 | 8.291 | 1,7% | 137 | 32,2% | 2.019 |
| Improductiva | 340 | 307 | 317 | 324 | 2,4% | 8 | -4,7% | -16 |
| Provisiones | 385 | 383 | 413 | 420 | 1,5% | 6 | 9,1% | 35 |
| Total Depósitos | 7.411 | 7.545 | 8.579 | 8.612 | 0,4% | 33 | 16,2% | 1.201 |
| Depósitos Monetarios | - | - | - | - | - | - | - | - |
| Depósitos de Ahorro | 2.546 | 2.645 | 2.807 | 2.811 | 0,1% | 4 | 10,4% | 265 |
| Depósitos a Plazo | 4.865 | 4.900 | 5.772 | 5.801 | 0,5% | 29 | 19,2% | 936 |

Figure 2: Principales Cuentas Sistema Financiero: Cooperativas (en millones de USD)
*Fuente.* Asobanca (2019)

| Mutualistas | nov-17 | dic-17 | oct-18 | nov-18 | Var. Mensual % | Var. Mensual Abs. | Var. Anual % | Var. Anual Abs. |
|---|---|---|---|---|---|---|---|---|
| Número de Instituciones | 4 | 4 | 4 | 4 | 0,0% | 0 | 0,0% | 0 |
| Activos | 901 | 916 | 962 | 964 | 0,3% | 3 | 7,0% | 63 |
| Pasivos | 823 | 837 | 880 | 882 | 0,2% | 2 | 7,1% | 59 |
| Patrimonio | 79 | 79 | 82 | 83 | 1,1% | 1 | 5,5% | 4 |
| Utilidad Neta | 3 | 4 | 3 | 4 | 29,6% | 1 | 12,9% | 0 |
| Ingresos | 94 | 103 | 90 | 99 | 10,3% | 9 | 5,5% | 5 |
| Gastos | 90 | 99 | 87 | 95 | 9,6% | 8 | 5,2% | 5 |
| Cartera Bruta | 528 | 532 | 593 | 602 | 1,5% | 9 | 14,0% | 74 |
| Por Vencer | 495 | 502 | 564 | 572 | 1,4% | 8 | 15,7% | 78 |
| Improductiva | 34 | 29 | 29 | 30 | 2,8% | 1 | -10,8% | -4 |
| Provisiones | 20 | 19 | 19 | 20 | 1,8% | 0 | 0,3% | 0 |
| Total Depósitos | 763 | 776 | 808 | 807 | 0,0% | 0 | 5,9% | 45 |
| Depósitos Monetarios | - | - | - | - | - | - | - | - |
| Depósitos de Ahorro | 255 | 268 | 256 | 256 | 0,0% | 0 | 0,3% | 1 |
| Depósitos a Plazo | 507 | 507 | 552 | 551 | -0,1% | 0 | 8,7% | 44 |

Figure 3: Principales Cuentas Sistema Financiero: Mutualistas (en millones de USD)
*Fuente.* Asobanca (2019)

En base a esta información, recalcando que se basa en datos que las mismas instituciones reportan, se determinan una serie de indicadores que son muy importantes para conocer la actividad que tienen dentro del sistema financiero.

| Bancos Privados | dic-17 | ene-18 | nov-18 | dic-18 | Var. Mensual Abs. | Var. Anual Abs. |
|---|---|---|---|---|---|---|
| ROE | 10,36 | 11,13 | 13,40 | 13,65 | 0,24 | 3,29 |
| ROA | 1,02 | 1,21 | 1,41 | 1,35 | -0,05 | 0,34 |
| Calidad de Activos | 133,89 | 136,71 | 137,49 | 134,87 | -2,62 | 0,98 |
| Eficiencia | 117,61 | 119,29 | 130,15 | 130,04 | -0,10 | 12,44 |
| Intermediación Financiera | 74,60 | 75,92 | 92,79 | 91,56 | -1,23 | 16,96 |
| Apalancamiento | 8,24 | 8,06 | 7,74 | 7,89 | 0,14 | -0,35 |
| Liquidez | 29,41 | 27,51 | 24,84 | 27,89 | 3,05 | -1,52 |
| Morosidad | 2,96 | 3,19 | 3,10 | 2,62 | -0,48 | -0,34 |
| Cobertura | 234,38 | 221,90 | 218,94 | 247,65 | 28,72 | 13,27 |
| Solvencia | 13,68 | 13,56 | 13,25 | 13,40 | 0,15 | -0,28 |

Figure 4: Indicadores Financiero: Bancos (en millones de USD)
*Fuente.* Asobanca (2019)

| Cooperativas | nov-17 | dic-17 | oct-18 | nov-18 | Var. Mensual Abs. | Var. Anual Abs. |
|---|---|---|---|---|---|---|
| ROE | 8,46 | 7,62 | 10,70 | 10,58 | -0,12 | 2,12 |
| ROA | 1,02 | 1,02 | 1,22 | 1,32 | 0,10 | 0,31 |
| Calidad de Activos | 113,95 | 113,81 | 114,65 | 115,05 | 0,40 | 1,10 |
| Eficiencia | 81,33 | 82,81 | 72,45 | 72,73 | 0,28 | -8,60 |
| Intermediación Financiera | 88,16 | 87,98 | 98,04 | 99,36 | 1,32 | 11,20 |
| Apalancamiento | 5,89 | 5,94 | 5,83 | 5,80 | -0,03 | -0,09 |
| Liquidez | 24,30 | 25,35 | 19,74 | 19,42 | -0,32 | -4,88 |
| Morosidad | 5,15 | 4,57 | 3,74 | 0,04 | -3,70 | -5,11 |
| Cobertura | 113,13 | 124,73 | 130,60 | 129,52 | -1,08 | 16,39 |
| Solvencia | 18,61 | 18,54 | 16,95 | 17,39 | 0,44 | -1,23 |

Figure 5: Indicador Financiero: Cooperativas (en millones de USD)
*Fuente.* Asobanca (2019)

| Mutualistas | nov-17 | dic-17 | oct-18 | nov-18 | Var. Mensual Abs. | Var. Anual Abs. |
|---|---|---|---|---|---|---|
| ROE | 4,84 | 5,11 | 4,55 | 5,35 | 0,81 | 0,51 |
| ROA | 0,42 | 0,42 | 0,38 | 0,45 | 0,07 | 0,03 |
| Calidad de Activos | 94,94 | 95,21 | 90,34 | 89,67 | -0,66 | -5,26 |
| Eficiencia | 124,66 | 122,02 | 112,74 | 110,04 | -2,69 | -14,62 |
| Intermediación Financiera | 69,16 | 68,45 | 73,36 | 74,48 | 1,12 | 5,33 |
| Apalancamiento | 10,48 | 10,59 | 10,73 | 10,63 | -0,10 | 0,16 |
| Liquidez | 11,50 | 12,89 | 7,03 | 7,04 | 0,00 | -4,47 |
| Morosidad | 6,38 | 5,50 | 4,92 | 4,99 | 0,07 | -1,39 |
| Cobertura | 58,44 | 65,02 | 66,38 | 65,72 | -0,66 | 7,28 |
| Solvencia | 12,58 | 12,39 | 12,61 | 12,57 | -0,04 | -0,01 |

Figure 6: Indicador Financiero: Mutualistas (en millones de USD)
*Fuente.* Asobanca (2019)

Como dato adicional, en la siguiente imagen se muestra la situación a la fecha Abril-2019 de los depósitos de bancos del Ecuador:

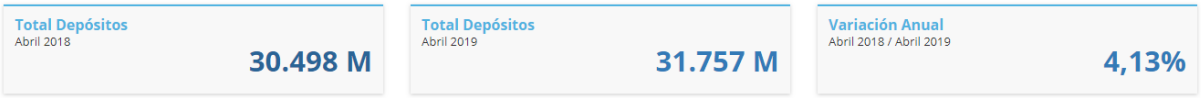



Figure 7: Total depósitos bancos (en millones de USD)
*Fuente.* https://www.superbancos.gob.ec/bancos/

Como un breve resumen explicativo tenemos un listado de 24 bancos privados, 64 cooperativas y 4 mutualistas, se observa que para diciembre del 2018 los bancos cerraron el año con un saldo de USD 27.325 millones en su cartera bruta, teniendo una tasa de crecimiento igual al 11.1% con respecto a diciembre de 2017; por otro lado, el valor de los depósitos mostró un saldo de USD 29.172 millones, corresponde a un crecimiento anual de 2.1%. Estos indicadores muestran que la banca inicia una nueva etapa a una economía de menor liquidez (Asobanca, 2019).

Del análisis que se explica en el párrafo anterior a breves rasgos es lo que se pretende aportar a las instituciones financieras con el sistema de información, este análisis se basa en un listado de bancos, cooperativas y mutualistas que se encuentran detallados en la sección de anexos y que son visualizados en el SIG. (ver Anexo I. Listado Instituciones Financieras 8).

## 2.3. Fuentes de información

De la sección anterior, se tuvo revisó un reporte de la situación actual del sistema tomado de la Asobanca, la finalidad de incluir este tema dentro del estado del arte es para poder dar una visión general al lector de a donde se desea llegar con el proyecto. Los datos son la base del sistema, sin estos no se podría construir un SIG; en esta sección nos adentramos a la revisión de los lugares de donde se puede extraer esta información.

En las siguientes secciones se hablará sobre la Superintendencia de Bancos SB y la Superintendencia de economía popular y solidaria SEPS.

### 2.3.1. Superintendencia de Bancos - SB

En la página oficial de la Superintendencia de Bancos tenemos una breve historia de lo que es y realiza la institución.

> Por primera vez se nombró una autoridad de supervisión de los bancos, mediante decreto ejecutivo en 1914, se creó el cargo de Comisario Fiscal de Bancos, su misión era vigilar la emisión y cancelación de los billetes de bancos, medida que entonces se dictó como de emergencia. En 1927, bajo inspiración de la misión Kemmerer (1925 – 1927), llamada así porque la presidió el doctor Edwin Walter Kemmerer, produjo en el país una verdadera transformación en la rama bancario y financiera al expedir: La Ley Orgánica de Bancos, la Ley Orgánica del Banco Hipotecario (Banco Nacional de Fomento) y la Ley Orgánica del Banco Central, que afianzaron el sistema financiero del país, así como otras leyes que regularon el manejo de la Hacienda Pública. Desde entonces, se estableció la supervisión de las operaciones bancarias mediante la creación de la SUPERINTENDENCIA DE BANCOS el día 6 de Septiembre de 1927. («Historia de la Superintendencia», 2019)

La misión de la Superintendencia indica: "Supervisar y controlar las actividades que ejercen las entidades financieras y de seguridad social, públicas y privadas, con el propósito de proteger los intereses de la ciudadanía y fortalecer los sistemas controlados". («Superintendencia de bancos: Misión y Visión», 2019)

La visión de la Superintendencia indica: "Ser una institución referente de supervisión y control que protege a la gente, promoviendo la estabilidad de los sistemas financiero y de

seguridad social, con personal reconocido por su capacidad técnica que aplica procesos y tecnología eficientes". («Superintendencia de bancos: Misión y Visión», 2019)

La SB debe exigir los distintas estructuras de información a Bancos y Asociaciones, esta información se encuentra libre para consulta en la página web de la Superintendencia, la información que nos resulta de interés para el proyecto son los balances generales de todos los bancos del país, (Tabla. 13). Se puede visualizar la estructura del balance accediendo al siguiente enlace https://www.superbancos.gob.ec/bancos/balance-general-prueba/ el cual nos redirigue a la sección de balances que podemos ver en las siguientes imágenes:


Figure 8: Fuente de información Superintendencia de Bancos
*Fuente.* http://oidprd.sbs.gob.ec/practg/pk_cons_bdd.p_bal_entdd_finnc

.: Balance General :.

| | |
|---|---|
| ENTIDAD: | 1004 |
| NOMBRE: | BP AUSTRO |
| FECHA DEL BALANCE: | 30-APR-19 |

| Código | Nombre de la Cuenta | Saldo |
|---|---|---|
| 1 | ACTIVO | 1,731,544,094.48 |
| 11 | FONDOS DISPONIBLES | 270,818,700.07 |
| 1101 | Caja | 52,252,949.26 |
| 110105 | Efectivo | 52,249,899.26 |
| 110110 | Caja chica | 3,050.00 |
| 1102 | Depósitos para encaje | 109,938,080.12 |
| 110205 | Banco Central del Ecuador | 109,927,567.72 |
| 110210 | Banco Nacional de Fomento | 10,512.40 |
| 110215 | Bancos locales | 0.00 |
| 1103 | Bancos y otras instituciones financieras | 92,493,161.71 |
| 110305 | BANCOS Y OTRAS INSTITUCIONES FINANCIERAS- BANCO CENTRAL DEL ECUADOR | 0.00 |
| 110310 | Bancos e instituciones financieras locales | 67,035,554.01 |
| 110315 | Bancos e instituciones financieras del exterior | 25,457,607.70 |
| 1104 | Efectos de cobro inmediato | 16,134,508.98 |
| 1105 | Remesas en tránsito | 0.00 |
| 110505 | Del país | 0.00 |
| 110510 | Del exterior | 0.00 |
| 12 | OPERACIONES INTERBANCARIAS | 0.00 |
| 1201 | Fondos interbancarios vendidos | 0.00 |
| 120105 | Bancos | 0.00 |
| 120110 | Otras instituciones del sistema financiero | 0.00 |
| 1202 | Operaciones de reporto con instituciones financieras | 0.00 |
| 120205 | Instituciones financieras públicas | 0.00 |
| 120210 | Bancos | 0.00 |
| 120215 | Otras instituciones del sistema financiero | 0.00 |
| 1299 | (Provisión para operaciones interbancarias y de reporto) | 0.00 |
| 129905 | (Provisión fondos interbancarios vendidos) | 0.00 |
| 129910 | (Provisión para operaciones de reporto con instituciones financieras) | 0.00 |
| 13 | INVERSIONES | 340,182,147.68 |
| 1301 | A valor razonable con cambios en el estado de resultados de entidades del sector privado | 0.00 |
| 130105 | De 1 a 30 días | 0.00 |
| 130110 | De 31 a 90 días | 0.00 |
| 130115 | De 91 a 180 días | 0.00 |
| 130120 | De 181 a 360 días | 0.00 |
| 130125 | De más de 360 días | 0.00 |
| 1302 | A valor razonable con cambios en el estado de resultados del Estado o de entidades del sector público | 0.00 |
| 130205 | De 1 a 30 días | 0.00 |
| 130210 | De 31 a 90 días | 0.00 |
| 130215 | De 91 a 180 días | 0.00 |
| 130220 | De 181 a 360 días | 0.00 |
| 130225 | De más de 360 días | 0.00 |
| 1303 | Disponibles para la venta de entidades del sector privado | 164,890,516.49 |
| 130305 | De 1 a 30 días | 33,850,440.10 |
| 130310 | De 31 a 90 días | 58,155,992.84 |
| 130315 | De 91 a 180 días | 29,236,384.46 |
| 130320 | De 181 a 360 días | 17,159,685.57 |

Figure 9: Ejemplo de balance de bancos
*Fuente.* http://oidprd.sbs.gob.ec/practg/pk_cons_bdd.p_bal_entdd_finnc

Esta ubicación corresponde a la primera fuente de información desde donde se extraerán los datos de balances generales. Como se puede observar en la imagen, los datos se cargan directamente en una página HTML, por lo que no existe ninguna descarga de archivos, el objetivo en este caso es leer el html y proceder a descargar la tabla al repositorio local. El proceso a seguir en este caso es un *Web Scrapping*, el cual es diferente para el caso de Cooperativas y Mutualistas que se detalla en la siguiente sección.

### 2.3.2. Superintendencia de economía popular y solidaria - SEPS

Tomando como referencia la definición que la misma entidad define en su página web, esta nos indica lo que es y hace la entidad de control.

> Es una entidad técnica de supervisión y control de las organizaciones de la economía popular y solidaria, con personalidad jurídica de derecho público y autonomía administrativa y financiera, que busca el desarrollo, estabilidad, solidez y correcto funcionamiento del sector económico popular y solidario. La SEPS inició su gestión el 5 de junio de 2012, día en que Hugo Jácome –Superintendente de Economía Popular y Solidaria– asumió sus funciones ante el pleno de la Asamblea Nacional. («¿Qué es la SEPS?», 2019)

La misión de la SEPS indica: "Supervisar y controlar, dentro del ámbito de sus atribuciones, a las organizaciones de la Economía Popular y Solidaria en busca de su estabilidad y correcto funcionamiento para el bienestar de sus integrantes y de la comunidad en general" («Superintendencia de Economía Popular y Solidaria: Misión y Visión», 2019)

La visión de la SEPS nos indica: "En el año 2022, seremos una institución técnica líder regional y referente internacional, en supervisión y control de las organizaciones de Economía Popular y Solidaria, que promueve la consolidación del sector y que contribuye al buen vivir de la ciudadanía". («Superintendencia de Economía Popular y Solidaria: Misión y Visión», 2019)

El objetivo de la entidad es muy similar a la de la Superintendencia de Bancos, cada una esta enfocada a un grupo de instituciones financieras, pues como vemos en el listado de cooperativas y mutualistas (ver Tabla 14) estas son numerosas, por esa razón, la entidad las subdivide en cinco segmentos, la asignación de la institución dentro de cada segmento esta dado por el volumen de sus activos, por lo que se entiende que el segmento 1, es el listado de las cooperativas mas grandes del país, les sigue el segmento 2 y así sucesivamente, por esa razón es que el proyecto de TFM se basa en los tres primeros segmentos y mutualistas. La tabla siguiente indica como realizan la segmentación:

| Segmento | Activos |
|---|---|
| 1 | *Mayor a 80'000.000,00* |
| 2 | *Mayor a 20'000.000,00 hasta 80'000.000,00* |
| 3 | *Mayor a 5'000.000,00 hasta 20'000.000,00* |
| 4 | *Mayor a 1'000.000,00 hasta 5'000.000,00* |
| 5 | *Hasta 1'000.000,00* |
| | *Cajas de Ahorro, bancos comunales y cajas comunales* |

Figure 10: Segmentación de Cooperativas y Mutualistas en base a los activos
*Fuente.* https://www.seps.gob.ec/estadistica?boletin-financiero-sf-y-snf

La SEPS dispone de una dirección web desde donde se puede obtener información de las cooperativas y mutualistas, esta se convierte en otra fuente de información desde la cual se procede a extraer los balances generales, como se visualiza en la siguiente imagen:

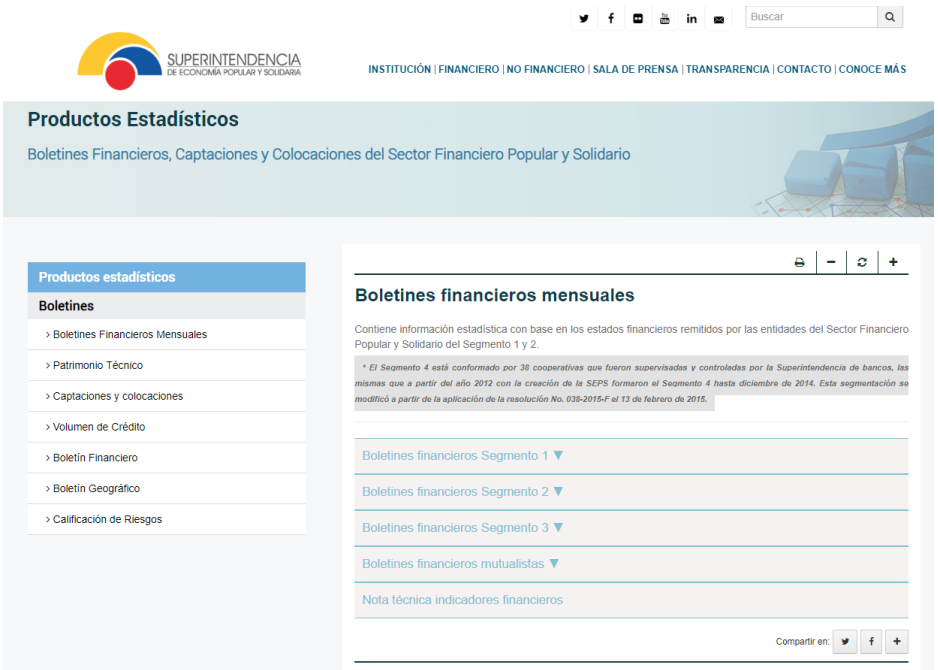



Figure 11: Boletínes Mensuales SEPS
*Fuente.* https://www.seps.gob.ec/estadistica?boletines-financieros-mensuales

El formato de información que encontramos en este sitio es el de un archivo Excel (xlsm) para el caso de fechas actuales, pero para consultas que van desde el año 2015 hacia atrás, estos se encuentran empaquetados en un comprimido de formato .zip, en donde los datos se encuentran consolidados y no en detalle como se tiene en los archivos actuales, cabe mencionar que dentro del archivo de Excel, esta se subdivide en distintas hojas, la que interesa es la tercera, ya que en esa se encuentra los balances generales, y que se procederá a leer y extraer.

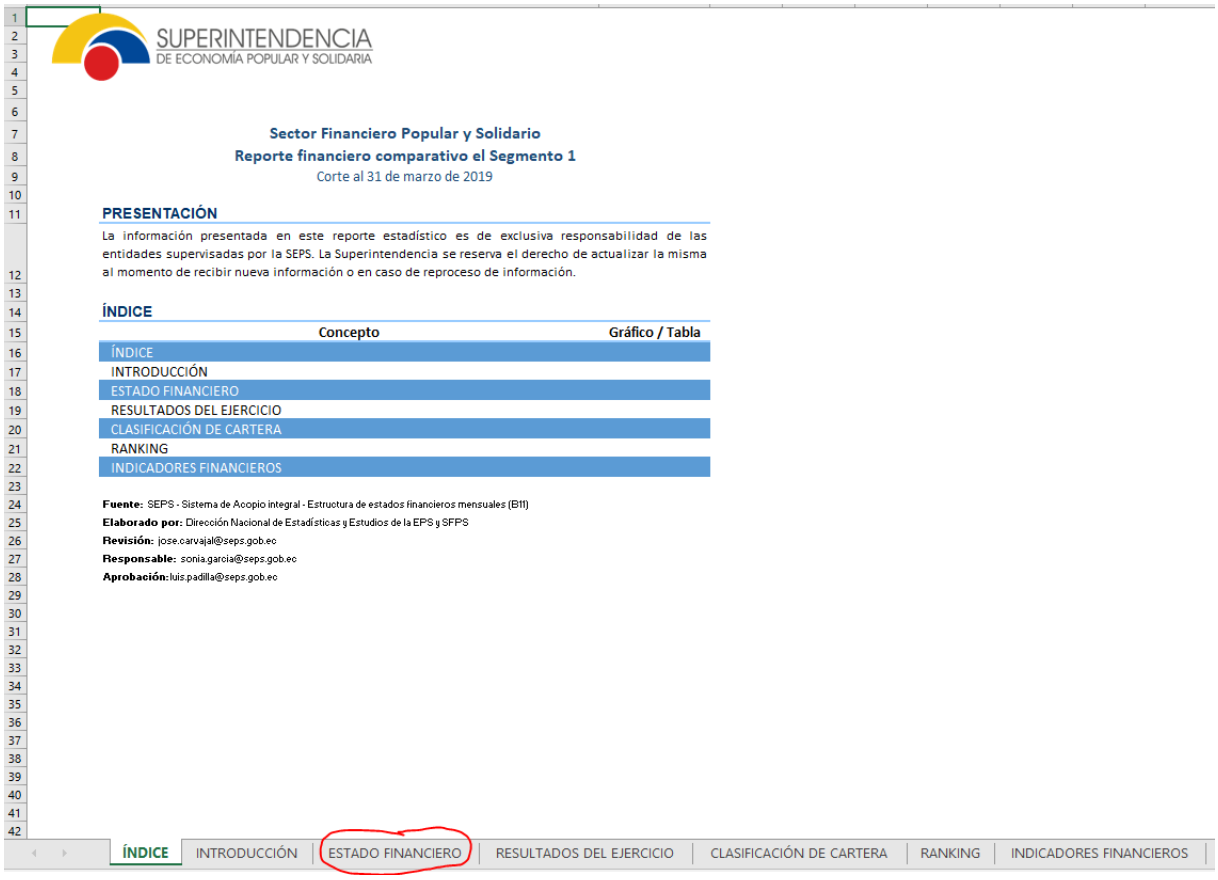
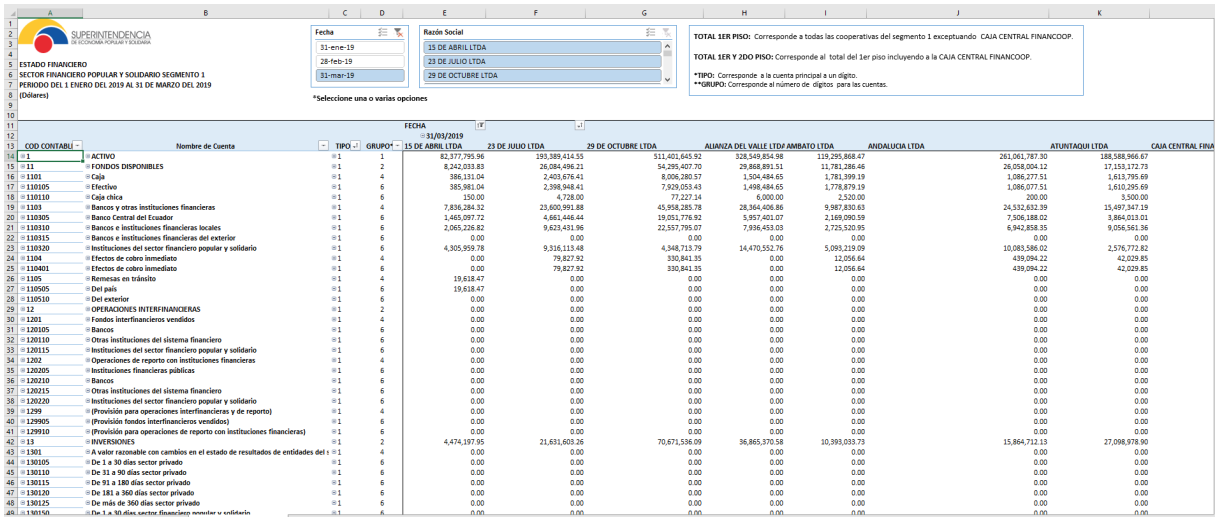



Figure 12: Archivo Excel (xlsm)
*Fuente.* https://www.seps.gob.ec/estadistica?boletines-financieros-mensuales

En la pestaña de estados financieros, se encuentra la información de las cooperativas que se encuentran dentro del segmento que se esta consultado. En la imagen siguiente se tiene la información de los estados financieros del segmento 1.



SUPERINTENDENCIA DE ECONOMÍA POPULAR Y SOLIDARIA

ESTADO FINANCIERO
SECTOR FINANCIERO POPULAR Y SOLIDARIO SEGMENTO 1
PERIODO DEL 1 ENERO DEL 2019 AL 31 DE MARZO DEL 2019
(Dólares)

Fecha: 31-ene-19, 28-feb-19, 31-mar-19
*Seleccione una o varias opciones

Razón Social: 15 DE ABRIL LTDA, 23 DE JULIO LTDA, 29 DE OCTUBRE LTDA

**TOTAL 1ER PISO:** Corresponde a todas las cooperativas del segmento 1 exceptuando CAJA CENTRAL FINANCOOP.

**TOTAL 1ER Y 2DO PISO:** Corresponde al total del 1er piso incluyendo a la CAJA CENTRAL FINANCOOP.

*TIPO: Corresponde a la cuenta principal a un dígito.
**GRUPO: Corresponde al número de dígitos para las cuentas.

FECHA: 31/03/2019

| COD CONTABLE | Nombre de Cuenta | TIPO | GRUPO | 15 DE ABRIL LTDA | 23 DE JULIO LTDA | 29 DE OCTUBRE LTDA | ALIANZA DEL VALLE LTDA | AMBATO LTDA | ANDALUCIA LTDA | ATUNTAQUI LTDA | CAJA CENTRAL FINA |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | ACTIVO | 1 | 1 | 82,377,795.96 | 193,389,414.55 | 511,401,645.92 | 328,549,854.98 | 139,295,868.47 | 261,061,787.30 | 188,588,966.67 | |
| 11 | FONDOS DISPONIBLES | 1 | 2 | 8,242,033.83 | 26,084,496.21 | 54,295,407.70 | 29,868,891.51 | 11,781,286.46 | 26,058,004.12 | 17,153,172.73 | |
| 1101 | Caja | 1 | 4 | 386,131.04 | 2,403,676.41 | 8,006,280.57 | 1,504,484.65 | 1,781,399.19 | 1,086,277.51 | 1,613,795.69 | |
| 110105 | Efectivo | 1 | 6 | 385,981.04 | 2,398,948.41 | 7,929,053.43 | 1,498,484.65 | 1,778,879.19 | 1,086,077.51 | 1,610,295.69 | |
| 110110 | Caja chica | 1 | 6 | 150.00 | 4,728.00 | 77,227.14 | 6,000.00 | 2,520.00 | 200.00 | 3,500.00 | |
| 1103 | Bancos y otras instituciones financieras | 1 | 4 | 7,836,284.32 | 23,600,991.88 | 45,958,285.78 | 28,364,406.86 | 9,987,830.63 | 24,532,632.39 | 15,497,347.19 | |
| 110305 | Banco Central del Ecuador | 1 | 6 | 1,465,097.72 | 4,661,446.44 | 19,051,776.92 | 5,957,401.07 | 2,169,090.59 | 7,506,188.02 | 3,864,013.01 | |
| 110310 | Bancos e instituciones financieras locales | 1 | 6 | 2,065,226.82 | 9,623,431.96 | 22,557,795.07 | 7,936,453.03 | 2,725,520.95 | 6,942,858.35 | 9,056,561.36 | |
| 110315 | Bancos e instituciones financieras del exterior | 1 | 6 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | |
| 110320 | Instituciones del sector financiero popular y solidario | 1 | 6 | 4,305,959.78 | 9,316,113.48 | 4,348,713.79 | 14,470,552.76 | 5,093,219.09 | 10,083,586.02 | 2,576,772.82 | |
| 1104 | Efectos de cobro inmediato | 1 | 4 | 0.00 | 79,827.92 | 330,841.35 | 0.00 | 12,056.64 | 439,094.22 | 42,029.85 | |
| 110401 | Efectos de cobro inmediato | 1 | 6 | 0.00 | 79,827.92 | 330,841.35 | 0.00 | 12,056.64 | 439,094.22 | 42,029.85 | |
| 1105 | Remesas en tránsito | 1 | 4 | 19,618.47 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | |
| 110505 | Del país | 1 | 6 | 19,618.47 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | |
| 110510 | Del exterior | 1 | 6 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | |
| 12 | OPERACIONES INTERFINANCIERAS | 1 | 2 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | |
| 1201 | Fondos interfinancieros vendidos | 1 | 4 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | |
| 120105 | Bancos | 1 | 6 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | |
| 120110 | Otras instituciones del sistema financiero | 1 | 6 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | |
| 120115 | Instituciones del sector financiero popular y solidario | 1 | 6 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | |
| 1202 | Operaciones de reporto con instituciones financieras | 1 | 4 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | |
| 120205 | Instituciones financieras públicas | 1 | 6 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | |
| 120210 | Bancos | 1 | 6 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | |
| 120215 | Otras instituciones del sistema financiero | 1 | 6 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | |
| 120220 | Instituciones del sector financiero popular y solidario | 1 | 6 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | |
| 1299 | (Provisión para operaciones interfinancieras y de reporto) | 1 | 4 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | |
| 129905 | (Provisión fondos interfinancieros vendidos) | 1 | 6 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | |
| 129910 | (Provisión para operaciones de reporto con instituciones financieras) | 1 | 6 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | |
| 13 | INVERSIONES | 1 | 2 | 4,474,197.95 | 21,631,603.26 | 70,671,536.09 | 36,865,370.58 | 10,393,033.73 | 15,864,712.13 | 27,098,978.90 | |
| 1301 | A valor razonable con cambios en el estado de resultados de entidades del s | 1 | 4 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | |
| 130105 | De 1 a 30 días sector privado | 1 | 6 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | |
| 130110 | De 31 a 90 días sector privado | 1 | 6 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | |
| 130115 | De 91 a 180 días sector privado | 1 | 6 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | |
| 130120 | De 181 a 360 días sector privado | 1 | 6 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | |
| 130125 | De más de 360 días sector privado | 1 | 6 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | |
| 130150 | De 1 a 30 días sector financiero popular y solidario | 1 | 6 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | |

Figure 13: Estados Financieros de Cooperativas y Mutualistas S1
*Fuente.* https://www.seps.gob.ec/estadistica?boletines-financieros-mensuales

Como vemos el proceso de extraer los datos de las cooperativas resulta largo, puesto que se debe primero descargar el archivo, y luego proceder a leerlo y extraer la información presente en la hoja 3 del documento.

Para concluir esta sección de Fuentes de Información, resumiendo se tienen dos orígenes en donde tenemos los datos en distintos formatos; actualmente la lectura la realizan manual, luego lo almacenan en archivos de Excel para después migrarlos a una base de datos, pues ese es uno de los objetivos del proyecto, facilitar la extracción de esta información y agruparlos en un solo repositorio de datos logrando así una mejor gestión, tener datos históricos, realizar análisis y sobre todo ser la base para poder construir el Sistema de Información Gerencial.

## 2.4. Sistemas de Información Gerencial para la toma de decisiones

### 2.4.1. Definiciones Sistema de Información Gerencial

Se puede definir al sistema de información como una estructura de decisión y una actividad de transformación el cuál esta basado en conjunto de reglas de gestión y de información. (Otero & Huerga, 2019).

Otra definición planteada por Novatrix (2015) afirma: "Los sistema de información gerencial y la tecnología agilizan los procesos de las empresas y acortan tiempos, mejoran sus finanzas y aspectos administrativos, además de generar mayor productividad".

Un sistema de información gerencial tiene muchos aspectos positivos para una empresa, de hecho este puede ser aplicado a las distintas áreas, apoyando enormemente en el desarrollo de las actividades de toma de decisiones que sean necesarias ejecutar, pero en este sentido bajo un sustento teórico y técnico tomado de la información que la misma empresa genera.

### 2.4.2. ¿Cómo iniciar proyectos de Sistema de Información?

Es importante tener claro las razones que impliquen inicializar la construcción de un sistema de información. Rodríguez (2009) afirma: "La razón fundamental en el desarrollo de sistemas de información es que las aplicaciones son una herramienta y no un instrumento que debe de tenerse para utilizar la tecnología de la información; en consecuencia los sistemas de información deben de desarrollarse sobre la base de su propia capacidad para mejorar el desempeño de la organización, sin embargo estas razones no significan únicamente pérdidas y ganancias. La marcha de una empresa incluye también beneficios para sus empleados, clientes y otras personas con la que se tienen tratos" (p.5).

Rodríguez (2009) plantea 5 razones para implementar un proyecto de sistema de información gerencial, el cuál brinda una guía para empezar la construcción de un SIG:

- **Capacidad:** esta razón se asocia inherentemente a la capacidad de poder extraer información a una velocidad mucho mayor y de distintas fuentes que a un ser humano le resultaría complejo, y esta extracción de información implica un incremento en el volumen de los datos, por lo tanto se necesita de mayor velocidad en los procesamientos que por obvias razones nos brindan las computadoras, es decir la capacidad computacional.
- **Control:** como se tiene mayor información se puede tener un control en la exactitud y consistencia de los datos y proveer una mayor seguridad de los mismos dando acceso únicamente al personal autorizado.
- **Comunicación:** esta es una de las razones que en lo personal pienso es importante aclarar, puesto que un sistema de información brinda un adecuado flujo de la información entre las distintas áreas de una organización y de forma acelerada brindando una integración de las distintas áreas.
- **Costos:** un sistema de información permite tener un control de los costos que se tiene comúnmente en una empresa como son bienes, empleados, clientes, etc. y como se

podría reducir los costos en base a la información recolectada.

- **Competitividad:** esta razón esta ligada a la capacidad de una empresa para aprovechar la información y poder conseguir una ventaja competitiva mediante la atracción de clientes, mejorando la relación con los proveedores, desarrollando nuevos productos y muchas otras que puedan permitir el cumplimiento de los objetivos.

Un sistema de información debe brindar una ventaja competitiva en base a datos que la misma empresa genera, presentándose de forma clara y concisa atándola a los objetivos estratégicos. Debemos entender que el Sistema de Información Gerencial es usado para la toma de decisiones tratando de responder a preguntas como el estado actual y futuro de la organización, al control diario de operaciones, a la planeación táctica en el control administrativo y a la definición de políticas en niveles altos de gerencia (Tejeda Yépez, 2018).

La idea de crear un Sistema de Información Gerencial con la información de las instituciones financieras, es para brindarles a cada organización en este caso Bancos, Cooperativas y Mutualistas información que pueda darles una ventaja competitiva, ya que, al tener datos de todo su entorno financiero, puedan tomar adecuadas decisiones, y de esta manera crear oportunidades como por ejemplo la innovación de nuevos procesos internos o apoyando a la planeación de objetivos estratégicos, siempre centrados en el uso de nuevas tecnologías de información.

### 2.4.3. El Big Data y su relación con el mundo financiero

Pienso que en el mundo financiero es en donde por defecto se debería explotar todo lo relacionado a big data, debido a la gran cantidad de información que se genera a cada segundo por la transaccionalidad que produce cada cliente de una institución financiera.

El big data tiene distintas definiciones, pero en lo personal se refiere a capturar y administrar mucha información, dando paso a la creación de nuevos roles en las empresas como el científico de datos, que con herramientas potentes que existen hoy en día, facilite su manipulación y entendimiento.

Muchas empresas ven al big data como una herramienta tecnológica, mas no como el core del negocio. Las empresas se basan en sus avances tecnológicos y se encierran en un mundo donde lo importante es la transaccionalidad mas no en aprovechar los datos que sus sistemas generan, es necesario realizar un cambio de mentalidad y entender que se le debe

dar importancia a los datos y crear áreas especializadas dentro de una organización que las exploten. (Gadi, 2015).

Pensar que una institución financiera no aproveche los datos que genera es porque desconoce, su personal no esta capacitado para manejar y explotar esta información o simplemente no esta dentro de los objetivos estratégicos de la empresa, esa es la razón de un cambio de mentalidad. Por ejemplo, las instituciones financieras han creado sistemas complejos que miden la capacidad de pago o devolución de deudas de los clientes, pero esta es solo una pequeña parte del total de datos que se tienen, ya que hoy en día existe información que se encuentra libre y disponible en la internet, debido a que son los mismos usuarios que la comparten (Gadi, 2015).

Otro tema que es fundamental, es el reconocer que información de toda esta cantidad es relevante, de hecho pienso que ese es el objetivo del área de big data junto con la inteligencia de negocio, proveer soluciones que se adapten al entorno en el cuál operan, haciendo uso de metodologías que vayan desde la recolección de los datos hasta la presentación para que la organización pueda aprovechar y tomar acciones que generen una ventaja competitiva, ayudándoles a mejorar sus procesos y a la integración total de cada área, ya que cada una tendrá la información a su alcance.

Algunos ejemplos de proyectos big data en instituciones financieras tenemos por ejemplo fidelización de clientes en base a comentarios que estos dejan en las redes sociales, o análisis de comportamiento de los clientes en cuanto a su flujo de captaciones y poder ofrecerle nuevos productos o servicios y así premiar a su confianza, otro ejemplo podría ser en el de llevar a cabo procesos automáticos de aprendizaje en donde se pueda segmentar a los clientes en base a su comportamiento financiero, con la finalidad de que se pueda dar un trato personalizado por clientes, etc. las posibilidades de aprovechar al big data son variadas, y como indique al inicio de la sección, en una institución financiera es en donde más información se tiene para poder ejecutar esta clase de proyectos.

Al punto donde se desea llegar con esta sección es que se vea la razón del porque del planteamiento de este proyecto de máster, ya que desde mi punto de vista, veo al big data como core principal que apoyará a las instituciones financieras, ya que a partir de este, se puede presentar proyectos que analistas de datos en conjunto con el área de negocio pueden plantearse, uno de estos fue el de construir el sistema de información gerencial, el cuál se aprovecha de esta información recolectada, se puede pensar por el tamaño del proyecto como

un software sencillo, pero la visión que se tiene va mas allá de solo tener un sistema que consuma los datos recolectados.

## 2.5. Herramientas de almacenamiento y procesamiento big data

En esta sección se estudia las distintas herramientas que se han estudiado a lo largo del máster de Análisis y Visualización de Datos Masivos, de estos una parte fueron usados en el desarrollo del proyecto de TFM, el resto se los incluye para poder revisar las bondades que cada tecnología brinda y como pueden apoyar a realizar análisis de los datos.

### 2.5.1. Apache Hadoop

En R. D. Schneider (2012) se define a Apache Hadoop como: "Un marco de software de código abierto, bien adoptado y basado en estándares, construido sobre la base de los documentos MapReduce y Google File System de Google. Está pensado para aprovechar el poder del procesamiento en paralelo para procesar datos masivos, generalmente mediante el uso de gran cantidad de servidores de productos básicos de bajo costo" (p.12)

Hadoop brinda una arquitectura abierta, es decir tiene una compatibilidad con estándares abiertos y muchos lenguajes de programación, lo cuál es bueno para una empresa, ya que Hadoop se adapta a una infraestructura económica, es empleado como un almacén de datos y proporciona un independencia y flexibilidad en la extracción de la información, por lo tanto se debe considerar plantear una solución de alta disponibilidad, alto rendimiento, escalabilidad, predictibilidad y que permita trabajar con grandes cantidades de datos (R. D. Schneider, 2012).

#### Ecosistema de Apache Hadoop

No nos adentraremos a estudiar a fondo Hadoop puesto que como menciono es una plataforma tan potente que no se lo estaría explotando como se en este proyecto de TFM, sin embargo resumo brevemente para entender los componentes que lo conforman: Librerías y utilidades para integrar Hadoop con otros módulos denominado ***Hadoop Common***, el sistema de archivos distribuidos ***HDFS***, el motor de procesamiento ***MapReduce***, un gestor de los recursos del clúster llamado ***YARN*** y una base de datos no relacional que se integra con *HDFS* cuyo nombre es ***HBase*** (R. D. Schneider, 2012).

De estos componentes haré hincapié en dos que a mi parecer merecen ser brevemente

explicados que son HDFS y YARN, cada uno tiene su importancia dentro de un clúster Hadoop y de hecho son los elementos con los que se integra cualquier otro motor de procesamiento como es el caso de Apache Spark que se trata en secciones posteriores.

**Arquitectura HDFS**

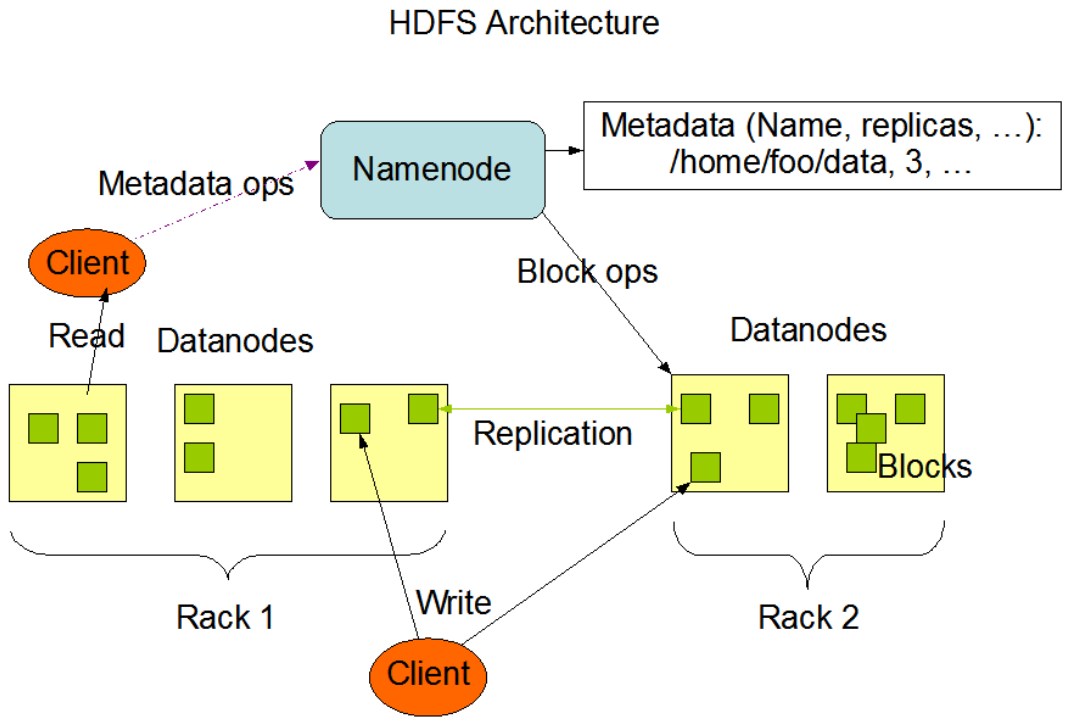



Figure 14: Arquitectura de HDFS.
*Fuente.* «HDFS Architecture» (2018)

De la figura anterior 14 podemos observar que se trata de una arquitectura máster/esclavo. Aquí cada componente es un nodo, se tiene un nodo máster que recibe el nombre de *Namenode* y los esclavos que reciben el nombre de *Datanode*. El Namenode administra el espacio de nombres del sistema de ficheros y controla los accesos a los archivos por parte de las aplicaciones cliente. El Datanode almacena los datos de los usuarios y puede existir mas de uno. Los datos se dividen en bloques que son distribuidos en distintos nodos de datos, esta división de los datos y su posterior almacenamiento son instruidos por el namenode el cuál almacena metadatos que indican la ubicación de los datos y el nodo a donde debe acceder al momento de leer la información. («HDFS Architecture», 2018).

Por lo general un nodo es una máquina o servidor que ejecuta un sistema operativo Linux y que dispone de la plataforma Java, de hecho Hadoop fue desarrollado en Java, pueden estar dispuestos en un mismo rack de servidores o en distintos racks, por lo que se entiende que

existirá un alto consumo en la red, por lo que es un punto a tomar en cuenta a la hora de implementar un clúster Hadoop.

Por defecto, Hadoop maneja un factor de replicación 3 es decir que por un bloque de datos este se replica 3 veces en 3 nodos diferentes consiguiendo una integridad a los datos, este factor de replicación es un valor que puede ser cambiado dependiendo de la cantidad de nodos que se disponga y se encuentra en los archivos de configuración de Hadoop [Figura 15]. Cabe mencionar que la arquitectura HDFS no impide que existan distintos datanodes en una misma máquina pero claramente esto no sucede en una implementación real («HDFS Architecture», 2018).

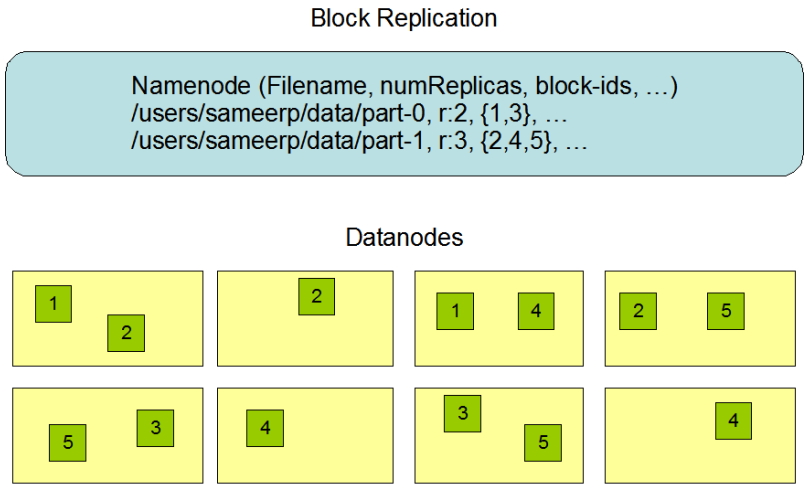



Figure 15: Replicación de Bloques
*Fuente:* «HDFS Architecture» (2018)

Puede existir varios datanodes pero solo existe un namenode esto se debe a la necesidad de simplificar la arquitectura pero implica tener un punto de fallo único, por lo que se debe considerar la implementación de otro clúster en caso de fallo del namenode (alta disponibilidad), en cuanto a los datanodes si uno falla, el resto continua su procesamiento teniendo al clúster funcionando adecuadamente («HDFS Architecture», 2018).

El sistema de archivos es muy similar al de Linux o Windows en donde el usuario puede crear, editar o eliminar directorios y ficheros. Permite la creación de una estructura jerárquica de directorios y ofrece seguridad por medio de permisos de acceso y cuotas de usuario («HDFS Architecture», 2018).

Si se desea aprender más sobre la plataforma se puede referir a la documentación que se tiene en la página oficial (Apache Hadoop. http://hadoop.apache.org/docs/current/).

### 2.5.2. Apache Hadoop Yarn

YARN nace con la idea de dividir las funcionalidades de la administración de los recursos y la programación de trabajos en procesos separados, con esto se consigue que se pueda ejecutar procesos de otras plataformas ajenas a Hadoop sobre un clúster Hadoop. De hecho Apache YARN fue una actualización a Hadoop que de igual forma dispone de una arquitectura en donde se tiene a un *Resource Manager (RM)* y a varios *Application Master (AM)* dados por aplicación (simple o compuesta). («YARN Architecture», 2019).

El *RM* gestiona los recursos de las aplicaciones. Cada nodo esclavo además de tener al *AM* tiene a un responsable del contenedor, en donde supervisa cpu, memoria, disco, red y mantiene comunicado del estado al *RM*, denominado *Node Manager (NM)*. En la comunicación entre el *RM* y *NM* se da un proceso de negociación de recursos que es supervisado por el *AM* («YARN Architecture», 2019).

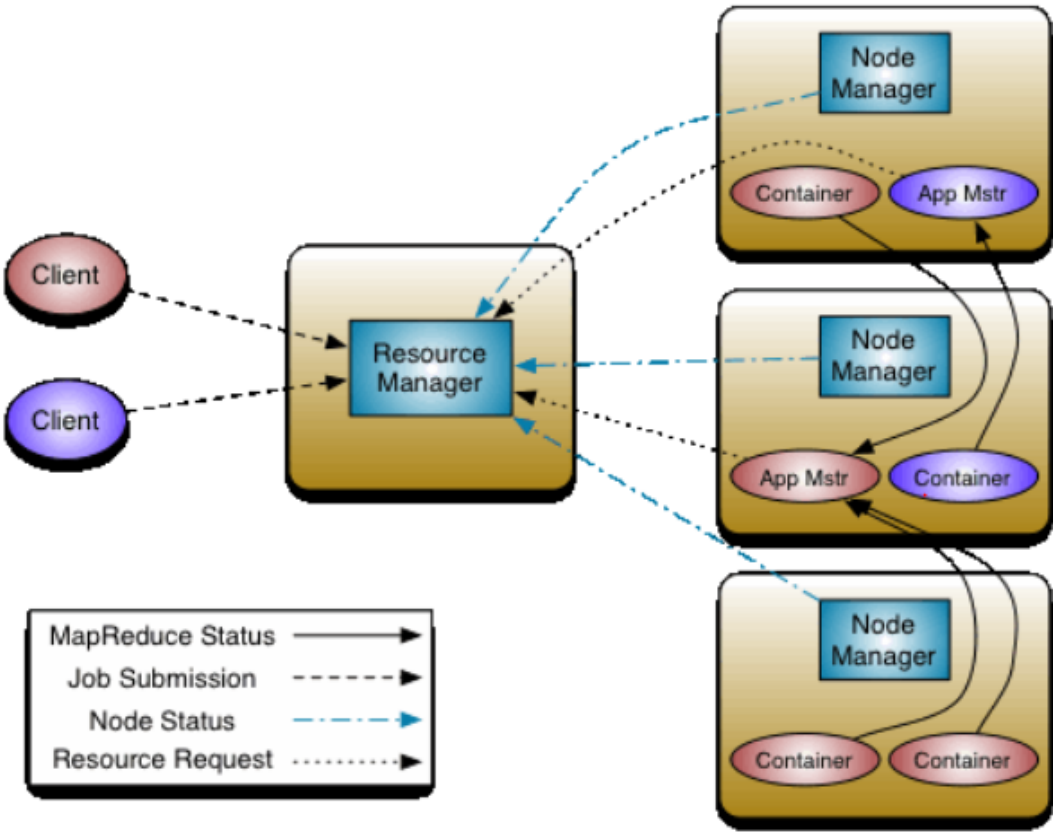



Figure 16: Arquitectura YARN
*Fuente.* «YARN Architecture» (2019)

El *RM* tiene dos componentes importantes que es el ***Scheduler*** que asigna los recursos

a cada aplicación, pero no realiza un control o monitoreo del estado de cada aplicación, para ello se tiene al otro componente que es el ***Application Manager*** que negocia los contenedores y reinicia el contenedor en caso de falla, por lo que tiene una comunicación con el Scheduler por cada ejecución de aplicación («YARN Architecture», 2019).

Como vemos del párrafo anterior existe una ejecución de una aplicación dentro de un contenedor, por lo que en YARN existe un concepto conocido como noción de reserva que garantiza la ejecución predecible ya que se reserva dinámicamente recursos, analizando el comportamiento de una aplicación a lo largo del tiempo («YARN Architecture», 2019).

Otra característica que vale mencionar de Yarn es que favorece trabajar con miles de nodos de tal forma que se pueda conectar varios pero para Yarn este es transparente, ya que aparecen como si se tratara de un solo clúster, favoreciendo una escala de múltiples trabajos independientes, a esto se le conoce como Federación («YARN Architecture», 2019).

Si se desea aprender más sobre la plataforma se puede referir a la documentación que se tiene en la página oficial (Apache YARN. https://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html).

### 2.5.3. Apache Spark

Como vimos en la sección anterior, con el surgimiento de Apache Yarn dio paso a una serie de soluciones que apoyaron a los desarrolladores en su hambre de crear aplicaciones que se integren con el ecosistema de Hadoop, de esto surgió Apache Hive, Apache Drill, Apache Pig, Apache Presto y muchas más pero la que nos compete revisar ahora es **Apache Spark** que es la herramienta que se usó en el trabajo de fin de máster.

Apache Spark ha ganado una enorme popularidad debido a la facilidad con la que un desarrollador puede adaptarse a este esquema, además que contiene una compatibilidad con distintos lenguajes de programación y herramientas de alto nivel para el procesamiento de datos estructurados («Apache Spark», 2019).

Una de las grandes ventajas de la herramienta es que es un clúster de procesamiento rápido y de propósito general que nació con la idea de reemplazar el procesamiento por lotes que ofrece el motor MapReduce integrado en Hadoop, debido a que este hace uso de la memoria en la ejecución de aplicaciones, si se compara un mismo proceso de ejecución entre MapReduce y Spark vemos que este lo realiza hasta 100 veces más rápido (R. Schneider &

Karmiol, 2019).

Una organización puede beneficiarse enormemente con Apache Spark ya que si se lo implementa se puede tener un buen retorno de la inversión por gastos de Spark, además de mencionar que es un software libre y que no es obligatorio que se lo integre con Hadoop. Una solución con Spark mejora los tiempos de ejecución, y el tiempo es un aspecto importante en una empresa (R. Schneider & Karmiol, 2019).

Spark permite integrarse con distintas bases de datos de lenguaje estructurado (SQL) y no estructurado (NoSQL), además que el resultado de Spark se puede integrar en herramientas de Business Intelligence de distintas empresas como Tableau, QlickSense, Microsoft, etc. Otra razón de usar Spark es que el resultado también puede ser aplicado en hojas de cálculo que de hecho es una herramienta muy utilizada en empresas (R. Schneider & Karmiol, 2019).

Apache Spark permite adaptarse a un clúster Hadoop, es por eso la importancia de entender que no es un reemplazo a todo el ecosistema de Hadoop, de hecho se puede decir que Spark es un reemplazo a uno de los componentes de Hadoop como es MapReduce, pero el componente HDFS se lo sigue usando; en la práctica se usa Spark para leer y escribir datos en HDFS.

**Spark SQL**

En «Apache Spark» (2019) nos muestran las distintas librerías de trabajo: Spark Streaming, MLIB (Machine Learning), GraphX (Gráfos) y Spark SQL/Dataframes. Esta última se revisa en esta sección y se hace hincapié ya que es la que fue empleada en el trabajo de fin de máster.

Spark SQL es un módulo que permite trabajar con datos estructurados mediante consultas SQL, para ello trabaja con un conjunto de datos denominado **Dataframes** que es similar al concepto manejado por el lenguaje R, y es equivalente a manipular tablas, en donde los registros se presentan en columnas y filas, y cada columna hace referencia al atributo de los datos (R. Schneider & Karmiol, 2019).

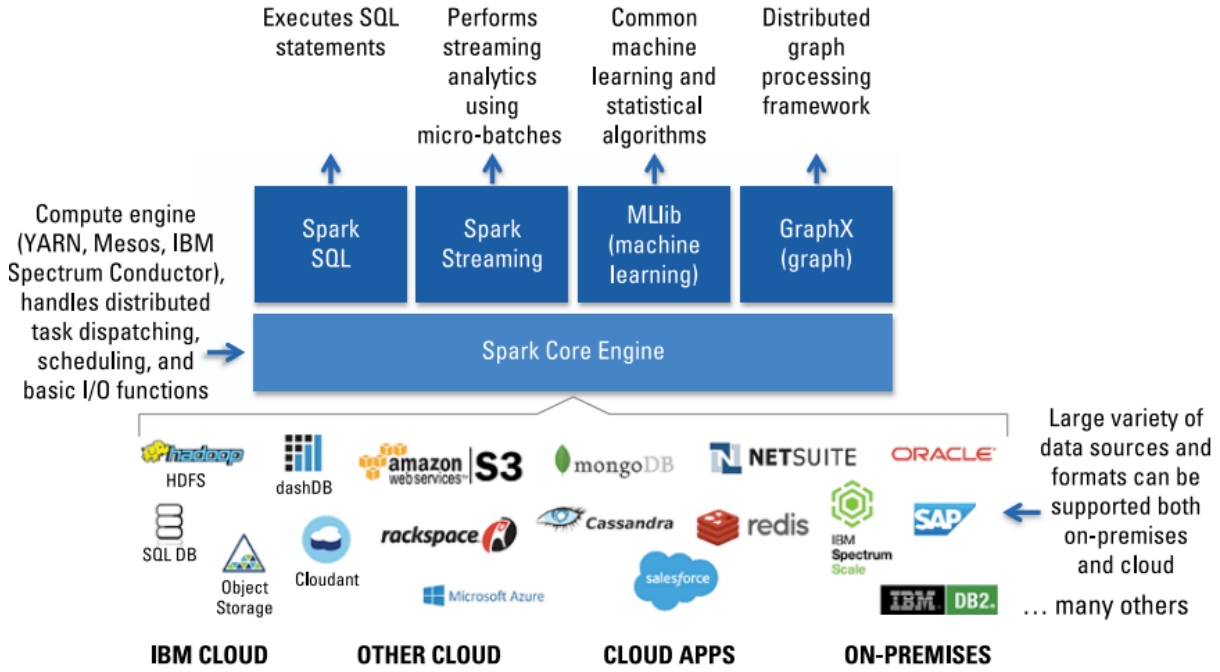



Figure 17: Ecosistema de Apache Spark
*Fuente.* R. Schneider and Karmiol (2019)

## 2.6. Herramientas de desarrollo de aplicaciones big data

Para el desarrollo del trabajo de fin de máster se trabajó con distintas herramientas que facilitaron la construcción del proyecto, si bien se los describe a breve rasgos, puesto que no es el objetivo central del proyecto de fin de máster, se invita al lector a revisarlos y estudiarlos con un mayor detenimiento, porque son herramientas que al día de hoy se adaptan fácilmente al desarrollo de soluciones big data.

### 2.6.1. Lenguaje R

En lo personal no lo considero únicamente como un lenguaje de programación sino también como un entorno de software, similar a realizar actividades en el programa Excel, ya que incluye funciones, que ofrecen realizar cálculos y gráficas de forma inmediata, sin necesidad de crear todo una aplicación para ello.

Es un software libre multiplataforma, se lo tiene para Windows, Linux y Mac y puede ser descargado directamente de la página oficial, para el trabajo de fin de máster el ambiente de desarrollo es Windows por lo que el enlace de descarga es https://cran.r-project.org/bin/

windows/base/.

Además de instalar el framework de R, se recomienda descargar el IDE para trabajar con el lenguaje y es **RStudio** el cuál brinda un ventana muy fácil y útil para el desarrollado, en donde se tiene el área de trabajo, una sección de depuración de variables, gráficas, paquetes y el directorio del proyecto [Figura. 18].

El lenguaje R fue usado para la etapa de extracción de la información, el cuál basado en una serie de paquetes permitió acceder a ubicaciones web y descargarse la información ya sea aplicando un web scrapping o una descarga de ficheros y lectura de los mismos.

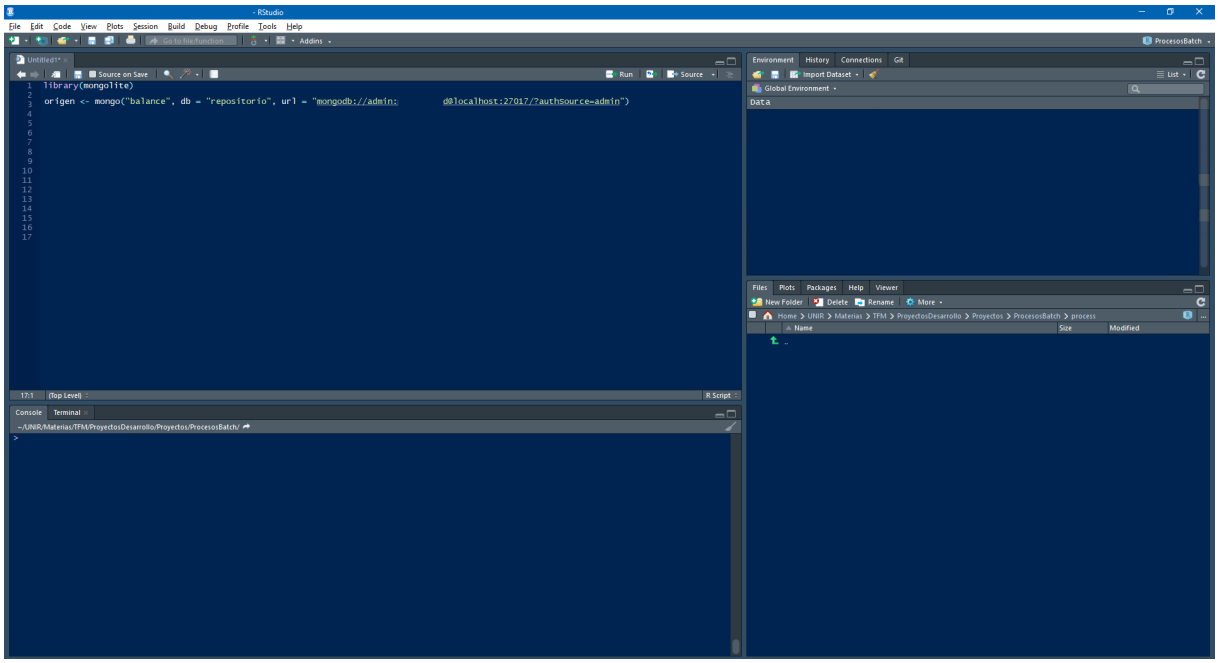



Figure 18: Entorno RStudio
*Fuente.* Elaboración propia

### 2.6.2. Lenguaje Python

Python es un lenguaje relativamente nuevo, pero que ha ganado mucha popularidad por lo fácil de aprender y usar, además que es flexible. Es considerado un lenguaje de alto nivel e incluye lo que normalmente se tiene en un lenguaje como es el tipo de estructura de datos, flujo de ejecución, funciones, etc.

Tomando como referencia a «Python» (2014), este lo define como un lenguaje interpretado y fuertemente tipado, multiplataforma y ofrece una programación multiparadigma ya sea programación orientada a objetos, estructurado, imperativa e incluso funcional.

De igual forma que tenemos con R, invitó al lector a investigar más sobre Python, ya que dispone de herramientas muy potentes para el análisis de datos, de hecho Python fue usado para realizar el procesamiento de los datos en el trabajo de fin de máster ya que permite incluir la librería de Apache Spark denominada PySpark con la cual nos conectamos a un clúster Spark y a las distintas fuentes de datos en este caso MongoDB y usamos el componente Spark SQL para trabajar como si estuviéramos tratando con datos estructurados pese a que se toma datos de una base NoSQL.

Spark junto a Python ofrecen un potente marco de trabajo ya que por cada necesidad de análisis o consolidación de datos se puede crear distintos archivos (.py) y empaquetarlos en uno solo y subirlo a un clúster Spark de producción, en Spark basta con indicar que archivo .py se debe ejecutar.

Se puede descargar la última versión estable que se tiene al momento de realizar este documento 3.7.3, como indique anteriormente el ambiente de desarrollo es Windows por lo que se descarga la versión correspondiente del siguiente enlace https://www.python.org/downloads/.

Adicional, para hacer uso de Python se lo puede realizar directamente desde la ventana de comandos en Windows o una terminal de Linux o Mac, o por medio de un entorno de trabajo (IDE), de los cuales existen varios como por ejemplo PyCharm, Anaconda, Eclipse, PyDev, IPython, etc. el que empleo en el TFM es Visual Studio Code.

Visual Studio Code ofrece una integridad con distintos lenguajes de programación como es Java y de desarrollo web como es Javascript, Angular. Además, incluye una terminal ya sea esta de Windows o de Python inclusive. El entorno es simple, dispone del área de trabajo, el explorador de archivos, la salida de la terminal, depuración, errores, también un gestor de extensiones, ventana de depuración, control de versiones, todo esto se puede visualizar en la siguiente imagen:

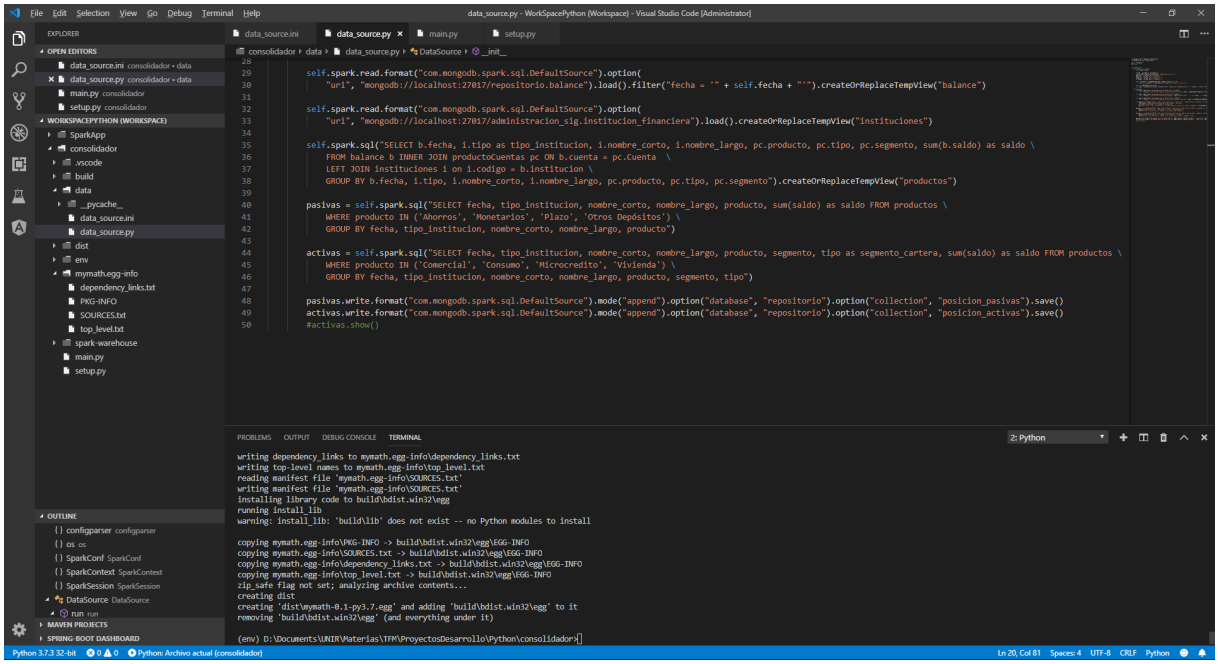



Figure 19: Entorno Visual Studio Code
*Fuente.* Elaboración propia

### 2.6.3. Lenguaje Java

El lenguaje Java no necesita presentación puesto que es un framework de trabajo conocido a nivel mundial y ampliamente usado en distintos ámbitos empresariales, educacionales, salud, etc. Inicialmente fue creado por Sun Microsystems, pero luego este fue adquirido por la empresa Oracle, la sintaxis de Java es similar a la de C o C++, comparado con los lenguajes anteriormente mencionados, la curva de aprendizaje es mayor, puesto que es un lenguaje compilado, orientado a objetos, con un tipado estático, robusto e interpretado, además que puede ejecutarse en distintos tipos procesadores (Alicante, 2012).

Java es robusto debido a que incluye un conjunto de paquetes integrado muy potente, a diferencia de los otros que deben ser descargados, entre estos tenemos la gestión de hilos, ejecución remota, etc. (Alicante, 2012).

No entraré en detalle en el estudio de Java ya que se tiene mucha información disponible en la web e incluso libros enteros que hablan del tema pero vale mencionarlo puesto que este lenguaje fue usado para la construcción de la lógica del Sistema de Información. Existe un framework de trabajo muy potente conocido como *Spring Boot* que permite crear y levantar API's de forma rápida y sencilla, con esto consigo construir servicios que permitió a la aplicación web la consulta de los datos. Spring Boot se integra con Java y rápidamente genera un

CRUD de la estructura MongoDB en objetos manipulables directamente en Java.

El entorno de desarrollo usado es el mismo Visual Studio Code descrito en la sección del lenguaje Python.

## 2.7. Herramientas de desarrollo web

Construir una aplicación web resulta tan llamativa como crear una aplicación de escritorio lo cual hace un tiempo atrás era impensable, prácticamente las páginas web mostraban información estática y únicamente texto. Hoy en día las aplicaciones web son tan ricas en contenido que permiten una mayor interacción del usuario, y con la potencia que las empresas han dado a los navegadores web ya sea Google Chrome, Microsoft Edge, Safari, Mozilla, etc. crear aplicaciones web depende de la imaginación que tengan los desarrolladores.

De todo el potencial que despegó por parte de los desarrolladores en la web, el que más se ha beneficiado ha sido **Javascript**, antes se lo usaba para procesos de validaciones en las entradas de los usuarios, con la estandarización de los navegadores y el tratamiento que han dado las empresas encargadas de su gestión, dieron paso a que Javascript se vuelva un estándar necesario en el desarrollo web, la última versión de Javascript a la fecha es la ECMAScript 2019 (Mozilla, 2019).

En el desarrollo del sistema de información gerencial se hace un amplio uso del lenguaje Javascript pero empleando un framework de trabajo denominad Angular y una librería para manipulación de elementos HTML como es D3.js.

### 2.7.1. Angular

Nació como proyecto de Google, cuya primera versión se baso en Javascript puro, pero el mantenimiento y actualización de la plataforma resultaba compleja por lo que dieron paso a que se tenga una reestructuración completa, lo que llevo a tener una versión rediseñada denominada Angular 2.

La segunda versión de Angular implico hacer un cambio del lenguaje que paso de Javascript a Typescript, el cuál permite elaborar aplicaciones a gran escala para cualquier navegador, host, sistema operativo, además de ser empleado para aplicaciones web, de escritorio y aplicaciones móviles. Siempre que se compila el código generado en Typescript

este se convierte en Javascript que es interpretado por los navegadores y de manera legible Angular.IO (2019).

Los componentes necesarios para compilar typescript es Node.js y Git, ambos elementos necesarios en los ambientes de desarrollo, no se entra en detalle en estos dos elementos, pero basta con indicar que brindan un potencial enorme al desarrollar aplicaciones web.

Entre los elementos que conforman Angular tenemos lo que son los módulos, componentes, archivos de estilos, archivos de enrutamiento de páginas, servicios, directivas y pipes, cada uno cumple distintas funcionalidades. Una aplicación Angular dispone de un módulo principal (similar al main) que se indica en la página *index.html* del sitio (Angular.IO, 2019).

Si se desea estudiar más sobre Angular invito al lector a realizar el "Tour of Heroes" que ayuda a inicializarse en el desarrollo de aplicaciones web con Angular https://angular.io/tutorial.

### 2.7.2. D3.js

D3.js es una biblioteca Javascript, se integra fácilmente con Angular o con aplicaciones Javascript directamente, la ventaja de D3 es que manipula el DOM de las páginas web por lo que no se ata a ningún software en específico convirtiéndose en una herramienta libre por lo que puede ejecutarse en distintos navegadores. D3 se basa en los datos y en como estos se desean presentar en HTML, SVG o CSS (Bostock, 2019).

El creador de D3, Mike Bostock (2019) no da un ejemplo claro de lo que se puede realizar con D3 como es el caso de la creación de una tabla HTML a partir de un conjunto de datos, esto indica una vinculación entre los datos y un objeto del DOM. A continuación una comparación en donde se visualiza la reducción de código con D3 en javascript.

```
1 var paragraphs = document.getElementsByTagName("p");
2 for (var i = 0; i < paragraphs.length; i++) {
3   var paragraph = paragraphs.item(i);
4   paragraph.style.setProperty("color", "blue", null);
5 }
```

Fragmento de código 1: Selección elemento DOM con Javascript

```
d3.selectAll("p").style("color", "blue");
```

Fragmento de código 2: Selección elemento DOM con D3.js

Como desventaja a D3, a medida que se desarrolla, el código empieza a crecer y el mantenimiento por parte del desarrollador se dificulta, además que crear una simple acción de tooltip requiere de mucho código, por esa razón existen paquetes que hacen uso de D3 y facilitan la manipulación de elementos que en este proyecto son visualizaciones o gráficas, una de estas y que se integra con Angular es el paquete ***Ngx Charts*** y se usa para la presentación de gráficas y permite una fácil configuración de las mismas (https://swimlane.github.io/ngx-charts/).

## 2.8. Herramientas de base de datos

Esta sección esta dedicada a detallar de manera breve la base de datos que fue empleada para registrar la configuración del Sistema de Información Gerencial, así como también la base de datos en donde se almacenará los datos del repositorio principal, es decir la información que se extraerá desde los orígenes.

### 2.8.1. MongoDB

MongoDB es una base de datos no relacional basada en documentos, que permite el crecimiento de la base de manera horizontal o vertical, es decir se puede expandir el espacio de datos de MongoDB agregando nuevos nodos o incrementando la capacidad de almacenamiento del servidor.

MongoDB dispone de dos versiones una que es empresarial y una versión comunitaria, para el TFM se usa software libre es decir la versión comunitaria. Vale destacar que una persona cuando inicia con MongoDB es común que se encuentre perdido debido a las grandes diferencias que tiene con una base relacional (MongoDB, 2019).

Como se menciona al inicio de la sección, MongoDB esta basado en documentos en este caso JSON, pero esto no implica que la información que se almacena a de estar de la misma manera en la base de datos, ya que JSON esta limitado en ciertas funcionalidades como es el caso de adherir mas formatos de tipo, MongoDB trabaja con una representación binaria

conocida como BSON (MongoDB, 2019).

Un ejemplo que podemos encontrar de un archivo BSON tenemos: (MongoDB, 2019):

```
{
    name: "sue",
    age: 26,
    status: "A",
    groups: ["news", "sports"]
}
```

Fragmento de código 3: Documento BSON

Las ventajas de utilizar documentos son (MongoDB, 2019):

- Los documentos (es decir, los objetos) corresponden a tipos de datos nativos en muchos lenguajes de programación.
- Los documentos incrustados y las matrices reducen la necesidad de uniones costosas.
- El esquema dinámico soporta el polimorfismo fluido.

## 2.9. ¿Cómo reconocer si se está trabajando en un entorno de big data?

De hecho, fue una de las primeras preguntas que me surgieron al momento de plantearme este proyecto de fin de máster, puesto que no sabía si la cantidad de información que manipularía fuera lo suficientemente grande, de hecho revisando los datos vemos que se tiene por cada institución financiera aproximadamente 8000 registros, y si sumamos las instituciones financieras tenemos un total de 22 bancos y mas de 130 cooperativas, el total de registros supera aproximadamente el millón, si esta data la obtenemos mensualmente, entonces el almacenamiento que se requiere anualmente será de 15 millones de registros aproximadamente.

Revisando los datos que se extraen se ve la necesidad de poder tener un almacén de datos que ofrezca las bondades que brinda MongoDB, y para su procesamiento Apache Spark, entonces en parte me llevo a plantear esta solución como un proyecto de big data.

De esto podemos indicar que se parte de la idea de lo que se desea realizar y a que punto inicialmente se desea llegar, manteniendo siempre una visión futura de que más se

podría realizar, pero para ello debemos establecer un marco de trabajo para que a partir de aquí se pueda seguir avanzando con nuevos proyectos.

En base a mi experiencia profesional, las empresas mantienen los datos almacenados en sus bases únicamente para reportes, realizando los procesos típicos de administración, como respaldos y posterior encerado de las mismas para permitir almacenar más datos; en caso de que se requiera reportes de determinadas fechas, se procedía a la recuperación de estos y a la generación de los reportes necesarios, como vemos esto requiere de mucho tiempo de trabajo, por eso "procesamiento big data" esta pensado para esto precisamente en almacenar gran cantidad de información y poder analizarla cuando se necesite.

Entonces, como saber si se está trabajando con big data, pues dependerá de la cantidad de información que manejaremos y del valor que se desee sacar a los datos, ya sea aplicando conocimientos orientados a estadística, inteligencia artificial o modelado de los datos, etc. Resulta de cualquier proceso que al momento de realizarlos una máquina sencilla o un servidor no lo pueda hacer por la cantidad de información que se esta manejando, en ese caso se puede plantear la idea de manejar tecnologías big data, que nos permitan realizar análisis de los datos y obtener información valiosa. Un caso de estudio por ejemplo es el que se plantea al inicio de esta sección, ahora imaginémonos esto en un entorno empresarial, la necesidad de manejar herramientas big data para que la empresa pueda tomar decisiones en base a los datos que generan yo lo veo como algo necesario, y esa necesidad es la que uno debe plantearse al momento de determinar si se trabaja o no con big data.

Del párrafo anterior, trato de poner un caso de ejemplo en donde se parte de la necesidad, es decir obtener promedios de los balances contables, y revisar si los datos que se disponen son suficientes y si los tenemos o en su defecto hay que extraerlos, pues algo similar deberían de plantearse las empresas al momento de saber si esta trabajando en un entorno big data. Si vemos que actualmente, nuestros procesos no dan con los resultados esperados, pues se necesitan de mas datos para los análisis y mas datos mayor cantidad de recursos, pues hoy en día existen tecnologías que permiten a las áreas de TI a plantearse soluciones que no necesariamente son costosas, algunas son de software libre como lo revisamos en las secciones anteriores y con las cuales se trabaja en este proyecto de TFM, sino que dependiendo de la capacidad de las empresas podrán ahorrarse en determinados costos ya sea implementando una solución propia o adquiriendo una existente, actualmente existen algunas empresas como por ejemplo Hortonworks, Cloudera, Databricks, etc. que brindan soporte, y capacitaciones para el uso de sus herramientas.

En el siguiente capítulo, se plantean los objetivos concretos y la metodología que se usó para elaborar el proyecto de TFM, además de un análisis de cuáles de las herramientas que se trataron en este capítulo 2 fueron con las que se trabajaron y el porque.

# 3. Objetivos concretos y metodología de trabajo

## 3.1. Objetivo general

Desarrollar un sistema de información sobre el mercado financiero ecuatoriano haciendo uso de tecnologías de captura, almacenamiento y procesamiento de datos masivos (big data) que permitan disponer de información integrada, oportuna y confiable sobre las captaciones y colocaciones de las instituciones financieras del Ecuador y de esta forma presentarlas adecuadamente a través de una aplicación web consiguiendo que una institución financiera pueda tomar decisiones a nivel gerencial.

## 3.2. Objetivos específicos

- Diseñar un repositorio centralizado de datos que permita el almacenamiento de datos masivos y ofrezca un crecimiento horizontal.
- Implementar un repositorio centralizado para que almacene los datos de las instituciones financieras obtenidas de las instituciones de control.
- Implementar un robot automático que consulte la existencia de datos en los orígenes y de existir, proceder a descargárselos para su posterior almacenamiento en el repositorio central.
- Diseñar una base de datos que permita la administración y gestión del sistema de información gerencial.
- Implementar una base de datos que permita registrar usuarios, permisos, instituciones financieras, configuraciones de procesos batch y módulos que conforman el sistema de información gerencial.
- Validar que el software desarrollado cumpla las necesidades de los usuarios en términos de usabilidad y accesibilidad.

- Construir procesos que realizan el tratamiento de los datos para establecer una hegemonía en los datos a partir de los descargados.
- Hacer usos de tecnologías de software libre para la implementación del sistema de información gerencial.
- Evaluar el sistema actual y compararlo con el nuevo para verificar si se cumple con el objetivo general.
- Evaluar las herramientas de big data disponibles y determinar cuales se ajustan a las necesidades actuales del proyecto.
- Permitir que a futuro el repositorio pueda seguir creciendo con más datos para que se puedan aplicar posteriores proyectos en base a la información obtenida.

# 4. Metodología

Es importante definir una metodología para poder establecer un marco de trabajo que permita la continuidad del software con nuevos procesos de análisis sobre los datos, inicialmente el aprovechamiento de estos es por medio de un sistema de información gerencial los cuales son desarrollados con la finalidad de proporcionar información acerca del estado actual de un tema de estudio, y además, permitir la comparaciones en el tiempo para que en base a lo revisado, ejecutar acciones que brinden un cambio para bien.

Como vemos se plantea el desarrollo de una aplicación y como tal se lo debe considerar como proyecto de software, en donde se junten metodologías de ingeniería de software, así como el proceso de gestión de proyectos.

Al proyecto de software se lo debe ver como un conjunto de fases que se ejecutan en secuencia, es decir que una fase depende de la anterior, estas fases guiarán en la metodología de trabajo, es decir una guía de paso a paso para conseguir el cumplimiento de los objetivos, por esa razón se definen 4 fases iniciales:

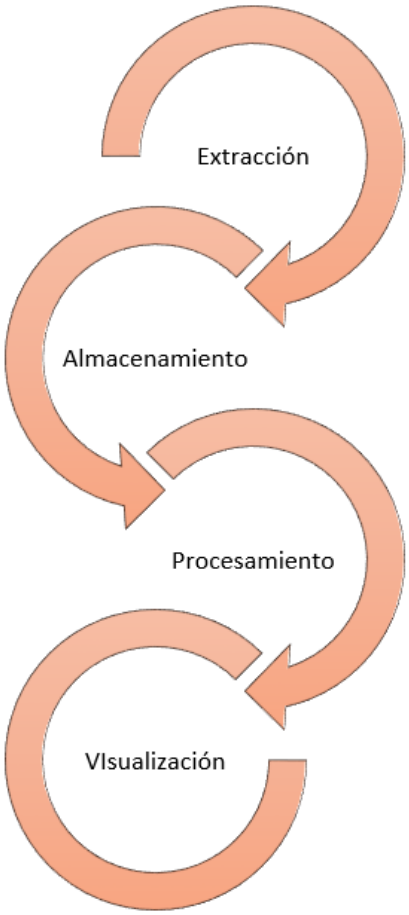



Figure 20: Fases del proceso de desarrollo
*Fuente.* Elaboración propia

Cada una de estas fases sigue un proceso de desarrollo individual, ya que se centran en una definición propia, es decir, para cada fase se tiene:

- **Extracción:** esta fase se basa en la extracción de la información, para este proyecto, son los balances generales. Se debe analizar, diseñar, implementar y probar procesos que se encarguen de automatizar la captura de los datos. Estos procesos son construidos de forma independiente, con la intención de que un equipo de trabajo pueda añadir nuevos para posibles extracciones futuras.

- **Almacenamiento:** en esta fase se diseño un modelo de almacenamiento centralizado que facilito generar consultas que el sistema necesitará. El diseño esta orientando a la elaboración de un datalake de información, el cuál se convirtió en el repositorio de datos.

  También en base a los requerimientos de la aplicación, se diseño un modelo de base de datos que resulta en el cimiento del sistema.

  Es importante diferenciar el datalake, de la base de datos del sistema, en el primer caso

este únicamente guarda los datos que se extraen de las fuentes, el segundo corresponde a las parametrizaciones necesarias para el Sistema de Información Gerencial.

- **Procesamiento:** esta fase va mas o menos ligada con el proceso de extracción, la diferencia está en que aquí se trabajó con datos ya almacenados en el datalake y se definió la arquitectura de los procesos para permitir el crecimiento de nuevos.

  Es esta fase, también existe una metodología de trabajo ya que se ejecutó las etapas de análisis, diseño, implementación y prueba de los servicios que leen los datos procesados, estos datos corresponden al repositorio y a la base de parametrizaciones del sistema.

- **Visualización:** esta fase consistió en el análisis, diseño, implementación y pruebas de la aplicación web, esta aplicación es independiente del back-end. De esta forma, se consiguió que se puede elaborar otros proyectos sin necesidad de tener que cruzar por el sistema de información gerencial.

Como revisamos cada fase a diferencia del almacenamiento, dispone de las etapas que se tienen en la metodología de desarrollo de software, por lo cuál se adaptó el Proceso Unificado de Desarrollo de Software (Unified Software Development Process, USDP), con esto trato de cumplir con las actividades propias de cada fase del sistema de información gerencial.

En forma resumida, para este proyecto y planteando un escenario de trabajo futuro, el proceso de desarrollo de software parte de la construcción/modificación de un módulo del Sistema de Información Gerencial, este módulo puede requerir nuevos datos en la base del sistema y/o nuevos datos en el repositorio, en cualquiera de los dos casos se parte de la metodología mencionada, para este proyecto de TFM, se plantea seguir la siguiente estructura:

| Fases del Ciclo de Vida | Actividades | Artefactos |
|---|---|---|
| **Inicio** | - Planteamiento del problema<br>- Definir la fase de aplicación<br>- Definición del alcance del problema | - Historias de usuario<br>- Especificación de los requerimientos del sistema |
| **Elaboración** | - Definir cronograma de desarrollo<br>- Identificar los orígenes de los datos<br>- Solicitar permisos para el acceso a datos<br>- Modelo de base de datos<br>- Modelo de diseño<br>- Definir la arquitectura del sistema | - Cronograma de trabajo<br>- Documento de origen de datos (Host, Formato, Campos, Dirección IP, etc)<br>- Documento de arquitectura del sistema<br>- Diseño de la base de datos<br>- Diseño inicial del sistema |
| **Construcción** | - Modelo de despliegue<br>- Construcción de procesos de extracción de datos<br>- Modelo de clases<br>- Modelo de actividades<br>- Modelo de secuencia<br>- Construcción de servicios de consulta de los datos | - Desarrollo de la base de datos<br>- Desarrollo del proceso/módulo<br>- Plan de pruebas |
| **Transición** | - Puesta en producción del sistema de información<br>- Análisis de la información capturada<br>- Creación de acta de cierre de proyecto | - Sistema Funcionando<br>- Acta de cierre de proyecto |

Table 1: Fases del ciclo de desarrollo
*Fuente.* Elaboración propia

En el proceso de trabajo del proceso unificado si bien puede estar estructurado en distintas fases como vemos en la imagen 21, para el proyecto nos enfocaremos en las de análisis, diseño, codificación, pruebas y evaluación.

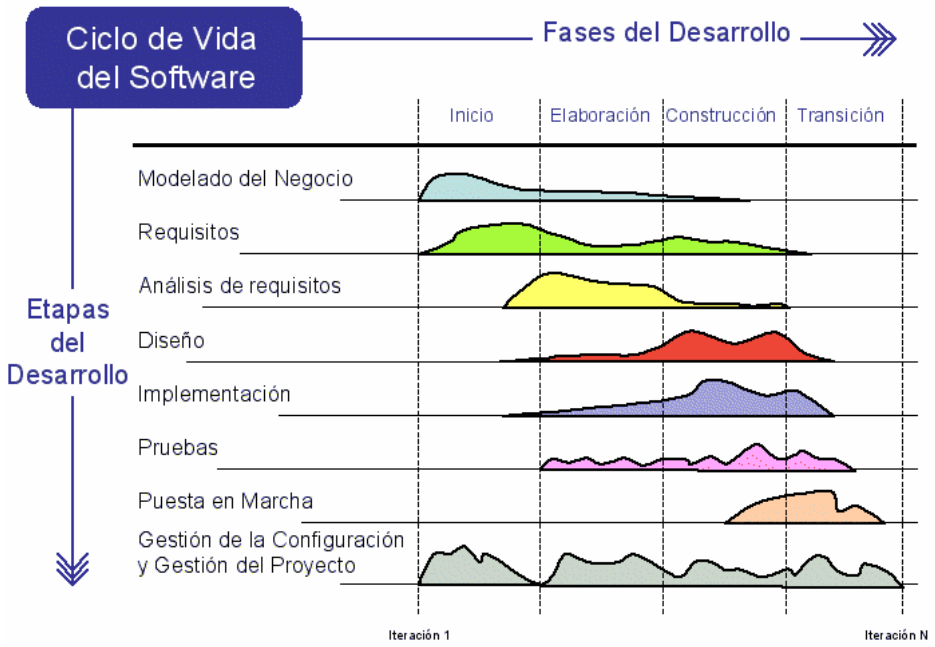



Figure 21: Fases del ciclo de desarrollo
*Fuente.* https://i13.servimg.com/u/f13/11/74/78/81/desarr10.png

## 4.1. Análisis de los requisitos

En esta sección se plantearon los requerimientos funcionales, no funcionales, el comportamiento del sistema y las propiedades. Para este proceso se mantuvo una reunión con la gerente de la empresa "Logical Consulting" y el área de sistemas que vinieron a ser los skateholders. Para levantar esta información se mantuvo reuniones con ellos y se pudo definir las necesidades del proyecto.

De las actividades que se lograron identificar planteamos las siguientes:

### 4.1.1. Usuarios

Para el sistema de información gerencial se plantea los siguientes usuarios:

1. **Usuario Administrador:** acceden a toda la información disponible a partir de una interfaz de administración Web en donde tendrá acceso a toda las configuraciones del sistema. Incluido el control de los procesos batch.
2. **Usuario Institución:** acceden a información de la institución a la cuál esta atada, únicamente podrá visualizar los módulos a los cuales tenga acceso y podrá analizar los

datos que le correspondan según a la categoría que pertenezca. La mayoría de usuarios entran en este perfil.

3. **Usuario Invitado:** son aquellos que acceden a una versión libre y solo pueden analizar un número especifico de instituciones (parametrizable) y no tienen acceso a todos los módulos (parametrizable), en caso de que se el usuario desee registrarse lo puede hacer, esto se notifica inmediatamente al usuario administrador que define el perfil al cuál pertenecerá.

## 4.2. Planteamiento de requerimientos funcionales, no funcionales y características del sistema

En esta etapa se definen los procesos básicos que debe realizar el sistema, así como los requerimientos funcionales, disponibilidad, rendimiento y seguridad.

### 4.2.1. Recolección de los datos existentes

Como se trato en el capítulo del estado del arte 2, en la sección de fuentes de información, inicialmente se trabajan con dos que se extraen de las páginas web de las entidades de control del Ecuador como son SB y la SEPS, en el primer caso los datos se encuentran alojados dentro de una página web, la cuál resulta de una sección table de un HTML, en el siguiente caso, los datos se subdividen por segmento y se los extrae a partir de un documento de excel, a este documento se lo debe analizar para extraer los datos de interés. En esta etapa, se definen que datos de los que se extrajo sirve para presentar la información en el sistema.

## 4.3. Diseño del sistema

Para el proyecto de desarrollo del Sistema de Información Gerencial se debe realizar un diseño correctamente estructurado que represente los modelos físicos y lógicos, así como los componentes que conforman el sistema, así como los recursos a ser empleados.

En esta etapa se define el diseño de la base de datos, en este caso el repositorio central y la base de datos del sistema, definir los formularios del sistema web de administración y el diseño del dashboard del sistema de información (gráficas, charts, kpi). También se debe

diseñar la arquitectura del software que como al inicio de la metodología este sera, una para el repositorio centralizado, para el servidor de aplicaciones y el servidor de microservicios.

### 4.3.1. Modelado del software y base de datos

Para realizar el modelado siguiente el proceso unificado si uso los diagramas de secuencia, procesos, bases de datos, despliegue y componentes; con esto se logra identificar las relaciones que existen entre los distintos componentes del sistema.

### 4.3.2. Diseño de interfaz de usuario

En base a reuniones mantenidas con los skateholders se establecieron unos prototipos de pantallas que serán construidas en el sistema de información gerencial, en base a estas interfaces se pretende dar un dirección en la navegación del sistema además que sea simple, sencilla y amigable para los usuarios.

## 4.4. Generación del código

Como el nombre de la etapa lo indica, aquí el modelado de la aplicación se convierte a un sistema software que es interpretado por la computadora, es decir la generación del código fuente. Se eligen las herramientas de software libre que ofrezcan el procesamiento de datos masivos y tecnologías que ofrezcan un desarrollo a nivel de interfaz de usuario, además incluir un acceso a los datos por medio de tecnologías de microservicios.

Las herramientas seleccionadas van atadas a la fase:

- **Extracción:** las herramientas seleccionadas para esta etapa tenemos al Lenguaje R, que permitió construir una aplicación de tipo web scrapping para obtener los balances generales de los bancos, también fue empleado para la descarga y lectura de los archivos de excel de las cooperativas y mutualistas.
- **Almacenamiento:** las herramientas empleadas en este proceso de almacenamiento para los datos extraídos fue MongoDB. Este gestor de datos no relacional viene a ser el repositorio centralizado, pero también es usado como una base de datos intermedia que almacena los datos obtenidos de la web, que luego mediante la tecnología de Apache

Spark son almacenados en la base del repositorio, también se uso a MongoDB como base de datos del sistema de información gerencial, la decisión de usar esta base en lugar de una relacional es que es software libre y como el proyecto al no tratarse de un sistema transaccional sino que es un sistema para la toma de decisiones, por lo general se tienen mas consultas, en este caso MongoDB al ser no relacional las gestionará de forma más ágil y rápida.

- **Procesamiento:** el motor de procesamiento a usar es Apache Spark este se puede implementar en modo clúster, por la cantidad de servidores que se disponga. Se construyó procesos Spark que tomaron datos de una fuente en este caso MongoDB (se pueden tener otras), realiza un procesamiento y los almacena en el mismo MongoDB. Estos procesos se construyeron en Python.
- **Visualización:** la tecnología empleada para la administración del sistema es Angular y para el sistema de información son tecnologías javascript como es dc.js y d3.js

## 4.5. Implementación

Esta fase define la instalación y configuración de los servidores a utilizar para la implementación, como servidor de aplicaciones se uso HTTPD, como API GATEWAY para que administre las API's se uso Gravitee.IO y para ambientes de desarrollo y prueba se trabajo en modo standalone de Apache Spark y con un servidor MongoDB, pero en ambientes productivos se necesita como mínimo un factor de réplica 3 para MongoDB y sobre los mismos 3 servidores se configuró el clúster de Apache Spark. Para la ejecución de procesos automáticos de Apache Spark se trabajo con Apache Oozie, el cuál es un gestor de procesos hadoop/spark/sqoop/java etc. Este permite definir un flujo de trabajo automático parametrizable.

## 4.6. Pruebas

En esta etapa de pruebas se verifica que la aplicación funcione de acuerdo a lo esperado y revisando que cumpla con los objetivos, el diseño y análisis planteado. Para esta etapa se define una plantilla de pruebas que se irá marcando como pasó o falló determinada función y en caso de existir errores deben ser corregidos antes de la puesta en producción del sistema.

# 5. Desarrollo específico de la contribución

En esta sección se detalla el desarrollo de la aplicación guiado por las etapas por las que atraviesa el proyecto de Sistema de Información Gerencial (Extracción, Almacenamiento, Procesamiento, Visualización), se describe cada uno de los procesos de la ingeniería del software que tuvieron que analizarse, diseñarse, desarrollarse e implementarse, partiendo desde la identificación de los requerimientos ya sean estos funcionales y no funcionales, el diseño del modelo del proceso de negocio, bases de datos y finalmente en las secciones realizar el estudio de usabilidad y la validación del cumplimiento de los distintos casos de uso.

## 5.1. Inicio

La recolección de los requisitos fue elaborado en conjunto con la empresa mencionada Logical Consulting, por lo que la validez del mismo parte de la solución final que se desea obtener, estos son la base de lo que se desea diseñar y su posterior implementación.

La información que se obtuvo de las reuniones (Anexo II. Recolección de requisitos. 8) fue la siguiente:

- Definición de la necesidad de actualizar Sistema de Información Gerencial Actual.
- Descripción de las tareas a realizarse para satisfacer la necesidad planteada.
- Descripción del sistema actual.
- Descripción de la información a recolectar.
- Definición de los acuerdos legales entre la empresa y el gestor de la solución.
- Detalles generales.

## 5.2. Elaboración

Consiste en el levantamiento del análisis de la solución al problema y como se pretende resolverlo en base a la aplicación informática que se desea construir, en esta fase se levantan los requerimientos de cada uno de los usuarios que intervienen y la interacción de estas funcionalidades con cada actor.

### 5.2.1. Identificación de los requisitos

El levantamiento de los requerimientos parten del usuario final, en este caso la administradora de la empresa Logical Consulting, que además es la administradora del sistema. Se realizaron reuniones en donde se definen cada uno de los actores que intervienen en la solución, en base a esto se definió los siguientes actores y el rol que cumplen en sus actividades cotidianas.

| Tipo | Actor | Descripción |
|---|---|---|
| **Etapa: Extracción** | | |
| INTERNOS | A001.<br>Administrador del Sistema | Descripción:<br>• Este actor tiene la capacidad para acceder a la base de datos y cambiar la parametrización de los procesos automáticos como la frecuencia, dirección de extracción de los datos, también puede analizar los resultados de los procesos automáticos, en caso de fallo tiene la facultad de ejecutar el proceso de extracción de forma manual.<br><br>Roles:<br>• Acceso a la base de datos de los procesos automáticos.<br>• Acceso al servidor donde se ejecuta los procesos automáticos.<br>• Administración de los orígenes.<br>o Frecuencia ejecución<br>o Reprocesar extracción de información manualmente. |
| | A002.<br>Robot Automático. | Descripción:<br>• Se considera como actor porque se encarga de ejecutar diariamente la extracción de los datos.<br>• El administrador puede agregar más orígenes y parametrizarlos en la base de datos de los procesos batch. |
| **Etapa: Almacenamiento** | | |

| | | |
|---|---|---|
| INTERNOS | A003.<br>Administrador de Base de datos | Descripción:<br>• Encargado del mantenimiento y configuración de las bases de datos.<br>• Generación de reportes directos de la base de datos por petición del administrador.<br><br>Roles:<br>• Administración de las colecciones de datos.<br>• Administración de los datos (inserción, actualización, eliminación).<br>• Administración de las bases de datos.<br>• Administración de los índices.<br>• Consultas de datos. |
| **Etapa: Procesamiento** | | |
| INTERNOS | A004.<br>Administrador de clúster de procesamiento | Descripción:<br>• Encargado del monitoreo, configuración y mantenimiento de los nodos del clúster de procesamiento.<br><br>Roles:<br>• Acceso a los servidores.<br>• Acceso a la configuración del clúster.<br>• Monitoreo de las aplicaciones que se ejecutan sobre el clúster. |
| **Etapa: Visualización** | | |

| | | |
|---|---|---|
| INTERNOS | A001.<br>Administrador del Sistema | Descripción:<br>• Tiene acceso a todas las opciones del sistema, a la revisión de los datos de todas las instituciones financieras, mantenimiento de usuarios y asignación de permisos.<br><br>Roles:<br>• Cargar dashboard de todas las instituciones.<br>• Filtrar datos de consulta en dashboard.<br>• Comparaciones datos entre instituciones.<br>• Visualización variaciones datos de instituciones.<br>• Administración de usuarios. |
| EXTERNOS | A005.<br>Usuario Institución<br>(Usuario registrado) | Descripción:<br>• Usuario registrado y aprobado por la administradora del sistema que accede a la aplicación y consulta la información de su institución, puede visualizar datos de distintas fechas y comparar su estado actual con otra institución.<br><br>Roles:<br>• Acceso al dashboard principal.<br>• Filtrar datos de consulta en el dashboard.<br>• Comparaciones de datos entre instituciones.<br>• Visualización de variaciones de datos de instituciones. |
| | A006.<br>Usuario Invitado | Descripción:<br>• Usuario no registrado en la aplicación web, pero que puede acceder para consultar la información actual de las instituciones financieras, pero no tiene acceso a todos los módulos del sistema.<br><br>Roles:<br>• Acceso al dashboard principal. |

Table 2: Actores.

*Fuente.* Elaboración propia

En la tabla siguiente se visualizan los requerimientos funcionales y no funcionales:

| Codigo | Nombre Requerimiento | Descripción |
|---|---|---|
| RF001 | Extracción balances generales | Se refiere al proceso de extracción de los datos de forma automática |
| RF002 | Administración procesos automáticos | Se asocia a la gestión de los procesos batch, en cuanto a la frecuencia de extracción de los datos y la parametrización de los orígenes |
| RF003 | Administración clúster de procesamiento | Se refiere a la gestión del clúster de procesamiento. |
| RF004 | Sistema basado en roles | Se refiere a los permisos que debe tener un usuario, se ata el usuario a lo que puede visualizar en el dashboard. |
| RF005 | Visualización de dashboard | Examinar los datos en un dashboard principal que muestra gráficas de estructura, tendencia, tablas de variación. |
| RF006 | Consulta de captaciones | Consiste en información de los saldos captados por cada institución. |
| RF007 | Consulta de colocaciones | Consiste en información de saldos de créditos colocados por cada institución. |
| RF008 | Consulta de variaciones | Consiste en una comparación de los datos de la fecha actual con respecto al mismo mes pero del año anterior. |
| RF009 | Comparación entre instituciones | Permitir al usuario comparar la institución a la que pertenece con otras del mismo nivel. |
| RF010 | Registro de usuarios | Permitir al usuario administrador el registro de los usuarios y asignación de permisos. |
| RF011 | Filtro cruzado | Permitir una interacción directa en los gráficos y combinarlo con otros gráficos. |
| RF012 | Administración base de datos | Monitoreo de los recursos de la base de datos, gestión de las bases de datos, generación de reportes a partir de las bases de datos. |

Table 3: Requerimientos Funcionales.

*Fuente.* Elaboración propia

| Codigo | Nombre Requerimiento | Descripción |
|---|---|---|
| RNF001 | Task scheduler SO | Se refiere a la gestión de tareas de ejecución por parte del administrador del sistema. |
| RNF002 | Rendimiento Base de datos | Se refiere a la capacidad de mantener los datos consistentes con configuraciones de réplica y sharding. |
| RNF003 | Seguridad Base de datos | Se refiere a la gestión de los usuarios que tienen acceso a la base de datos, y a los permisos y roles que tendrán los distintos usuarios. |
| RNF004 | Hardware Base de datos | La base de datos permite crecer horizontalmente, se simula la ejecución de dos servidores físicos mediante máquinas virtuales. Este servidor debe contar con:<br>- 8GB de memoria RAM mínimo.<br>- Mínimo 500GB de disco duro.<br>- Procesador 4 núcleos. |
| RNF005 | Software Base de datos | Para este caso la base de datos es MongoDB, y se puede ejecutar en:<br>- SO: Windows Server / Centos 7<br>- MongoDB Community Edition |
| RNF006 | Rendimiento clúster procesamiento | Se configurará el cluster en mismo número de servidores que se tienen para la base de datos, es decir 2 servidores, con una configuración máster y esclavo. |
| RNF007 | Hardware clúster procesamiento | Mismos usados para la base de datos. |
| RNF008 | Software clúster procesamiento | El software para usado para la configuración es Apache Spark, para esto se trabaja con el SO. Linux Centos 7 |
| RNF009 | Seguridad Sistema | Se basa por medio de credenciales básicas como es el nombre de usuario y contraseña, dependiendo de los permisos y roles podrá acceder a los distintos módulos. |
| RNF010 | Servidor de aplicaciones | El servidor de aplicaciones a usar para el Sistema es Apache Httpd |

| | | |
|---|---|---|
| RNF011 | Diseño de dashboard | El diseño del dashboard esta dado por un resumen de totales, gráficas, todos estos aplicando técnicas de visualización de datos. |
| RNF012 | Creación de micro servicios | Se planea trabajar con SpringBoot para la creación de API's para el acceso a los datos, publicarlos y luego ser llamados desde la aplicación web. |

Table 4: Requerimientos No Funcionales.

*Fuente.* Elaboración propia

### 5.2.2. Vistas de casos de uso

En la siguiente tabla se definen cada uno de los casos de uso que he planteado para el desarrollo del sistema, recordar que algunos casos de uso involucran las 4 fases descritas al inicio de la sección.

| Caso de uso | Nombre del caso de uso | RF001 | RF002 | RF003 | RF004 | RF005 | RF006 | RF007 | RF008 | RF009 | RF010 | RF011 | RF012 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CU001 | Consultar disponibilidad datos | X | | | | | | | | | | | |
| CU002 | Reprocesar extracción datos | X | | | | | | | | | | | |
| CU003 | Almacenar datos en el repositorio | X | | | | | | | | | | | |
| CU004 | Gestiona frecuencia de extracción de datos | | X | | | | | | | | | | |
| CU005 | Gestionar los reintentos de extracción de los datos | | X | | | | | | | | | | |
| CU006 | Gestionar los componentes del clúster | | | X | | | | | | | | | |

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CU007 | Gestionar permisos de usuarios registrados | | | | X | | | | | | | | |
| CU008 | Crear usuario institución | | | | X | | | | | | X | | |
| CU009 | Habilitar acceso al sistema | | | | | | | | | | X | | |
| CU010 | Visualizar dashboard de información | | | | | X | X | X | X | X | | X | |
| CU011 | Visualizar datos de captaciones | | | | | | | X | | | | | |
| CU012 | Visualizar datos de colocaciones | | | | | | | | X | | | | |
| CU013 | Visualizar variaciones en los datos | | | | | | | | | X | | | |
| CU014 | Filtrar datos de consulta | | | | | | | | | | | X | |
| CU016 | Autenticar usuario | | | | X | X | X | X | X | X | X | | X |
| CU017 | Generar reportes desde la base de datos | | | | | | | | | | | | X |
| CU018 | Gestionar los accesos y consultas de las bases de datos | | | | | | | | | | | | X |

Table 5: Tabla de casos de usos y requerimientos

*Fuente.* Elaboración propia

En la siguiente tabla se muestra la relación existente entre los actores y los casos de uso descritos en la [Tabla 3].

| Caso de Uso | Nombre del Caso de Uso |
|---|---|
| **Actor: A001. Administrador del Sistema** | |
| CU004 | Gestionar frecuencia de datos de extracción |
| CU005 | Gestionar los reintentos de extracción de los datos |
| CU007 | Gestionar permisos de usuarios registrados |
| CU008 | Creación de usuario institución |
| CU009 | Habilitar acceso a sistemas |
| CU010 | Visualizar dashboard de información |
| CU011 | Visualizar datos de captaciones |
| CU012 | Visualizar datos de colocaciones |
| CU013 | Visualizar variaciones en los datos |
| CU014 | Filtrar datos de consultas |
| CU015 | Gestionar balances de las instituciones financieras |
| CU016 | Autenticar usuario |
| **Actor: A002. Robot Automático** | |
| CU001 | Consultar disponibilidad de los datos |
| CU002 | Reprocesar extracción de los datos |
| CU003 | Almacenar datos en el repositorio |
| **Actor: A003. Administrador de Base de datos** | |
| CU015 | Gestionar balances de las instituciones financieras |
| CU017 | Generar reportes desde la base de datos |
| CU018 | Gestionar los accesos y consultas de las bases de datos |
| **Actor: A004. Administrador del clúster de procesamiento** | |
| CU006 | Gestionar los componentes del clúster |
| **Actor: A005. Usuario Registrado** | |

| | |
|---|---|
| CU010 | Visualizar dashboard de información |
| CU011 | Visualizar datos de captaciones |
| CU012 | Visualizar datos de colocaciones |
| CU013 | Visualizar variaciones en los datos |
| CU014 | Filtrar datos de consultas |
| CU016 | Autenticar usuario |
| **Actor: A006. Usuario Invitado** | |
| CU010 | Visualizar dashboard de información |
| CU010 | Visualizar dashboard de información |
| CU011 | Visualizar datos de captaciones |
| CU012 | Visualizar datos de colocaciones |

Table 6: Tabla de actores y casos de usos.

*Fuente.* Elaboración propia

### Diagrama de casos de usos

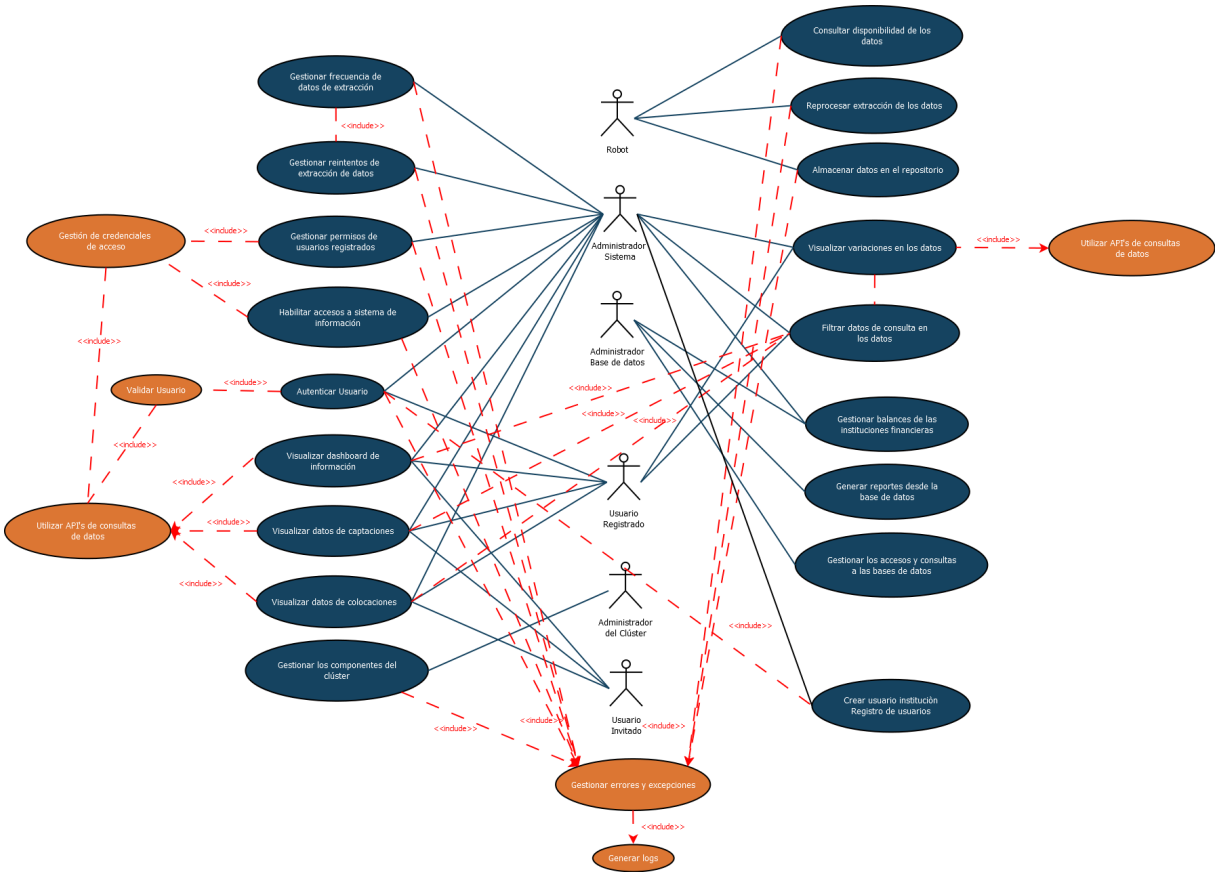



Figure 22: Diagrama de casos de uso general
*Fuente.* Elaboración propia

En la figura 22 se observa el diagrama de casos de uso en forma general de como están asociados los 18 casos de uso con respecto a los actores, pero se observa que se añaden 5 casos de uso adicional que están marcados con color naranja, estos son necesarios para dar una seguridad a los datos del repositorio y controlar los procesos automáticos, así como los casos que podrán manipular los usuarios no registrados (invitados).

### Especificación de casos de uso

A continuación se detallan los casos de uso principales del sistema como es la Visualización del dashboard de información, Visualizar datos de captaciones y Visualizar datos de colocaciones, ya que son los casos en donde los actores interactúan con mayor frecuencia.

| **Identificador:** | CU010 |
|---|---|
| **Actor (es)** | A001, A005, A006 |

| | |
|---|---|
| **Propósito** | Visualizar dashboard, en donde se detalla mediante gráficos de barras, tendencia y resúmenes el estado a la fecha actual de las captaciones, colocaciones, variaciones de las instituciones financieras. |
| **Resumen** | 1. El CU comienza con el actor, decide visualizar el estado actual del sistema financiero ecuatoriano a una fecha en concreto<br>2. El actor ingresa a la aplicación, e interactúa con el dashboard principal.<br>3. El actor puede seleccionar directamente en la gráfica los datos de las captaciones, colocaciones y sus variaciones.<br>4. El CU finaliza cuando el usuario ha consultado lo necesario y cierra o sesión o cierra la ventana del navegador. |
| **Precondiciones** | Haberse autenticado en el sistema<br>El administrador del sistema haya autorizado el ingreso<br>Que el actor este atado a una institución financiera |
| **Postcondiciones** | Visualiza los datos solicitados de su institución con respecto a las demás. |

Table 7: Descripción CU "Visualizar dashboard principal"

*Fuente.* Elaboración propia

| **Identificador:** | CU011 |
|---|---|
| **Actor (es)** | A001, A005, A006 |
| **Propósito** | Visualizar los datos de captaciones, es decir los productos de ahorros, monetarios, plazo y otros depósitos, tanto en estructura como en su variación y comparación con el resto de las instituciones |
| **Resumen** | 1. El CU comienza con el actor decide visualizar el estado de su institución financiera con respecto a las demás a una fecha en concreto<br>2. El actor ingresa a la aplicación y selecciona el módulo de captaciones para consultar sus datos financieros.<br>3. El actor puede interactuar con los gráficos y seleccionar que institución desea revisar.<br>4. El CU finaliza cuando el usuario ha consultado lo necesario y cierra sesión, cierra la ventana del navegador o regresa al dashboard principal. |
| **Precondiciones** | Haberse autenticado en el sistema.<br>El administrador del sistema haya autorizado el ingreso.<br>Que el actor este atado a una institución financiera.<br>Que tenga habilitado el módulo de captaciones. |
| **Postcondiciones** | Visualiza los datos solicitados de su institución con respecto a las demás. |

Table 8: Descripción CU "Visualizar datos de captaciones"

*Fuente.* Elaboración propia

| **Identificador:** | CU012 |
|---|---|
| **Actor (es)** | A001, A005, A006 |

| | |
|---|---|
| **Propósito** | Visualizar los datos de colocaciones, es decir los productos de consumo, comercial, microcrédito y vivienda, tanto en estructura como en su variación y comparación con el resto de las instituciones |
| **Resumen** | 1. El CU comienza con el actor decide visualizar el estado de su institución financiera con respecto a las demás a una fecha en concreto.<br>2. El actor ingresa a la aplicación y selecciona el módulo de colocaciones para consultar sus datos financieros.<br>3. El actor puede interactuar con los gráficos y seleccionar que producto desea consultar.<br>4. El CU finaliza cuando el usuario ha consultado lo necesario y cierra sesión o cierra la ventana del navegador o regresa al dashboard principal. |
| **Precondiciones** | Haberse autenticado en el sistema.<br>El administrador del sistema haya autorizado el ingreso.<br>Que el actor este atado a una institución financiera.<br>Que tenga habilitado el módulo de captaciones. |
| **Postcondiciones** | Visualiza los datos solicitados de su institución con respecto a las demás. |

Table 9: Descripción CU "Visualizar datos de colocaciones"

*Fuente.* Elaboración propia

### 5.2.3. Vista Lógica

#### Diagramas de Secuencia

En el gráfico siguiente se describe la secuencia seguida por el usuario administrador o el usuario registrado al momento de llevar a cabo la autenticación para poder acceder al sitio, es importante notar que se hace uso de un API de autenticación, que se encarga de gestionar la seguridad de los servicios, ya que para autenticarse hace uso de un token, y luego de validar se usa otro token de acceso el cual es enviado en cada consulta.

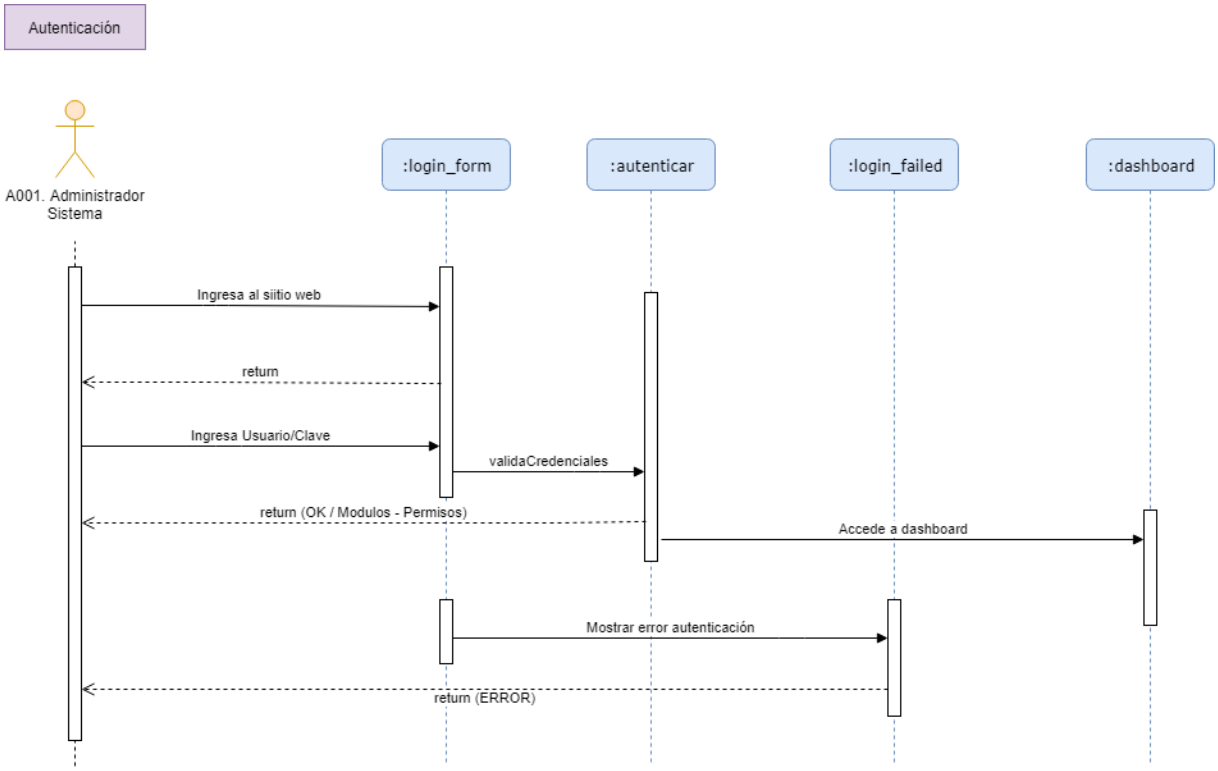



Figure 23: Diagrama de secuencia: Autenticación
*Fuente.* Elaboración propia

En el siguiente diagrama se visualiza la consulta de los datos en el dashboard de información, notese que en esta secuencia se agregan las consultas de captaciones, colocaciones y variaciones. Además, se puede visualizar que existe una secuencia de validación de token por cada consulta.

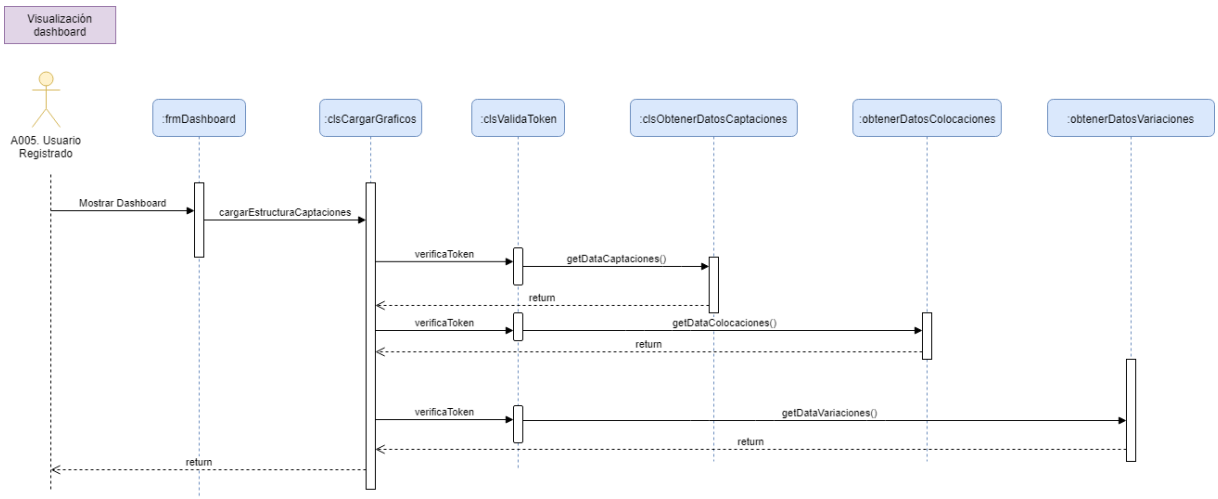



Figure 24: Diagrama de secuencia: Visualización de datos en el dashboard
*Fuente.* Elaboración propia

En los siguientes gráficos se agrega la secuencia del Actor A002. Robot, es importante detallarlo porque este robot, que en sí es un proceso automático, se encarga de extraer los datos con los que se alimenta la aplicación y la secuencia que sigue nos explica la forma de trabajar del mismo.

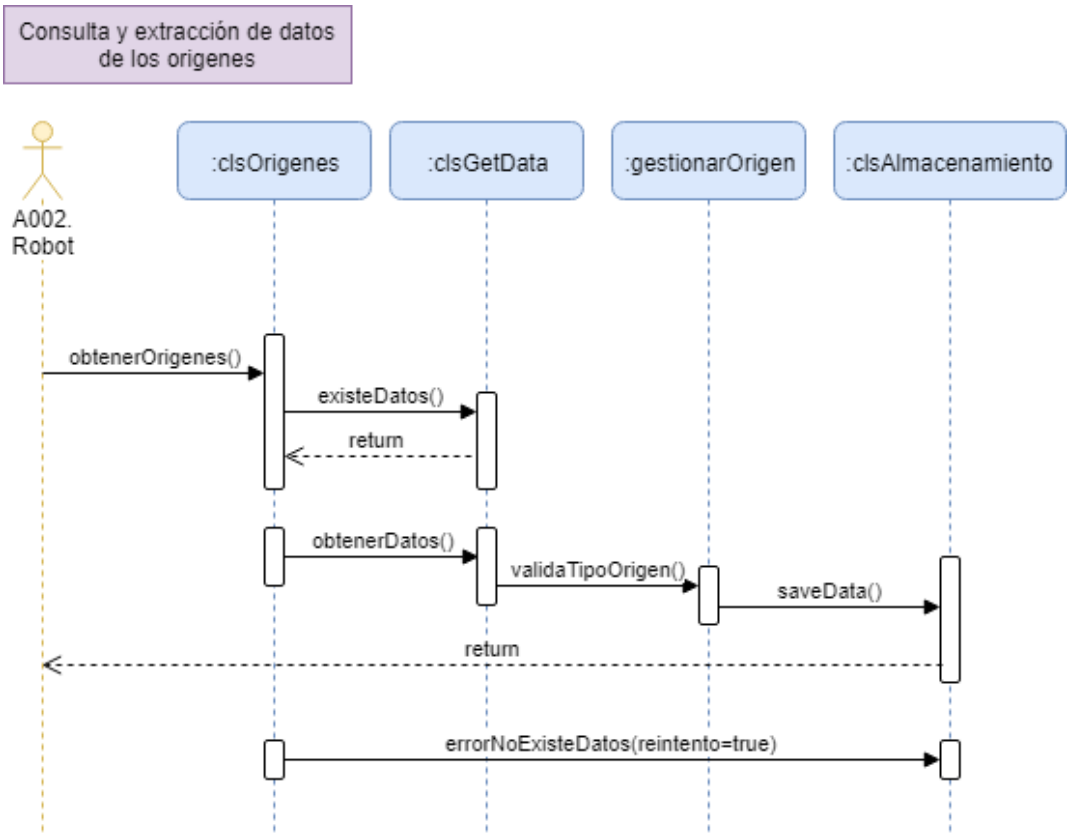



Figure 25: Diagrama de secuencia: Consulta y extracción de datos de los orígenes
*Fuente.* Elaboración propia

En el diagrama de secuencia anterior tenemos la consulta y extracción de los datos, el actor por cada origen existente de datos, verifica la existencia de los mismos, y si existen registros procede a descargarlos, gestiona el formato y la disposición de la información, y luego procede a almacenar toda la data en la base. Este proceso es repetitivo y dependerá de la frecuencia con la que se carga la información en los orígenes inicialmente son mensual.

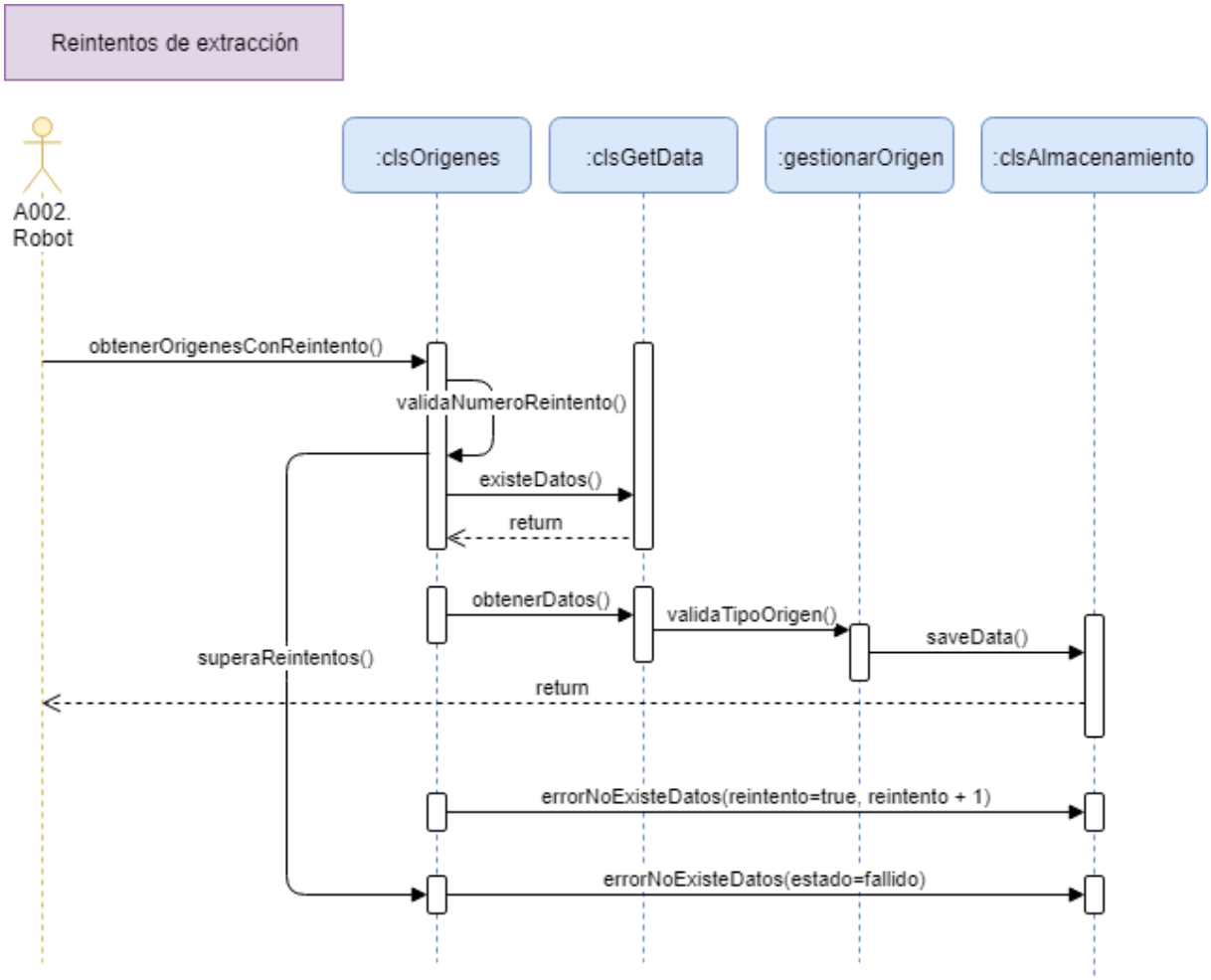



Figure 26: Diagrama de secuencia: Reintentos en la extracción de datos
*Fuente.* Elaboración propia

En el diagrama de secuencia anterior, se visualiza el proceso del robot para verificar los datos y proceder a extraerlos nuevamente, esto lo analiza diariamente, en caso de que no se obtuvo los datos y no se recopilo, el numero de reintentos aumenta, hasta llegar al total de reintentos, en este caso sale del proceso, dejando al origen como estado fallido.

**Modelo de datos**

En la figura siguiente se muestra el modelo de base de datos físico, el cuál esta conformado por una base de datos no relacional tanto para el repositorio de datos, configuraciones del proceso automático de extracción de datos y el modelo de datos del Sistema de Información Gerencial.

En el modelo de datos de la imagen siguiente se puede ver como esta conformado en un inicio el datalake o repositorio de datos, si bien esta es una base de datos no relacional,

las líneas que se observan en el diagrama no indican relación sino el flujo que siguen los datos para consolidarlos y presentar la información lista para ser consumida por el Sistema de Información Gerencial, partimos de la tabla principal que son los balances, le siguen las tablas consolidadas de captaciones y colocaciones, y a partir de estas se encuentran las variaciones.

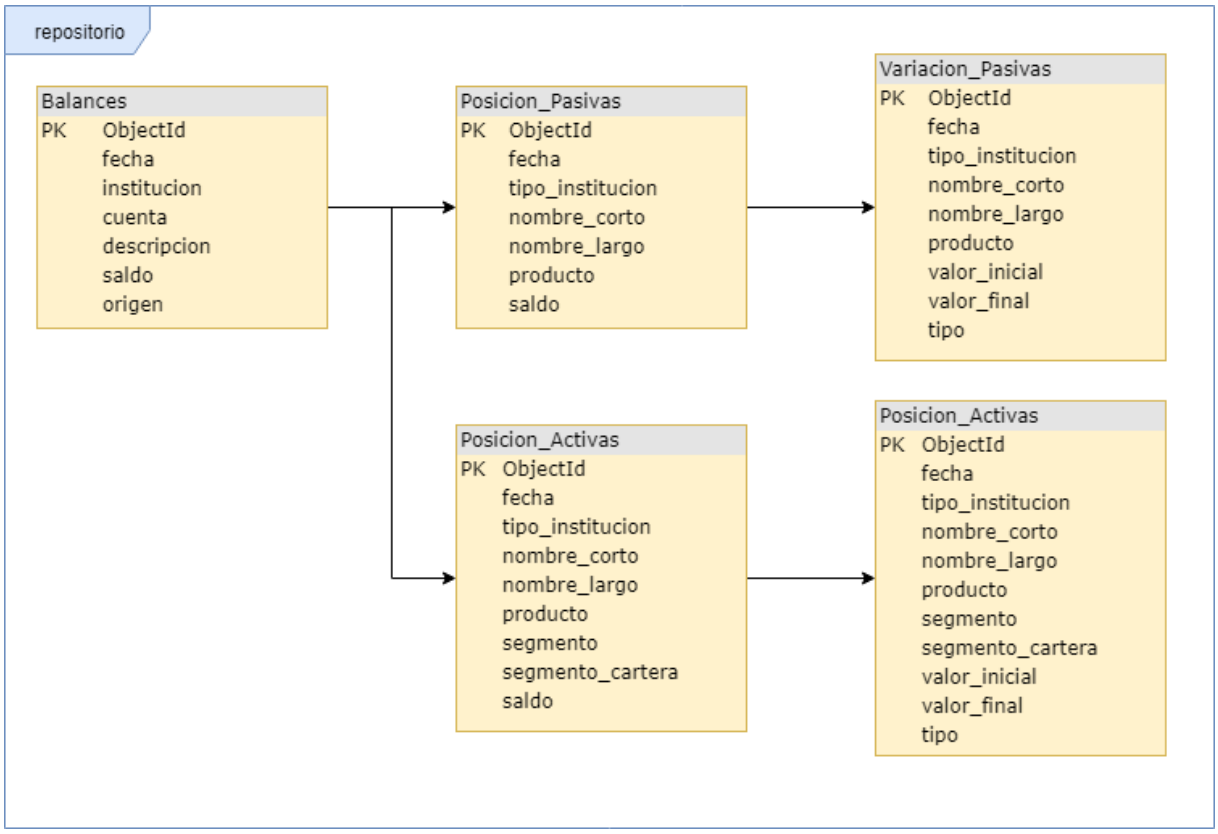



Figure 27: Modelo de datos: Datalake (Repositorio de datos)
*Fuente.* Elaboración propia

El modelo siguiente como se mencionó en un inicio viene a ser la base de datos para el control de los procesos automáticos, es decir los ejecutados por el actor A002. Robot, esta base de datos se encarga de configurar y parametrizar los distintos orígenes existentes, la idea es poder agregar nuevos orígenes que tengan su propia configuración. Además, de disponer una colección que permita mantener un historial de los procesos ejecutados y asignarles un estado en caso de fallo.

Este no es un modelo de datos relacional sino que existe un patrón de diseño de la estructura de datos mediante documentos embebidos de la tabla Configuración en la tabla Fuente, y sus atributos varían de acuerdo a la necesidad del origen, actualmente se manejan esos campos. La tabla historial en cambio es un patrón de diseño con documento referido de Fuente en Historial.

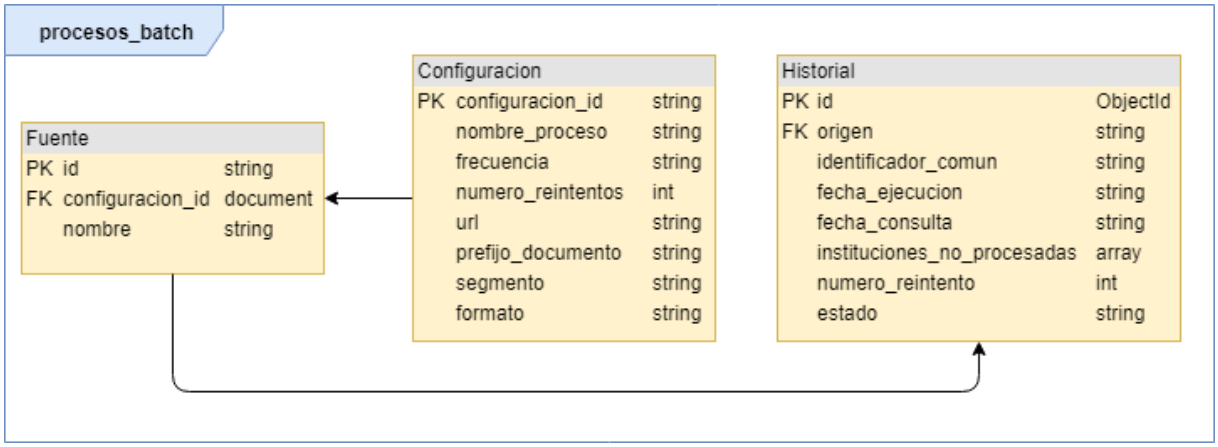



Figure 28: Modelo de datos: Procesos batch
*Fuente.* Elaboración propia

El siguiente modelo define la estructura de configuración necesaria para el Sistema de Información Gerencial como son los usuarios, módulos, roles y permisos, así como también la configuración de las instituciones financieras. Adicional, el sistema debe permitir la configuración de las cuentas contables con respecto a los productos. Nuevamente se aclara que la base de datos es una no relacional, por lo que no existe los "joins", en su defecto el modelo se refiere a documentos referenciados que serán consultados al momento de realizar búsquedas.

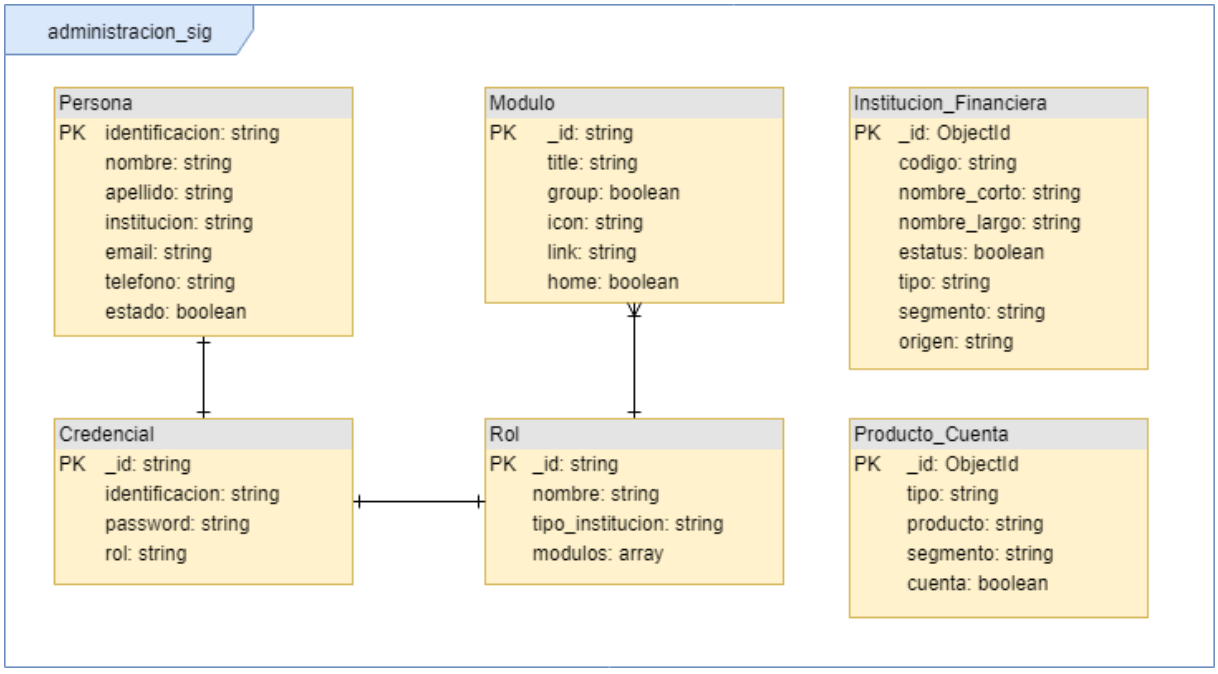



Figure 29: Modelo de datos: Administración Sistema de Información Gerencial
*Fuente.* Elaboración propia

### 5.2.4. Vista de Despliegue

En este sección se describe gráficamente la arquitectura del sistema apoyándose en diagramas de despliegue, es importante aclarar que se tienen 3 tipos de arquitectura: aplicación web, clúster de procesamiento y extracción de los datos, en los tres se integran con la base de datos ya que actúa como una fuente de información para cada caso, el cuál detallaremos a continuación.

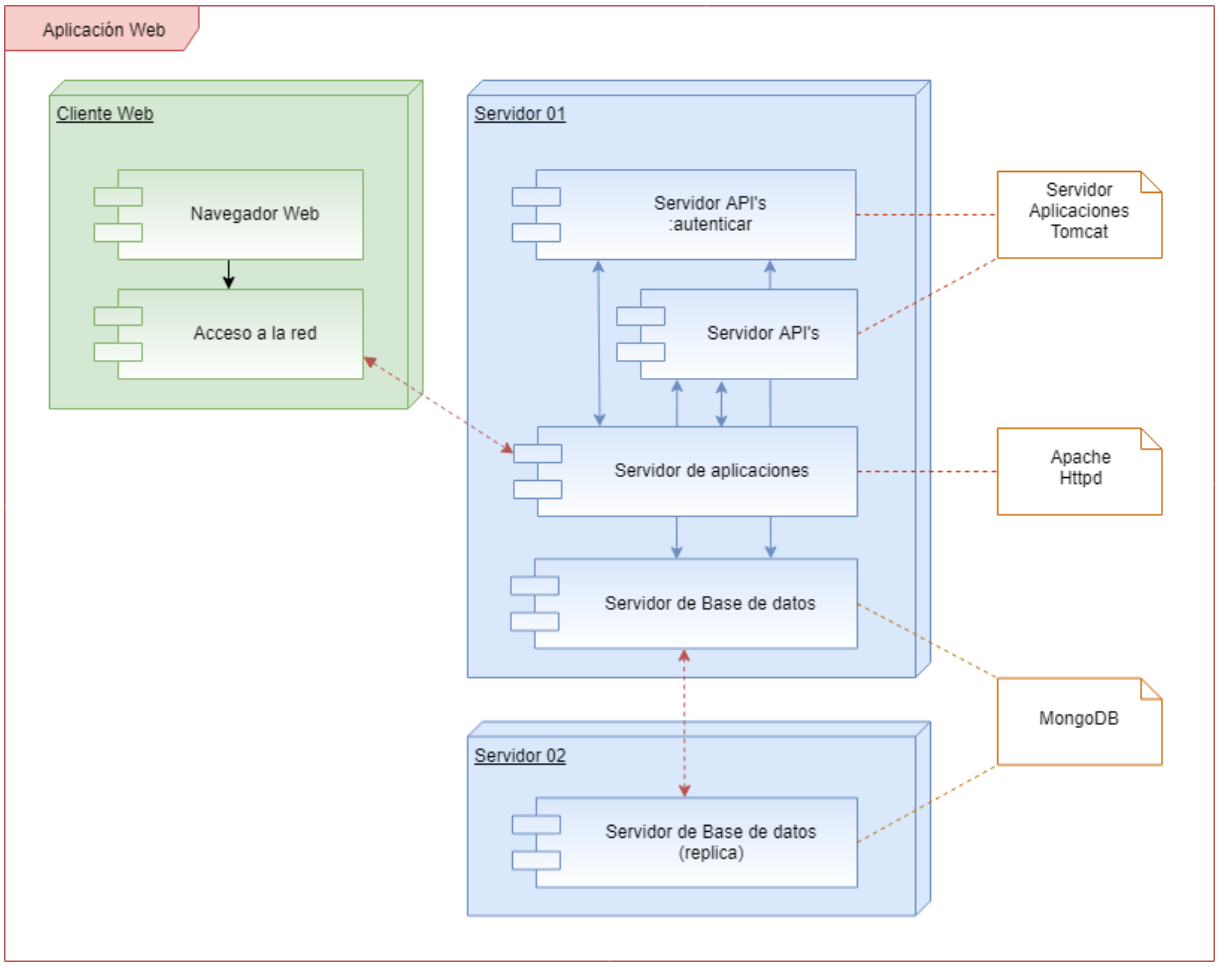



Figure 30: Diagrama de despliegue del Sistema de Información Gerencial
*Fuente.* Elaboración propia

En este gráfico 30, se puede visualizar el componente del lado del cliente, el cuál solo necesita tener un navegador web y acceso a la red, con estos dos puede ya acceder al Sistema de Información Gerencial. Del lado del sistema, se trabaja con dos servidores inicialmente, es importante aclarar que la arquitectura planteada puede permitir el crecimiento tanto vertical como horizontal, es decir se puede incrementar en disco duro, procesamiento, memoria en el mismo servidor, o adicionar un nuevo servidor como tenemos en el diagrama y habilitar la base

de datos en modo replica sets.

Se puede ver que el cliente web, accede por medio de la red al servidor de aplicaciones, el cual permite conectarse a una API de autenticación y a las API's de consultas de los datos del repositorio necesarios para los dashboard, cada conjunto de API's se conectan a la base de datos.

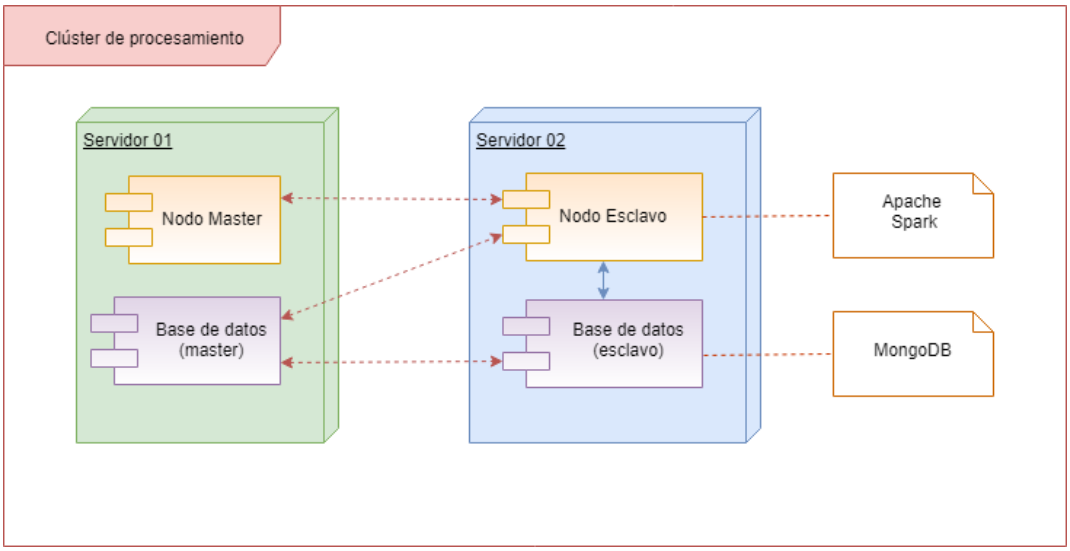



Figure 31: Diagrama de despliegue del clúster de procesamiento
*Fuente.* Elaboración propia

En la imagen que antecede 31, vemos un esquema gráfico del clúster de procesamiento, en los mismos servidores se levantó un clúster Apache Spark, con la finalidad de aprovechar el procesamiento paralelo y los beneficios que ofrece, ya que se plantea trabajar con datos masivos.

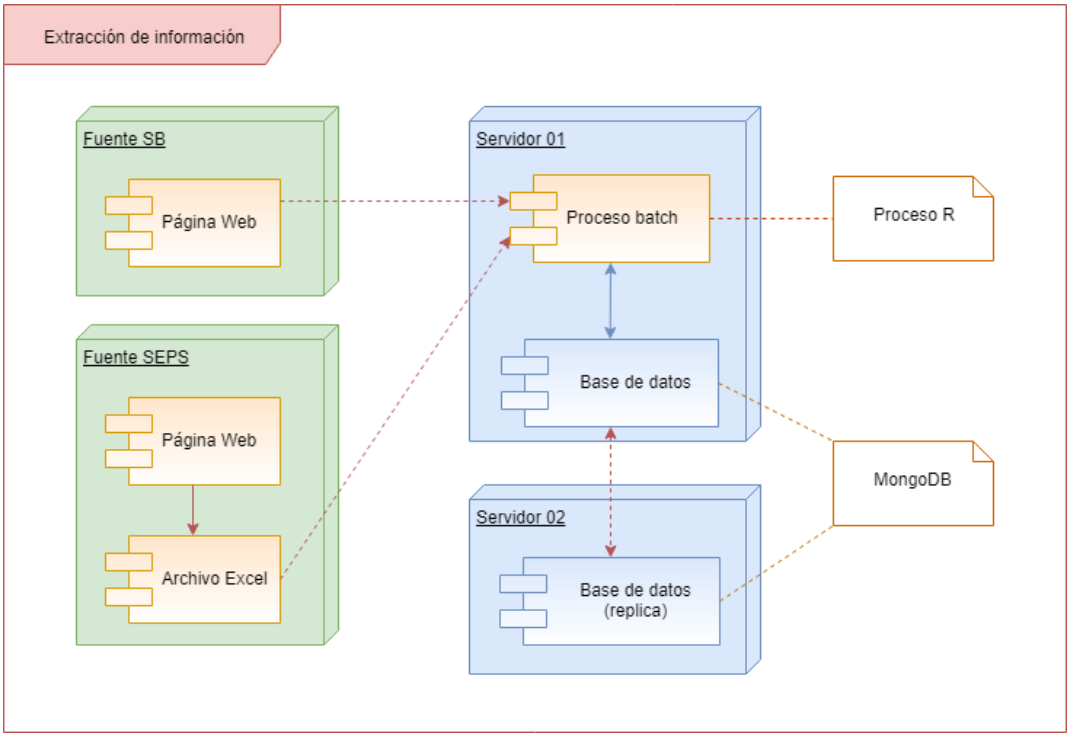



Figure 32: Diagrama de despliegue de extracción de datos
*Fuente.* Elaboración propia

En este último diagrama 32 observamos que existe un proceso automático que se encarga de leer los datos desde dos fuentes que son la SB y la SEPS, en el primer caso es una página web, mientras el segundo es un archivo excel, en ambos casos el proceso batch extrae los datos y los almacena en la base.

**Arquitectura Física**

Se levantó información de la infraestructura que debería tener el sistema, lo ideal es contar con 4 servidores, las funciones de cada servidor sería lo siguiente:

- Se tiene un servidor de autenticación que expone una API que accede a la base de datos de seguridad, donde se encuentran los usuarios y sus credenciales para permitir la autenticación de los mismos.
- Un servidor para la instalación de los micro servicios.
- Un servidor que se encargara de ejecutar el robot automático, en donde se usa el Scheduler Task del sistema operativo y extrae los datos al repositorio.
- Un servidor para la implementación de las bases de datos internas, como mencionamos en secciones anteriores, la base de datos es MongoDB, por lo que se podría hacer uso

de varios servidores más y habilitar seguridad y rendimiento en la base (replica sets, sharding). O a su vez se podría usar los mismos servidores, exceptuando el que tiene salida a la red externa.

- En los mismos servidores mencionados, se puede implementar el clúster de Apache Spark o usar servidores adicionales, de la misma forma este permite el crecimiento horizontal similar a MongoDB.

Lo anteriormente mencionado se refleja en el gráfico siguiente.

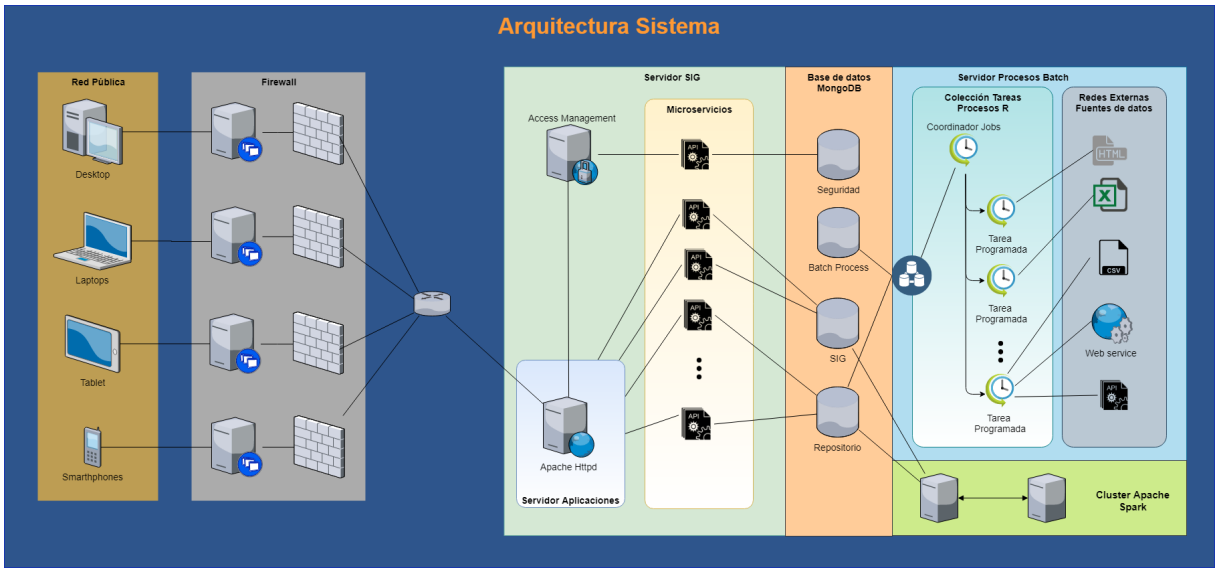



Figure 33: Arquitectura Física del Sistema
*Fuente.* Elaboración propia

### 5.2.5. Implementación

Lo ideal es implementar la plataforma del repositorio, las bases del datos del sistema y el clúster de Spark en varios nodos físicos, pero debido a que la empresa Logical Consulting dispone de un servidor se procederá a instalar dos máquinas virtuales dentro de un mismo servidor, siguiendo el mismo enfoque de la arquitectura física.

- **Implementación Servidor Sistema Operativo:** es importante definir el sistema operativo en donde esta montado la plataforma de trabajo, en este caso se trabaja con Centos 7, ya que la mayoría de componentes se ejecutan sobre ese sistema, y este se instala en cada maquina virtual.

- **Implementación Servidor de Aplicación:** el servidor a utilizar es Apache Httpd, el cuál se sencillo de instalar y fácil de administrar, se basa en configurar los archivos que vienen

en el sistema. Además, que ofrece una rápida instalación de la aplicación web ya que en sí la aplicación es de tipo Javascript, por lo que la puesta en producción del sistema es sencillo, basta con copiar y pegar la carpeta del sistema en la ruta /www del servidor.

- **Implementación Micro servicios:** la tecnología a usar para la creación de los micro servicios es SpringBoot, el cuál me permite desarrollar los CRUD necesarios de acceso a la base de datos y empaquetarlo en un .jar para luego ser desplegado sobre el servidor, la ventaja de esta tecnología es que cada micro servicio implementa un servidor de aplicaciones embebido, por lo que el administrador no se debe de preocupar por instalar un servidor y configurarlo para implementar n-servicios.

- **Implementación Servidor de base de datos:** ya se ha venido mencionado conforme se ha avanzado en cada sección, el motor de base de datos a usar es MongoDB, primero porque es una base de datos no relacional que favorece las consultas rápidas, además que es muy flexible en el almacenamiento de los registros, sobre este motor se configuró 3 bases de datos tanto para la administración del sistema, configuración de los procesos batch y el repositorio de datos, el cuál se decidió manejarlo por separado para que este sea usado no únicamente por el Sistema de Información Gerencial, por esa razón de mantenerlo aparte, si bien es cierto que el diagrama de arquitectura 33 incluye 4 bases de datos, se ha separado en 2 la seguridad y la administración del SIG, para mostrar como es el flujo de conexión a la base de datos para la autenticación de usuarios.

- **Implementación clúster Apache Spark:** esta implementación consiste en habilitar a Spark en modo clúster ejecutándose sobre dos nodos. La razón de usar Spark, es que a futuro será muy potente para llevar a cabo tareas de análisis sobre los datos, además que se esta almacenando datos históricamente y el procesamiento de estos es enorme y es ahí donde se verá el potencial de usar Spark. Para la implementación se aplicó el siguiente proceso a ambos nodos:

  1. Se parte de un sistema operativo Centos 7, el cuál debe estar instalado en cada maquina virtual.
  2. Se actualiza el nombre del host de cada servidor, cada maquina virtual tiene las siguientes direcciones IP: 192.168.100.8 y 192.168.100.9 y a cada uno se le coloca el nombre **master-server** y **slave-server** respectivamente.
  3. Proceder a descargar de la página de Apache Spark la última versión disponible de spark (https://spark.apache.org/downloads.html).

4. Una vez descargado proceder a subirlo a los dos nodos (master y slave).
5. Colocarlos dentro de la carpeta ***/opt*** el archivo descargado.
6. Se debe desempaquetar el binario de Spark, para ello usamos el siguiente comando: tar -xzf spark-2.4.2.bin-hadoop-2.7.tgz, esto lo realizamos dentro de la mismo carpeta.
7. Con la finalidad de poder agregar varias versiones de Spark en un futuro, procedemos a crear un enlace virtual a la carpeta /opt/spark-2.4.2.bin-hadoop-2.7.tgz que tendra el nombre /opt/spark, para ello ejecutamos el comando ln -s /opt/spark-2.4.2.bin-hadoop-2.7.tgz /opt/spark
8. Procedemos a aplicar un cambio al archivo de configuraciones de Spark, el cuál es el mismo tanto para el máster como esclavo, se debe añadir la siguiente línea al archivo /opt/spark/conf/spark-env.sh la configuración es: SPARK_MASTER_HOST=192.168.100.8
9. Una vez configurado ambos nodos, se procede a levantar el servicio, este difiere de uno al otro; para el caso del master se procede a ejecutar el archivo: /opt/spark/sbin/start-master.sh, para el caso del esclavo es: /opt/spark/sbin/start-slave.sh spark://192.168.100.8:7077, en este se debe especificar como parámetro la url del servidor máster, este mismo proceso se repite para todos los nodos esclavos que se vayan a agregar.
10. Al final, luego de haber levantado los servicios del clúster, se procede a verificar su funcionamiento, esto lo vemos en el enlace http://192.168.100.8:8080.

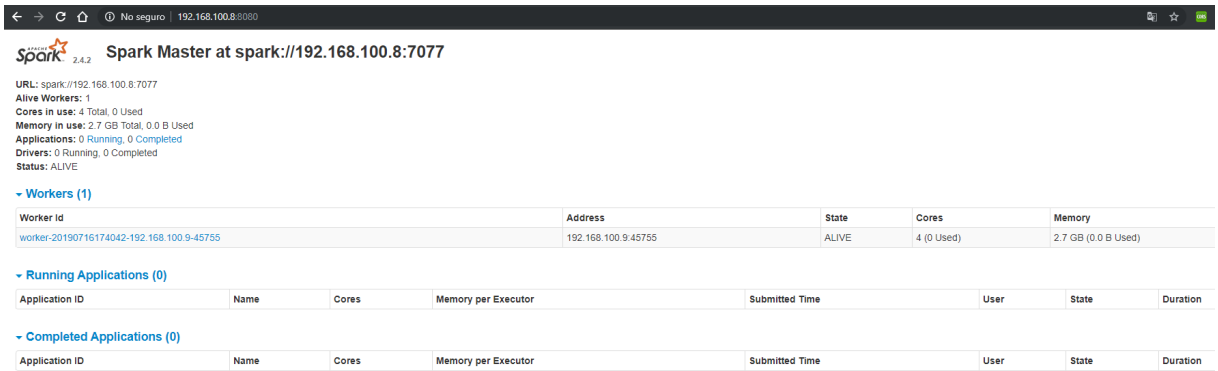



Figure 34: Consola Spark UI
*Fuente.* Elaboración propia

IMPORTANTE: Se debe saber que se puede cambiar los parámetros del puerto en donde se ejecuta el máster spark, así mismo la configuración del puerto en donde se carga la

consola de control y monitoreo de las aplicaciones spark, por defecto ambos se cargan en el puerto 7077 y 8080, estos cambios se realizan en el archivo de configuración anteriormente mencionado.

## 5.3. Construcción

En esta sección se plantea explicar los distintos aspectos que considero son importantes de todo el proceso de construcción del software propuesto, partiendo de las 4 fases explicadas con anterioridad, para ello de manera breve se expone como se construyo cada componente del proyecto, y estos son:

- Robot automático de extracción de datos
- Proceso de consolidación de datos. Python y Spark
- Clientes ligeros y librerías web
- Gestor de base de datos Mongo
- Plataforma Spring Boot para las API's
- Aseguramiento de los datos. Gravitee.IO

### 5.3.1. Robot Automático de extracción de datos

Este componente del proyecto del sistema es primordial porque es el que permite poder obtener la información de forma periódica y alimentar el repositorio de datos que es el que brinda la fuente que alimenta el sistema de información gerencial, por esa razón requirió de una gran parte del tiempo construirlo. Se definió dos procesos batch:

- **Extracción de datos:** se ejecuta una sola vez por cada periodo de tiempo configurado, puede ser diario, mensual y anual, por esa razón este batch se ejecuta todos los días y dependiendo de la configuración extraerá los datos, este proceso lo realiza una sola vez, valida que no se tenga una ejecución previa para el mismo periodo, si lo existe no lo realiza y termina, sino existe entonces procede a leer los datos del origen y descargar la información, en este caso se pueden presentar dos situaciones: que no existe todavía datos o al momento de extraer datos se corto el acceso, cualquiera que fuese el caso o bien no existen datos o se tienen datos incompletos, se procede a colocar este proceso de extracción, para el periodo de ejecución, con estado de reintento y termina el proceso.

- **Reprocesos:** se ejecuta diariamente, y reprocesa aquellos que quedaron con un estado de reintento, es decir todos aquellos que fallaron se reintentará por un periodo de tiempo definido el cual es configurable, si luego de terminar este periodo, no se logro extraer información entonces significa que los datos del origen han cambiado o no esta configurado adecuadamente, en su defecto el administrador debe proceder a realizar la extracción de datos manualmente.

Para poder controlar los procesos de extracción de datos y facilitar al usuario agregar nuevas fuentes de información se creo una base de datos con la siguiente estructura:

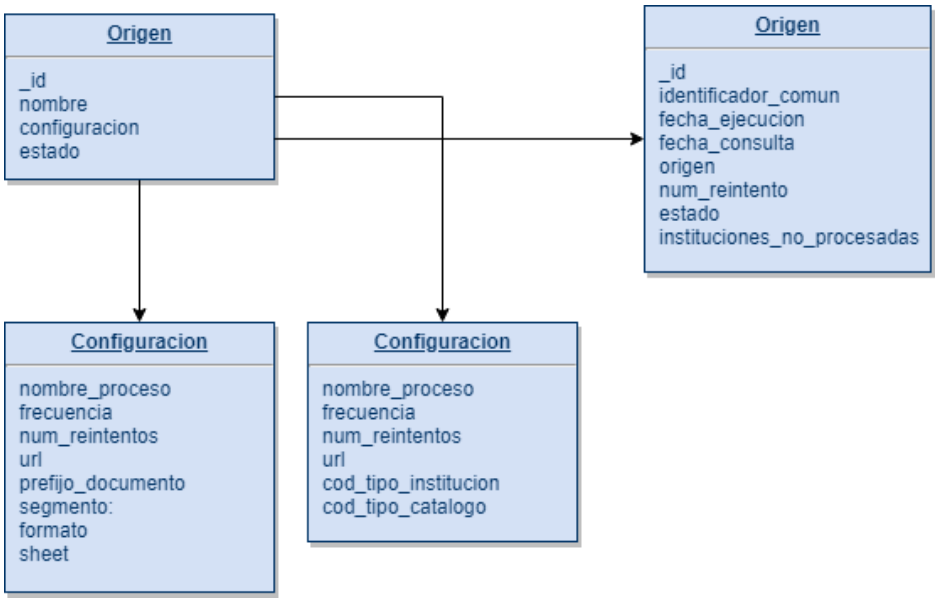



Figure 35: Estructura Base de Datos Procesos Batch
*Fuente.* Elaboración propia

La base de datos cuyo nombre es **procesos_batch**, fue creado en el gestor de base de datos Mongo debido a la flexibilidad que brinda a la hora de crear modelos, para ello usamos un patrón de relación **uno-a-uno con documentos embebidos**, y esto se refleja en la [Figura 35] existen dos colecciones principales que son Origen e Historial, ambas conforman la configuración necesaria para los batch pero el documento embebido en la colección origen de nombre **configuración** contiene una estructura variable, eso se refleja en las dos tablas adicionales de la estructura, con esto logramos agregar nuevos orígenes con distintos parámetros de configuración, actualmente se tiene orígenes con dos tipos de configuración.

Elaborada la base de datos, se creó el proceso que lee las configuraciones, para este caso se necesito de un lenguaje que permitiera seguir creando nuevos procesos con distintos orígenes y solo configurar el nombre del proceso por base de datos, por ello se tomo al

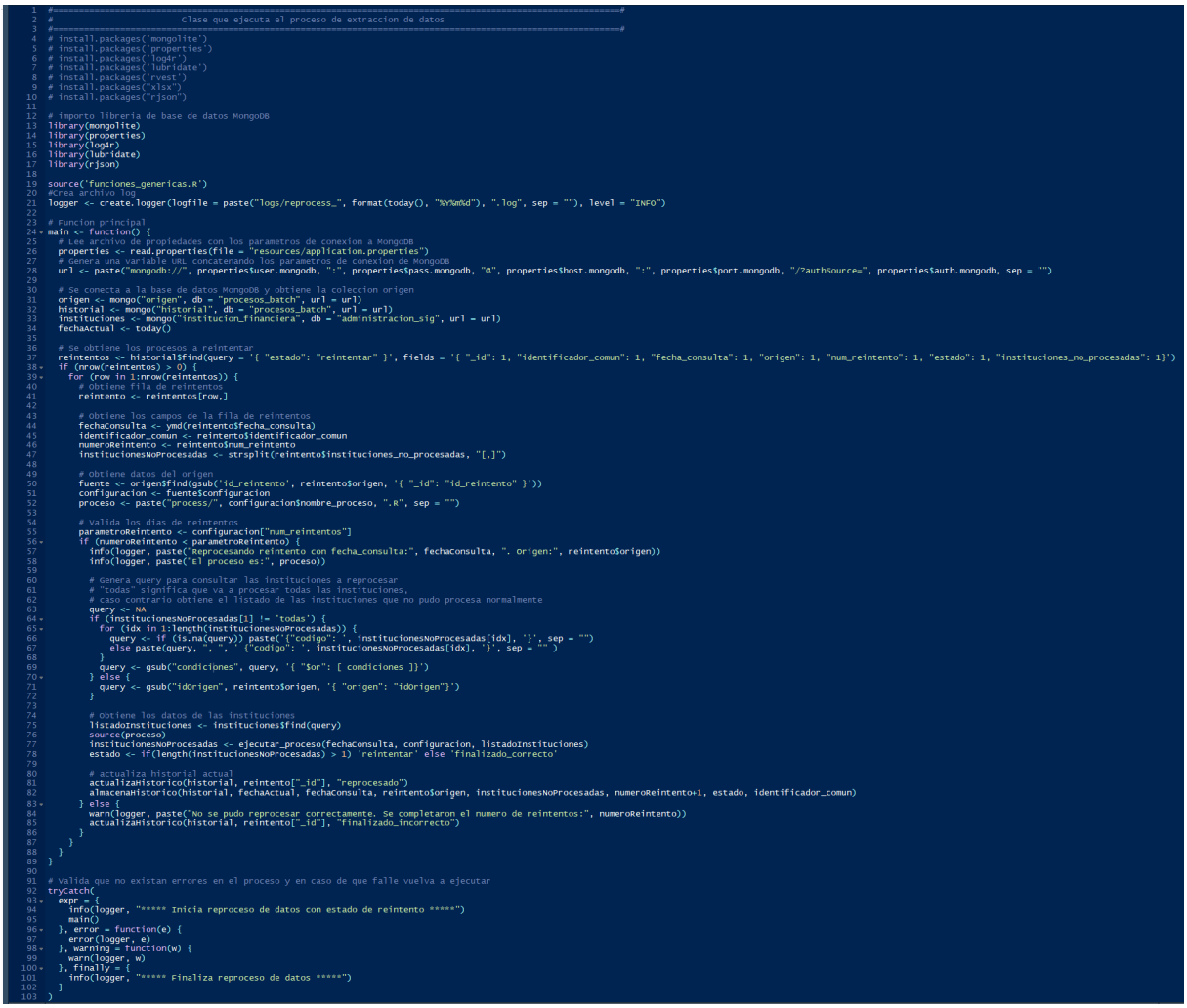
**Lenguaje R** ya que permite añadir nuevas fuentes por medio de la función **source**, como se explica anteriormente, debido a que existen dos procesos batch se busco la forma que se cree un proceso único que sirve tanto para el proceso normal como para el reproceso, para lograr esto se definió tener una función genérica que es la función principal de cada tipo de origen la cual recibe el nombre de **ejecutar_proceso**, recibe tres parámetros que son **fecha de consulta (fechaConsulta), configuración (configuracion) e instituciones**, es decir, todo proceso batch de extracción de datos de algún nuevo origen debe tener esta función.

La clase o proceso R inicial que carga los datos de los orígenes normalmente se denomina **batch_extraccion** y el proceso que se encarga de ejecutar los reprocesos se denomina **batch_reproceso**, si hacemos una analogía con Java, estos procesos vienen a ser las clases principales que contienen un método main, a continuación el código fuente empleado para cada caso:

```r
#=============================================================================================#
#                          Clase que ejecuta el proceso de extraccion de datos
#=============================================================================================#
# install.packages('mongolite')
# install.packages('properties')
# install.packages('log4r')
# install.packages('lubridate')
# install.packages('rvest')
# install.packages("xlsx")
# install.packages("rjson")

# importo libreria de base de datos MongoDB
library(mongolite)
library(properties)
library(log4r)
library(lubridate)
library(rjson)

source('funciones_genericas.R')
#Crea archivo log
logger <- create.logger(logfile = paste("logs/process_", format(today(), "%Y%m%d"), ".log", sep = ""), level = "INFO")

# Funcion principal
main <- function() {
  # Lee archivo de propiedades con los parametros de conexion a MongoDB
  properties <- read.properties(file = "resources/application.properties")
  # Genera una variable URL concatenando los parametros de conexion de MongoDB
  url <- paste("mongodb://", properties$user.mongodb, ":", properties$pass.mongodb, "@", properties$host.mongodb, ":", properties$port.mongodb, "/?authSource=", properties$auth.mongodb, sep = "")

  # Se conecta a la base de datos MongoDB y obtiene la coleccion origen
  origen <- mongo("origen", db = "procesos_batch", url = url)
  historial <- mongo("historial", db = "procesos_batch", url = url)

  # Retorna los orígenes que tengan estado activo
  listaOrigenes <- origen$find(query = '{ "estatus": true}', fields = '{ "_id": 1 }')

  # Obtiene fecha actual de procesamiento
  # fechaActual = today()
  fechaActual <- ymd('20190201')
  # Esta funcion por cada origen ejecuta el proceso de batch
  f <- function(origenes, salida) {
    # Obtiene el documento de origen, la configuracion del origen y el nombre de proceso
    fuente <- origen$find(gsub('idorigen', origenes['_id'], '{ "_id": "idOrigen"}'))
    configuracion <- fuente$configuracion
    proceso <- paste("process/", configuracion$nombre_proceso, ".R", sep = "")

    info(logger, gsub("nombre_proceso", configuracion$nombre_proceso, "=== Ejecutando origen: nombre_proceso ==="))
    info(logger, gsub("fecha_actual", fechaActual, "Fecha Actual: fecha_actual"))

    # Obtiene la fecha consulta y obtiene el historial de ejecucion
    fechaConsulta <- obtenerPeriodo(configuracion$frecuencia, fechaActual)
    # fechaConsulta <- ymd("20190430")
    info(logger, gsub("fecha_consulta", fechaConsulta, "Fecha Consulta: fecha_consulta"))

    queryHistorial <- '{ "fecha_consulta": "fechaConsulta", "origen": "idOrigen" }'
    queryHistorial <- gsub('fechaConsulta', fechaConsulta, queryHistorial)
    queryHistorial <- gsub('idOrigen', origenes['_id'], queryHistorial)
    historialDF <- as.data.frame(historial$find(queryHistorial))

    # Verifica historial (Si no existe un antecedente de ejecucion)
    if (nrow(historialDF) == 0) { # No existe historial
      # Obtengo listado de instituciones financieras
      instituciones <- mongo("institucion_financiera", db = "administracion_sig", url = url)
      listadoInstituciones <- instituciones$find(gsub("idOrigen", origenes['_id'], '{ "origen": "idOrigen", "estatus": true }'))

      # Llama al proceso
      if (file.exists(proceso)) {
        source(proceso)
        institucionesNoProcesadas <- ejecutar_proceso(fechaConsulta, configuracion, listadoInstituciones)
        estado <- if(length(institucionesNoProcesadas) > 1) 'reintentar' else 'finalizado_correcto'
        identificador_comun <- paste(origenes['_id'], "_", fechaConsulta, sep = '')
        almacenaHistorico(historial, fechaActual, fechaConsulta, origenes['_id'], institucionesNoProcesadas, 0, estado, identificador_comun)
      }
    } else {
      warn(logger, gsub("fecha_consulta", fechaConsulta, "Ya se ejecuto proceso para el periodo actual: fecha_consulta"))
    }
  }

  # Similar a realizar un for por cada fila del dataframe listaOrigenes ejecuta la funcion f
  apply(listaOrigenes, 1, f)
}

# Valida que no existan errores en el proceso y en caso de que falle vuelva a ejecutar
tryCatch(
  expr = {
    info(logger, "***** Inicia proceso de consulta de datos en origenes *****")
    main()
  }, error = function(e) {
    error(logger, e)
  }, warning = function(w) {
    warn(logger, w)
  }, finally = {
    info(logger, "***** Finaliza proceso de consulta de datos en origenes *****")
  }
)
```

Figure 36: Proceso main de extracción de datos
*Fuente.* Elaboración propia

Algo similar se tiene para el caso de los reprocesos:



```r
#=============================================================================================#
#                          Clase que ejecuta el proceso de extraccion de datos
#=============================================================================================#
# install.packages('mongolite')
# install.packages('properties')
# install.packages('log4r')
# install.packages('lubridate')
# install.packages('rvest')
# install.packages("xlsx")
# install.packages("rjson")

# Importo libreria de base de datos MongoDB
library(mongolite)
library(properties)
library(log4r)
library(lubridate)
library(rjson)

source('funciones_genericas.R')
#Crea archivo log
logger <- create.logger(logfile = paste("logs/reprocess_", format(today(), "%Y%m%d"), ".log", sep = ""), level = "INFO")

# Funcion principal
main <- function() {
  # Lee archivo de propiedades con los parametros de conexion a MongoDB
  properties <- read.properties(file = "resources/application.properties")
  # Genera una variable URL concatenando los parametros de conexion de MongoDB
  url <- paste("mongodb://", properties$user.mongodb, ":", properties$pass.mongodb, "@", properties$host.mongodb, ":", properties$port.mongodb, "/?authSource=", properties$auth.mongodb, sep = "")

  # Se conecta a la base de datos MongoDB y obtiene la coleccion origen
  origen <- mongo("origen", db = "procesos_batch", url = url)
  historial <- mongo("historial", db = "procesos_batch", url = url)
  instituciones <- mongo("institucion_financiera", db = "administracion_sig", url = url)
  fechaActual <- today()

  # Se obtiene los procesos a reintentar
  reintentos <- historial$find(query = '{ "estado": "reintentar" }', fields = '{ "_id": 1, "identificador_comun": 1, "fecha_consulta": 1, "origen": 1, "num_reintento": 1, "estado": 1, "instituciones_no_procesadas": 1}')
  if (nrow(reintentos) > 0) {
    for (row in 1:nrow(reintentos)) {
      # Obtiene fila de reintentos
      reintento <- reintentos[row,]

      # Obtiene los campos de la fila de reintentos
      fechaConsulta <- ymd(reintento$fecha_consulta)
      identificador_comun <- reintento$identificador_comun
      numeroReintento <- reintento$num_reintento
      institucionesNoProcesadas <- strsplit(reintento$instituciones_no_procesadas, "[,]")

      # obtiene datos del origen
      fuente <- origen$find(gsub('id_reintento', reintento$origen, '{ "_id": "id_reintento" }'))
      configuracion <- fuente$configuracion
      proceso <- paste("process/", configuracion$nombre_proceso, ".R", sep = "")

      # valida los dias de reintentos
      parametroReintento <- configuracion["num_reintentos"]
      if (numeroReintento < parametroReintento) {
        info(logger, paste("Reprocesando reintento con fecha_consulta:", fechaConsulta, ". Origen:", reintento$origen))
        info(logger, paste("El proceso es:", proceso))

        # Genera query para consultar las instituciones a reprocesar
        # "todas" significa que va a procesar todas las instituciones,
        # caso contrario obtiene el listado de las instituciones que no pudo procesa normalmente
        query <- NA
        if (institucionesNoProcesadas[1] != 'todas') {
          for (idx in 1:length(institucionesNoProcesadas)) {
            query <- if (is.na(query)) paste('{"codigo": ', institucionesNoProcesadas[idx], '}', sep = "")
            else paste(query, ", ", '{"codigo": ', institucionesNoProcesadas[idx], '}', sep = "" )
          }
          query <- gsub("condiciones", query, '{ "$or": [ condiciones ]}')
        } else {
          query <- gsub("idOrigen", reintento$origen, '{ "origen": "idOrigen"}')
        }

        # obtiene los datos de las instituciones
        listadoInstituciones <- instituciones$find(query)
        source(proceso)
        institucionesNoProcesadas <- ejecutar_proceso(fechaConsulta, configuracion, listadoInstituciones)
        estado <- if(length(institucionesNoProcesadas) > 1) 'reintentar' else 'finalizado_correcto'

        # actualiza historial actual
        actualizaHistorico(historial, reintento["_id"], "reprocesado")
        almacenaHistorico(historial, fechaActual, fechaConsulta, reintento$origen, institucionesNoProcesadas, numeroReintento+1, estado, identificador_comun)
      } else {
        warn(logger, paste("No se pudo reprocesar correctamente. Se completaron el numero de reintentos:", numeroReintento))
        actualizaHistorico(historial, reintento["_id"], "finalizado_incorrecto")
      }
    }
  }
}

# Valida que no existan errores en el proceso y en caso de que falle vuelva a ejecutar
tryCatch(
  expr = {
    info(logger, "***** Inicia reproceso de datos con estado de reintento *****")
    main()
  }, error = function(e) {
    error(logger, e)
  }, warning = function(w) {
    warn(logger, w)
  }, finally = {
    info(logger, "***** Finaliza reproceso de datos *****")
  }
)
```

Figure 37: Proceso main de reprocesos de datos
*Fuente.* Elaboración propia

Revisando el código vemos que en la línea 68 y 76 por cada batch se llama a la función source, esto lo que hace es añadir el proceso que toma los datos del origen, varía dependiendo del origen, y posterior a ello en la línea 69 y 77 se procede a llamar a la función ejecutar_proceso que se mencionó anteriormente.

Actualmente, se tiene 5 orígenes, uno de ellos lee los datos de una página web y los otros leen de un archivo de excel. En cualquier caso, el proceso R se encarga de leer los datos y almacenarlos en el repositorio Mongo.

Por cada origen tenemos creados 3 procesos R, uno para el caso de las instituciones bancarias privadas, otro para las cooperativas del segmento 1 al 3 y un proceso para el caso de las mutualistas. Como vemos usamos un mismo proceso para extraer tres orígenes que son

las cooperativas, únicamente modificamos en la base de datos las configuraciones para que se ejecute en el mismo. Cada uno de estos procesos deben tener la función **ejecutar_proceso**, como podemos ver en el siguiente segmento de código R:

```r
# ================================================================================
#             PROCESO DE EXTRACCION DE DATOS DE UNA PAGINA HTML DE LA PAGINA DE LA SB
# ================================================================================

library(lubridate)
library(rvest)
# Lee archivo de propiedades con los parametros de conexion a MongoDB
properties <- read.properties(file = "resources/application.properties")
# Genera una variable URL concatenando los parametros de conexion de MongoDB
conn <- paste("mongodb://", properties$user.mongodb, ":", properties$pass.mongodb, "@", properties$host.mongodb, ":", properties$port.mongodb, "/?authSource=",

# Funcion principal
ejecutar_proceso <- function(fechaConsulta, configuracion, instituciones) {
  # A partir de la fechaConsulta se toma el año y el mes
  anio <- format(fechaConsulta, "%Y")
  mes <- format(fechaConsulta, "%m")

  # Define lista de instituciones no procesadas
  # Obtiene los datos por cada institucion
  institucionesNoProcesadas <- c("Inicio")
  balance <- mongo("balance", db = "repositorio", url = conn)
  for (row in 1:nrow(instituciones)) {
    # Define la URL de donde va a leer
    url <- configuracion["url"]
    parameters <- names(configuracion)
    for (parameter in parameters){
      if (parameter != "url" && parameter != "nombre_proceso" && parameter != "frecuencia" && parameter != "num_reintentos")
        url <- gsub(paste("!", parameter, sep = ""), configuracion[parameter], url)
    }

    # Setea los parametros de anio, mes y codigo de institucion
    url <- gsub("!anio", anio, url)
    url <- gsub("!mes", mes, url)
    url <- gsub("!cod_institucion", instituciones[row, "codigo"], url)

    # Lee de la página web
    info(logger, paste("Institucion:", instituciones[row, "nombre_corto"]))
    info(logger, paste("URL Consulta:", url))
    webpage <- read_html(url)
    datos <- webpage %>% html_nodes('table')
    lista <- datos[[3]] %>% html_table()
```

Figure 38: Origen instituciones bancarias privadas. Proceso HTML.
*Fuente.* Elaboración propia

Para visualizar el código completo de cada fuente, por favor remitirse a la sección de Anexo III. Procesos R: Extracción Fuentes de Información (8).

### 5.3.2. Proceso de consolidación de datos. Python y Spark

Extraída la información de las fuentes y cargadas en el repositorio, se procede a trabajar con esos datos, para ello se hace uso de la herramienta Spark, ya que permite trabajar con un gran volumen de información y de forma paralela con el fin de optimizar el tratamiento de los datos.

El objetivo de esto es crear una aplicación Spark, que se encargue de leer los datos de la base de MongoDB y realice un tratamiento, agrupación o consolidación de los datos y generar nuevas estructuras de información que serán almacenadas en el mismo repositorio Mongo, es decir al final se tiene una nueva información homogénea que sirven como datos de origen que leerá el Sistema de información Gerencial.

El lenguaje utilizado para la construcción de una aplicación Spark es Python, para poder usar spark y python es necesario instalar la librería de spark denominada **PySpark**, esta in-

cluye un core de funciones y conjunto de librerías como **Spark SQL**, además para que Spark se conecte a Mongo es necesario agregar un plugin denominado **org.mongodb.spark:mongo-spark-connector_2.11:2.4.0**, en donde los últimos dígitos indican la versión de Scala y la versión de Spark, es importante tener en cuenta estos, ya que al final se debe subir la aplicación Spark a un clúster Spark que lo ejecute.

Partiendo de la premisa que se conoce Python y todo lo que rodea en la instalación de un ambiente virtual y posterior instalación de paquetes, se procede a explicar la estructura base de archivos, del proyecto Spark + Python + MongoDB:

- consolidador
  - consolidador
    - data
      - __init__.py
      - data_source.ini
      - consolidador_posicion.py
      - consolidador_variacion.py
    - __init__.py
    - __main__.py
  - setup.py

Los archivos que nos interesa revisar es el data_source.ini, consolidador_posicion.py y consolidador_variacion.py, estos en realidad conforman las aplicaciones spark propiamente, los demás son necesarios para empaquetar el proyecto. El archivo .ini contiene los parámetros de configuración que son:

- **app_name:** es el nombre de la aplicación Spark, necesaria con la que se podrá revisar en la interfaz de spark para analizar su estado.
- **master_url:** es la dirección del servidor máster en un clúster spark, en este proyecto es **spark://192.168.100.8:7077**, en ambientes de desarrollo o pruebas suele ir **local**.
- **mongodb_url:** es la dirección en donde se encuentra el motor de base de datos de mongodb, por ejemplo, **mongo://localhost:27017**.
- **mongo_jar:** este es el plugin de mongo para Spark, con este se podrá conectar a la base de datos, escribir, leer, actualizar y eliminar registros de la base, en este caso es **org.mongodb.spark:mongo-spark-connector_2.11:2.4.0**.

Los archivos .py es en donde se tiene toda la lógica de una aplicación spark, a continuación una parte del código empleado para leer y agrupar los datos.

consolidador ▸ consolidador ▸ data ▸ data_source.py

```python
# coding=utf-8
from pyspark import SparkConf, SparkContext
from pyspark.sql import SparkSession

from configparser import ConfigParser # se debe cambiar de configparse a ConfigParser en produccion
import os


class DataSource:
    def __init__(self, fecha):
        ROOT_DIR = os.path.dirname(os.path.abspath(__file__))

        config = ConfigParser()
        config.read(os.path.join(ROOT_DIR, 'data_source.ini'))
        master_url = config.get('spark', 'master_url')
        app_name = config.get('spark', 'app_name')
        mongo_jar = config.get('spark', 'mongo_jar')

        self.mongodb = config.get('spark', 'mongodb_url')
        self.fecha = fecha

        conf = SparkConf().setAppName(app_name).setMaster(master_url).set("spark.jars.packages", mongo_jar)

        self.sc = SparkContext(conf=conf)
        self.spark = SparkSession.builder.config(conf=conf).config("spark.mongodb.output.uri", self.mongodb).getOrCreate()

    def run(self):
        self.spark.read.format("com.mongodb.spark.sql.DefaultSource").option(
            "uri", self.mongodb + "administracion_sig.producto_cuenta").load().createOrReplaceTempView("productoCuentas")

        self.spark.read.format("com.mongodb.spark.sql.DefaultSource").option(
            "uri", self.mongodb + "repositorio.balance").load().filter("fecha = '" + self.fecha + "'").createOrReplaceTempView("balance")

        self.spark.read.format("com.mongodb.spark.sql.DefaultSource").option(
            "uri", self.mongodb + "administracion_sig.institucion_financiera").load().createOrReplaceTempView("instituciones")

        self.spark.sql("SELECT b.fecha, i.tipo as tipo_institucion, i.nombre_corto, i.nombre_largo, pc.producto, pc.tipo, pc.segmento, sum(b.saldo) as saldo \
            FROM balance b INNER JOIN productoCuentas pc ON b.cuenta = pc.Cuenta  \
            LEFT JOIN instituciones i on i.codigo = b.institucion \
```

Figure 39: Aplicación Spark desarrollada en Python.
*Fuente.* Elaboración propia

Existen dos aplicaciones Spark, la primera que se encarga de generar una agrupación de los saldos de las captaciones y colocaciones por institución y por fecha, de hecho estas aplicaciones reciben como entrada la fecha de ejecución, este archivo es **consolidador_posicion.py**, después de tener esta información agrupada, a partir de una nueva aplicación spark se realiza la comparación de la información actual con la de una fecha anterior para generar las respectivas variaciones que resulta en una nueva estructura, este archivo es **consolidador_variacion.py**, que sirve como entrada para el sistema de información gerencial.

Para visualizar el código completo de cada aplicación Spark, por favor remitirse a la sección de Anexo IV. Aplicaciones Spark (8).

Luego de crear la aplicación y realizar el empaquetado del proyecto, se procede a subirlo al servidor en donde se encuentre cargado un clúster spark, y para ejecutarlo se lo realiza por medio del comando **spark-submit**, existe una ejecución del comando por cada archivo py previamente mencionado, que menciono a continuación:

- **Posición:**
  spark-submit –packages org.mongodb.spark:mongo-spark-connector_2.12:2.4.0 /root/-consolidador/consolidador/__main__.py 2019-03-31 posicion
- **Variación:**
  spark-submit –packages org.mongodb.spark:mongo-spark-connector_2.12:2.4.0 /root/-consolidador/consolidador/__main__.py 2019-03-31 variacion

Revisando el comando spark-submit, este recibe como parámetros el paquete de mongodb, y el archivo py principal, en este caso es el __main__.py, el cual a su vez recibe también como parámetro la fecha de ejecución y una palabra que puede ser "posicion" o "variacion", que indica el proceso a ejecutar, es importante observar que no se hace uso de tildes, ni caracteres especiales.

### 5.3.3. Construcción de microservicios de consulta y administración

Como se desea acceder desde un ambiente web a una base de datos Mongo es necesario exista una interfaz intermedia que permita la conexión entre estas dos partes, para ello se construyó un conjunto de servicios o como se suele denominar API's que en si son un conjunto de procesos (código) que forman una librería que conforma una capa de abstracción a la cual se conectaran múltiples aplicaciones.

Bajo este concepto se planteó crear dos *API's* que conectan al SIG con el repositorio de datos y la administración del sistema, para ello se uso una tecnología que permite una rápida construcción de estos servicios, de hecho suelen denominarse **microservicios** debido a que son porciones de código pequeños e independientes los cuales se acoplan de forma fácil y flexible, esta tecnología se denomina Spring Boot, estos generan servicios que se empaquetan en un tipo **jar** y al momento de ejecutarse levantan un servidor embebido por lo general Tomcat, y de esta forma desplegar una API de forma rápida.

#### Microservicio "repositorio"

Este permite realizar las consultas a las tablas posición activas y pasivas, y variación activas y pasivas. Este servicio es importante ya que será el encargado de proveer la información para los dashboard del sistema de información gerencial.

Se tiene dos interfaces de entrada que corresponden a las captaciones y colocaciones

que las he denominado Activas y Pasivas (debido al nombre del tipo de cuentas del balance general), a continuación el código fuente de para las consultas de las pasivas:

```java
package ec.fin.logical.repositorio.controllers;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.bind.annotation.RestController;

import ec.fin.logical.repositorio.models.PosicionPasivas;
import ec.fin.logical.repositorio.models.VariacionPasivas;
import ec.fin.logical.repositorio.services.PosicionPasivasService;

@RestController
@RequestMapping("/pasivas")
public class PosicionPasivasRest {

    @Autowired
    private PosicionPasivasService service;

    @CrossOrigin("*")
    @RequestMapping(value = "/posicion/", method = RequestMethod.GET)
    @ResponseBody
    public List<PosicionPasivas> obtenerTodos() {
        try {
            return service.findAll();
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }
    }

    @CrossOrigin("*")
    @RequestMapping(value = "/posicion/buscar/fecha/{fecha}", method = RequestMethod.GET)
    @ResponseBody
    public List<PosicionPasivas> buscarPorFecha(@PathVariable(value = "fecha") String fecha) {
        try {
            return service.buscarPosicionPorFecha(fecha);
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }
    }

    @CrossOrigin("*")
    @RequestMapping(value = "/posicion/buscar/fecha/{fecha}/{tipoInstitucion}", method = RequestMethod.GET)
    @ResponseBody
    public List<PosicionPasivas> buscarPorFecha(@PathVariable(value = "fecha") String fecha, @PathVariable(value = "tipoInstitucion") String tipoInstitucion) {
        try {
            return service.buscarPosicionPorFecha(fecha, tipoInstitucion);
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }
    }

    @CrossOrigin("*")
    @RequestMapping(value = "/variacion/buscar/fecha/{fecha}", method = RequestMethod.GET)
    @ResponseBody
    public List<VariacionPasivas> buscarVariacionPorFecha(@PathVariable(value = "fecha") String fecha) {
        try {
            return service.buscarVariacionPorFecha(fecha);
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }
    }

    @CrossOrigin("*")
    @RequestMapping(value = "/variacion/buscar/fecha/{fecha}/{tipoInstitucion}", method = RequestMethod.GET)
    @ResponseBody
    public List<VariacionPasivas> buscarVariacionPorFecha(@PathVariable(value = "fecha") String fecha, @PathVariable(value = "tipoInstitucion") String tipoInstitucion) {
        try {
            return service.buscarVariacionPorFecha(fecha, tipoInstitucion);
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }
    }
}
```

Figure 40: Interfaz rest del servicio pasivas
*Fuente.* Elaboración propia

El código siguiente es la interfaz de consulta para las activas que de igual forma indican la interfaz de entrada desde el sistema de información gerencial.

```java
package ec.fin.logical.repositorio.controllers;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.bind.annotation.RestController;

import ec.fin.logical.repositorio.models.PosicionActivas;
import ec.fin.logical.repositorio.models.VariacionActivas;
import ec.fin.logical.repositorio.services.PosicionActivasService;

@RestController
@RequestMapping("/activas")
public class PosicionActivasRest {

    @Autowired
    private PosicionActivasService service;

    @CrossOrigin("*")
    @RequestMapping(value = "/posicion", method = RequestMethod.GET)
    @ResponseBody
    public List<PosicionActivas> obtenerTodos() {
        try {
            return service.findAll();
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }
    }

    @CrossOrigin("*")
    @RequestMapping(value = "/posicion/buscar/fecha/{fecha}", method = RequestMethod.GET)
    @ResponseBody
    public List<PosicionActivas> buscarPosicionPorFecha(@PathVariable(value = "fecha") String fecha) {
        try {
            return service.buscarPosicionPorFecha(fecha);
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }
    }

    @CrossOrigin("*")
    @RequestMapping(value = "/posicion/buscar/fecha/{fecha}/{tipoInstitucion}", method = RequestMethod.GET)
    @ResponseBody
    public List<PosicionActivas> buscarPosicionPorFecha(@PathVariable(value = "fecha") String fecha, @PathVariable(value = "tipoInstitucion") String tipoInstitucion) {
        try {
            return service.buscarPosicionPorFecha(fecha, tipoInstitucion);
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }
    }

    @CrossOrigin("*")
    @RequestMapping(value = "/variacion/buscar/fecha/{fecha}", method = RequestMethod.GET)
    @ResponseBody
    public List<VariacionActivas> buscarVariacionPorFecha(@PathVariable(value = "fecha") String fecha) {
        try {
            return service.buscarVariacionPorFecha(fecha);
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }
    }

    @CrossOrigin("*")
    @RequestMapping(value = "/variacion/buscar/fecha/{fecha}/{tipoInstitucion}", method = RequestMethod.GET)
    @ResponseBody
    public List<VariacionActivas> buscarVariacionPorFecha(@PathVariable(value = "fecha") String fecha, @PathVariable(value = "tipoInstitucion") String tipoInstitucion) {
        try {
            return service.buscarVariacionPorFecha(fecha, tipoInstitucion);
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }
    }
}
```

Figure 41: Interfaz rest del servicio activas
*Fuente.* Elaboración propia

En Spring Boot para definir a una clase como un elemento que se expondrá como un servicio rest, se lo indica con la anotación *@RestController*, seguido del path por el cuál se buscará el enlace desde la aplicación web para el caso de las pasivas es /**pasivas** y para el de activas es /**activas**.

Estas clases contienen los métodos que son del tipo **GET**, tanto para activas y pasivas se

tiene dos métodos que devuelven los datos de las posiciones y variaciones respectivamente, para cada uno se envía como parámetro la fecha de consulta y el tipo de institución, esto llamará al repositorio mongo que devolverá los datos, el retorno de los datos están en formato *JSON*.

Como se trabaja con la plataforma Spring Boot, este crea un proyecto del tipo *Maven* el cual contiene un archivo **pom.xml** aquí se define una dependencia que indicará la base de datos a la cuál se conecta, en este caso mongodb:

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-data-mongodb</artifactId>
  <scope>compile</scope>
</dependency>
```

Figure 42: Dependencia Mongo para proyecto Spring Boot
*Fuente.* Elaboración propia

Al definir esta dependencia, Spring Boot ya reconoce que la base de datos de este microservicio es del tipo MongoDB. Para definir los parámetros del servidor se hace uso del archivo application.properties, los datos que se presentan a continuación son de prueba:

```
spring.data.mongodb.host=localhost
spring.data.mongodb.port=27017
spring.data.mongodb.authentication-database=admin
spring.data.mongodb.username=admin
spring.data.mongodb.password=mipassword
spring.data.mongodb.database=repositorio
server.port=9090
```

Figure 43: Archivo de propiedades del microservicio "repositorio"
*Fuente.* Elaboración propia

En el mismo archivo de propiedades se añaden todas las configuraciones necesarias que leerá el microservicio, en la [Figura 43] vemos un parámetro adicional a los de la base de datos que es **server.port = 9090** este indicará en que puerto se expondrá la API.

Definido las configuraciones de la base de datos, se define la interfaz de consulta a la base, en las siguientes imágenes tenemos el caso de activas y pasivas.

```java
package ec.fin.logical.repositorio.dao;

import java.util.List;

import org.springframework.data.mongodb.repository.MongoRepository;
import org.springframework.data.mongodb.repository.Query;

import ec.fin.logical.repositorio.models.PosicionPasivas;

public interface PosicionPasivasDao extends MongoRepository<PosicionPasivas, String>{
    @Query("{ 'fecha': { $gte: ?0, $lte: ?1 } }")
    public List<PosicionPasivas> buscarPorFecha(String fechaInicial, String fechaFinal) throws Exception;

    @Query("{ $and: [ { 'fecha': { $gte: ?0, $lte: ?1 } }, { 'tipo_institucion': ?2 } ] }")
    public List<PosicionPasivas> buscarPorFecha(String fechaInicial, String fechaFinal, String tipoInstitucion) throws Exception;
}
```

Figure 44: DAO Posición pasivas
*Fuente.* Elaboración propia

```java
package ec.fin.logical.repositorio.dao;

import java.util.List;

import org.springframework.data.mongodb.repository.MongoRepository;
import org.springframework.data.mongodb.repository.Query;

import ec.fin.logical.repositorio.models.PosicionActivas;

public interface PosicionActivasDao extends MongoRepository<PosicionActivas, String>{
    @Query("{ 'fecha': { $gte: ?0, $lte: ?1 } }")
    public List<PosicionActivas> buscarPorFecha(String fechaInicial, String fechaFinal) throws Exception;

    @Query("{ $and: [ { 'fecha': { $gte: ?0, $lte: ?1 } }, { 'tipo_institucion': ?2} ] }")
    public List<PosicionActivas> buscarPorFecha(String fechaInicial, String fechaFinal, String tipoInstitucion) throws Exception;
}
```

Figure 45: DAO Posición activas
*Fuente.* Elaboración propia

```java
package ec.fin.logical.repositorio.dao;

import java.util.List;

import org.springframework.data.mongodb.repository.MongoRepository;
import org.springframework.data.mongodb.repository.Query;

import ec.fin.logical.repositorio.models.VariacionPasivas;

public interface VariacionPasivasDao extends MongoRepository<VariacionPasivas, String>{
    @Query("{ 'fecha': ?0 }")
    public List<VariacionPasivas> buscarPorFecha(String fecha) throws Exception;
    @Query("{ $and: [ { 'fecha': ?0 }, { 'tipo_institucion': ?1 } ] }")
    public List<VariacionPasivas> buscarPorFecha(String fecha, String tipoInstitucion) throws Exception;
}
```

Figure 46: DAO Variación pasivas
*Fuente.* Elaboración propia

```java
package ec.fin.logical.repositorio.dao;

import java.util.List;

import org.springframework.data.mongodb.repository.MongoRepository;
import org.springframework.data.mongodb.repository.Query;

import ec.fin.logical.repositorio.models.VariacionActivas;

public interface VariacionActivasDao extends MongoRepository<VariacionActivas, String>{
    @Query("{ 'fecha': ?0 }")
    public List<VariacionActivas> buscarPorFecha(String fecha) throws Exception;
    @Query("{ $and: [ { 'fecha': ?0 }, { 'tipo_institucion': ?1 } ] }")
    public List<VariacionActivas> buscarPorFecha(String fecha, String tipoInstitucion) throws Exception;
}
```

Figure 47: DAO Variación activas
*Fuente.* Elaboración propia

**Micro servicio "administracion"**

Este permite llevar a cabo toda la configuración necesaria para el correcto funcionamiento del sistema, entre las distintas opciones que se tiene son:

- Autenticación de Usuarios: Se refiere a todo el proceso de verificar la existencia del usuario y validar sus credenciales.
- Creación de Usuarios: Se refiere al proceso de creación de usuarios en lo que respecta a los datos personales del usuario como es identificación, nombres, apellidos, correo electrónico y teléfono.
- Asignación de Permisos: Dentro del mismo proceso de creación de usuarios se tiene la asignación del rol y de la institución a la que pertenece.
- Creación de credenciales: Se refiere al proceso de creación del nombre de usuario y contraseña con el que ingresará al sistema.
- Listado de instituciones: Se refiere a la consulta de las instituciones financieras existentes en el sistema.
- Listado de personas: Se refiere al listado de todos los usuarios registrados en el sistema con el fin de que el administrador pueda visualizar sus datos personales.
- Listado de roles: Se refiere a la consulta y retorno de un listado de los roles existentes en el sistema pudiendo ser estos *admin*, *invitado*, *institucion (banco, cooperativa, mutualista)*.

Se sigue el mismo proceso de construcción de este micro servicio, se genera el controlador de consulta **REST**, y la interfaz de acceso a la base de datos **DAO**. Para revisar los

códigos fuentes generados por favor remitirse a la sección de anexos [Anexo V. Microservicio "administracion" 8]

### 5.3.4. Aplicación Web: Sistema de información gerencial

Esta sección trata sobre la aplicación web construida para permitir al usuario el consultar los datos de forma gráfica, aquella información que el robot extrajo en su momento, esta información visualizada fue tratada y mapeada de acuerdo a reglas predefinidas, el resultado de las 3 primeras fases conllevan al siguiente resultado, el cuál fue construido enteramente con Angular y D3 usándola a través de la librería *ngx-charts* que se integra fácilmente con Angular.

#### Autenticación de usuarios

Es la pantalla inicial del sistema aquí el usuario ingresará su username y password, el cuál si es correcto dará paso al dashboard de información.

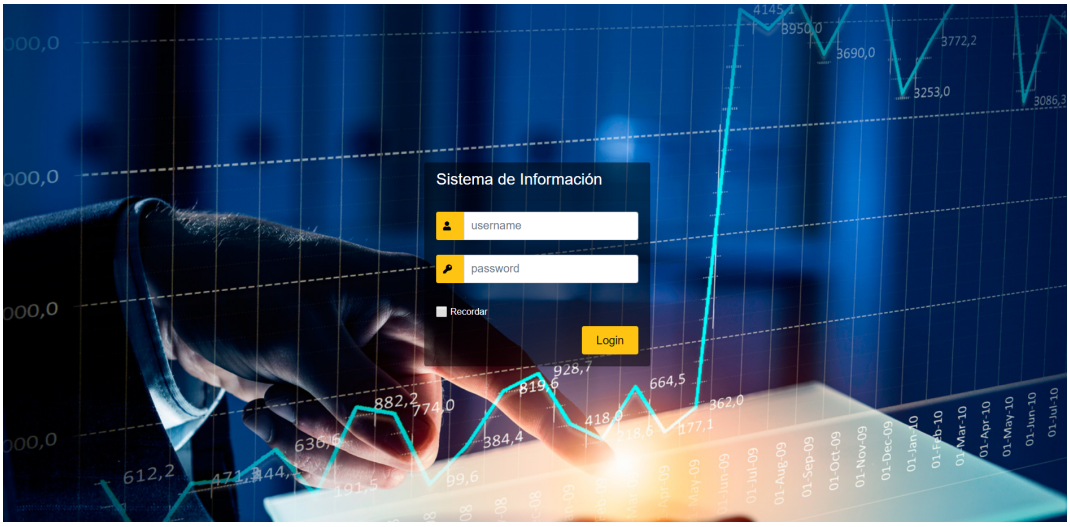



Figure 48: Pantalla de inicio de sesión
*Fuente.* Elaboración propia

Por definición de requerimientos, el usuario puede acceder a la aplicación si ha sido registrado por el administrador, caso contrario si no dispone de credenciales lo haría a través del usuario invitado creado. La pantalla valida que se ingrese el username o el password, caso contrario le presentará los respectivos mensajes de error. Si la autenticación es correcta se da paso al sistema.


Figure 49: Estructura interfaz sistema de información gerencial
*Fuente.* Elaboración propia

El sistema esta estructurado por una cabecera en donde se visualiza el botón de mostrar/ocultar el menú, el título del sistema de información gerencial, el botón de administración del sistema y el nombre del usuario que inicio sesión si pulsamos el nombre, se presenta un menú contextual desde donde podrá cerrar sesión. Vale aclarar que el botón de administración depende de los permisos del usuario.


Figure 50: Barra superior del sistema de información gerencial
*Fuente.* Elaboración propia

En la misma pantalla se presentan los distintos módulos que fueron asignados al usuario de acuerdo al rol otorgado, en este caso tenemos un menú que es asignado al usuario administrador o usuario institución.

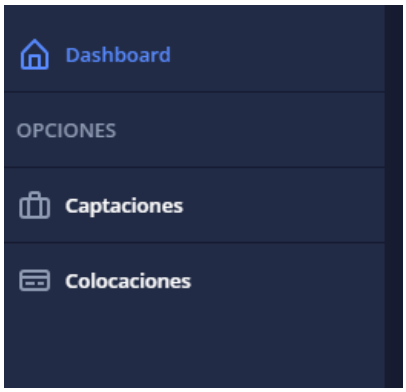



Figure 51: Menú del sistema de información gerencial
*Fuente.* Elaboración propia

### Módulo Dashboard


Figure 52: Dashboard principal
*Fuente.* Elaboración propia

Es la pantalla principal, se observa la tendencia desde 12 meses atrás hasta el mes actual, con esto se logra ver el comportamiento de las captaciones o colocaciones, el usuario puede interactuar por medio de los botones superiores del dashboard, adicional se presenta un resumen del saldo ya sea de las captaciones o colocaciones diferenciada por el tipo de institución. Si nos desplazamos en el dashboard tenemos 4 gráficas adicionales que permiten tener una visualización de como esta la estructura actual de saldos de los créditos y de las captaciones, diferenciadas por su producto, y también como ha sido las variaciones de los saldos con respecto al mes anterior, fin de año anterior y del mismo mes pero año anterior.


Figure 53: Dashboard principal continuación
*Fuente.* Elaboración propia

Continuando con la pantalla del dashboard principal en los gráficos de variaciones se tiene un combo que permite cambiar la variación de mensual a acumulada o anual tanto para colocaciones como para captaciones.

El dashboard principal si bien es el mismo para todos, varía en la presentación de los datos, y esto esta dado por los permisos y la institución que le fue asignada, en las imagenes siguiente se puede ver el caso de un usuario que pertenece al Banco del Austro y que solo tiene acceso a consultar instituciones de tipo Banco y del usuario invitado, solo tiene acceso al módulo Dashboard.

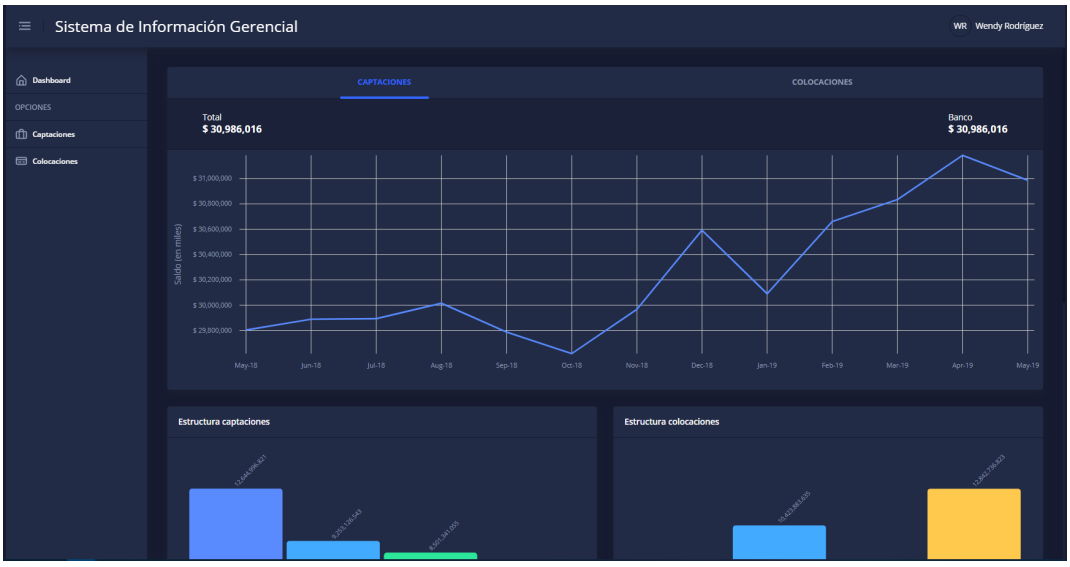



Figure 54: Dashboard usuario con permisos de institución banco
*Fuente.* Elaboración propia


Figure 55: Dashboard usuario invitado
*Fuente.* Elaboración propia

### Módulo Captaciones

Este módulo se centra únicamente a analizar los datos de las captaciones de las instituciones financieras a mas detalle, es decir visualizar la información ya por instituciones y como ha sido su comportamiento de forma individual.


Figure 56: Módulo captaciones
*Fuente.* Elaboración propia

El usuario dependiendo de sus permisos tiene acceso a los filtros de tipo de institución en caso de ser administrador, en su defecto solo podrá consultar los datos que le corresponden, resaltando a la institución que pertenece.


Figure 57: Módulo captaciones usuario banco
*Fuente.* Elaboración propia

De igual forma, el usuario puede analizar la estructura de cada institución con solo dar clic en la barra de aquella que desee consultar y todos los gráficos se actualizan como se puede ver en la siguiente imagen.


Figure 58: Selección de institución
*Fuente.* Elaboración propia

La información de bancos es de fácil visualización debido a la cantidad de instituciones que se tiene, lo cuál no resulta lo mismo para las cooperativas por lo que el gráfico en ese sentido se encuentra limitado a las primeras 22 instituciones que van de mayor a menor, para poder visualizar el resto se procede a habilitar una tabla en donde podrá consultar la misma información de los gráficos, y se estructura en tres secciones que son Bancos, Cooperativas y Mutualistas, estas pueden expandirse y poder tener una tabla mas detallada de todas las instituciones.

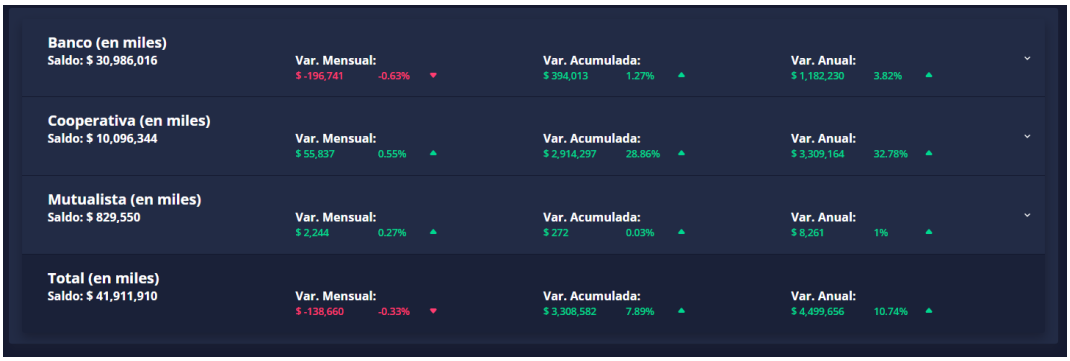



Figure 59: Tabla resumen captaciones
*Fuente.* Elaboración propia

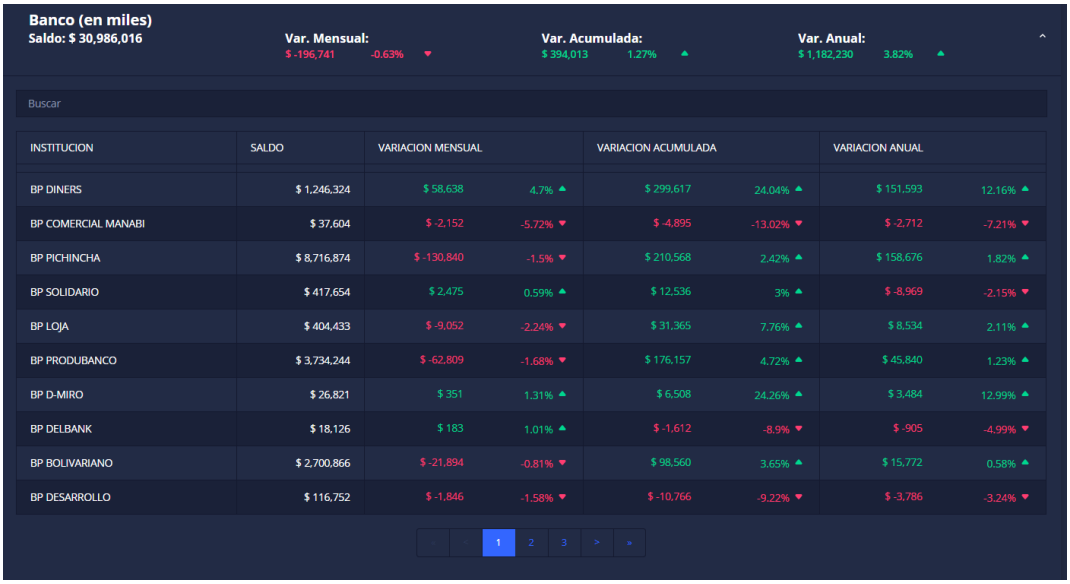



Figure 60: Tabla resumen captaciones detallada
*Fuente.* Elaboración propia

### Módulo Colocaciones

Para evitar confundir al usuario con una nueva forma de navegación se crea este módulo con la misma estructura y con las mismas interacciones, a continuación se puede ver la información del módulo.

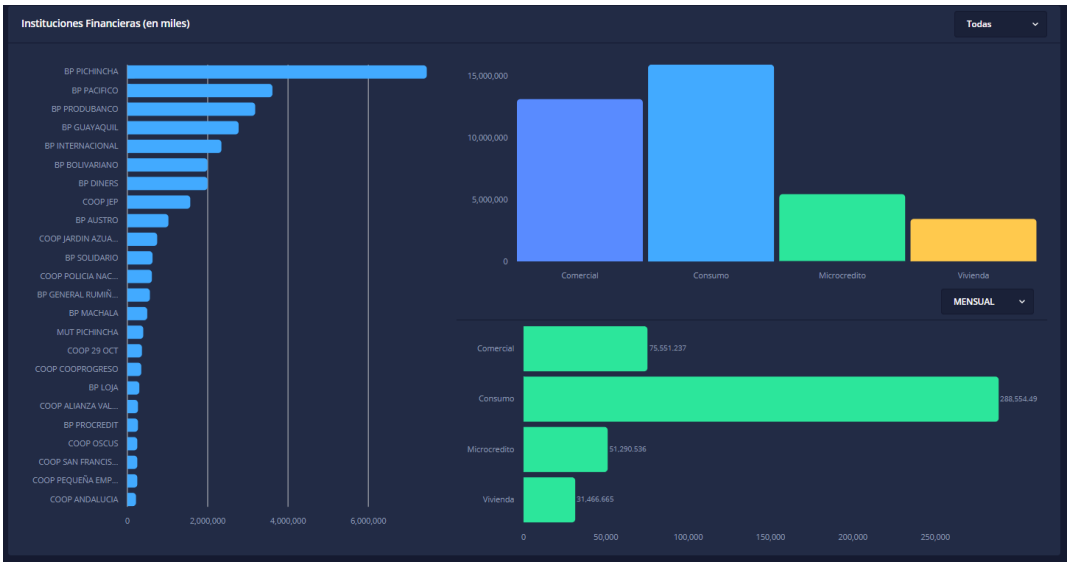



Figure 61: Módulo Colocaciones
*Fuente.* Elaboración propia


Figure 62: Tabla resumen colocaciones
*Fuente.* Elaboración propia

Se debe tener presente que lo mismo que ocurre en las captaciones con respecto a los permisos de los usuarios se mantiene en este módulo.

Para llegar a la construcción de este sistema se requirió investigar el desarrollo en Angular, me delimitaré a exponer los componentes del dashboard principal, del módulo de captaciones y colocaciones, en la presentación de los gráficos.

**Componente dashboard**

En angular se puede crear un componente conformado por una página web y archivo de SCSS, que son opcionales, ya que el mismo componente puede integrar la página HTML y los estilos, el dashboard esta compuesto por tres secciones, la primera que es el gráfico de tendencia con los datos de resúmenes, la segunda sección que son los dos gráficos de captaciones y colocaciones en donde se visualiza los saldos actuales por producto y la tercera sección que son los gráficos de variaciones, estas fueron construidas de forma independiente que luego son integradas en la página dashboard principal, por lo que existe una comunicación entres estos componentes en donde se transfieren los datos que son consultados por la API.

**Componente 1era Sección:** contiene un componente principal que carga el elemento web de "pestañas". Dentro de cada elemento pestaña se accede a otros dos componentes que son el resumen de saldos y la tabla de tendencia que se muestran a continuación:

```
<nb-card size="large">
  <nb-tabset fullWidth (changeTab)="changeTab($event)">
    <nb-tab tabTitle="Captaciones">
      <div class="chart-container">
        <sig-chart-panel-resumen [resumen]="resumenPasivas"></sig-chart-panel-resumen>
        <sig-chart-tendencia [datos]="seriePasivas" *ngIf="mostrarPasivas"></sig-chart-tendencia>
      </div>
    </nb-tab>
    <nb-tab tabTitle="Colocaciones" [lazyLoad]="true">
      <div class="chart-container">
        <sig-chart-panel-resumen [resumen]="resumenActivas"></sig-chart-panel-resumen>
        <sig-chart-tendencia [datos]="serieActivas" *ngIf="mostrarActivas"></sig-chart-tendencia>
      </div>
    </nb-tab>
  </nb-tabset>
</nb-card>
```

```
@Component({
  selector: 'sig-dashboard-panel',
  templateUrl: './chart-panel.component.html',
  styleUrls: ['./chart-panel.component.scss']
})
export class ChartPanelComponent implements OnInit {

  fecha = '2019-05-31';
  mostrarPasivas = false;
  mostrarActivas = false;
  tendenciaPasivas = 'tendencia-pasivas';
  tendenciaActivas = 'tendencia-activas';
  resumenPasivas: any[];
  resumenActivas: any[];
  seriePasivas: any[];
  serieActivas: any[];
  posicionActivas: PosicionActivas[];
  posicionPasivas: PosicionPasivas[];

  constructor(private pasivasService: PasivasService,
              private activasService: ActivasService) { }

  ngOnInit() {
    const tipoInstitucion = localStorage.getItem('tipoInstitucion');
    this.pasivasService.getPosicionPasivas(this.fecha, tipoInstitucion)
      .subscribe(res => {
        this.posicionPasivas = res;
        this.calcularResumenPasivas();
        this.calcularSeriesPasivas();
      }, error => {
        console.log('Error al llamar al servicio');
      });

    this.activasService.getPosicionActivas(this.fecha, tipoInstitucion)
      .subscribe(res => {
        this.posicionActivas = res;
        this.calcularResumenActivas();
        this.calcularSeriesActivas();
      }, error => {
        console.log('Error al llamar al servicio');
      });
  }

  /**
   * Funcion que calcula los resumenes de las captaciones
   */
  calcularResumenPasivas() {
    this.resumenPasivas = d3.nest()
      .key(dc.pluck('tipo_institucion'))
      .sortKeys(d3.ascending)
      .rollup((d): any => {
        return d3.sum(d, (v: any) => v.fecha === this.fecha ? v.saldo : 0);
      })
      .entries(this.posicionPasivas)
      .map((d): any => {
        return {
          titulo: d.key,
          valor: d.value
        };
      });

    const total = {
      titulo: 'Total',
      valor: d3.nest()
        .rollup((d): any => {
          return d3.sum(d, (v: any) => v.fecha === this.fecha ? v.saldo : 0);
        })
        .entries(this.posicionPasivas)
    };

    this.resumenPasivas.unshift(total);
  }

  calcularSeriesPasivas() {
    const datos = d3.nest()
      .key(dc.pluck('fecha'))
      .sortKeys(d3.ascending)
      .rollup((d): any => {
        return d3.sum(d, (v: any) => v.saldo / 1000);
      })
      .entries(this.posicionPasivas)
      .map((d): any => {
        return {
          name: d.key,
          value: d.value
        };
      });
```

```
  /**
   * Funcion que calcula los resumenes de las colocaciones
   */
  calcularResumenActivas() {
    this.resumenActivas = d3.nest()
      .key(dc.pluck('tipo_institucion'))
      .sortKeys(d3.ascending)
      .rollup((d): any => {
        return d3.sum(d, (v: any) => v.fecha === this.fecha ? v.saldo : 0);
      })
      .entries(this.posicionActivas)
      .map((d): any => {
        return {
          titulo: d.key,
          valor: d.value
        };
      });

    const total = {
      titulo: 'Total',
      valor: d3.nest()
        .rollup((d): any => {
          return d3.sum(d, (v: any) => v.fecha === this.fecha ? v.saldo : 0);
        })
        .entries(this.posicionActivas)
    };

    this.resumenActivas.unshift(total);
  }

  calcularSeriesActivas() {
    const datos = d3.nest()
      .key(dc.pluck('fecha'))
      .sortKeys(d3.ascending)
      .rollup((d): any => {
        return d3.sum(d, (v: any) => v.saldo / 1000);
      })
      .entries(this.posicionActivas)
      .map((d): any => {
        return {
          name: d.key,
          value: d.value
        };
      });
    this.serieActivas = [{
      name: 'Colocaciones',
      series: datos
    }];
  }

  changeTab(selectedTab: any) {
    if (selectedTab.tabTitle === 'Colocaciones') {
      this.mostrarActivas = true;
      this.mostrarPasivas = false;
    } else {
      this.mostrarPasivas = true;
      this.mostrarActivas = false;
    }
  }
}
```

Figure 63: Código de elemento web "Pestañas" 1era sección
*Fuente.* Elaboración propia

```
@Component({
  selector: 'sig-chart-tendencia',
  template: `
    <ngx-charts-line-chart
      [scheme]="colorScheme"
      [results]="datos"
      [xAxis]="showXAxis"
      [yAxis]="showYAxis"
      [legend]="showLegend"
      [showXAxisLabel]="showXAxisLabel"
      [showYAxisLabel]="showYAxisLabel"
      [xAxisLabel]="xAxisLabel"
      [yAxisLabel]="yAxisLabel"
      [autoScale]="autoScale"
      [xAxisTickFormatting]="this.dateTickFormatting"
      [yAxisTickFormatting]="this.nummberFormatting"
      [showGridLines]="showGridLines">
      <ng-template #tooltipTemplate let-model="model">
        <h5>{{ this.dateTickFormatting(model.name) }}: {{ this.nummberFormatting(model.value) }}</h5>
      </ng-template>
    </ngx-charts-line-chart>
  `,
})
export class ChartTendenciaComponent implements OnDestroy {
  @Input() datos: any;

  showLegend = false;
  showXAxis = true;
  showYAxis = true;
  showXAxisLabel = false;
  xAxisLabel = 'Fecha';
  showYAxisLabel = true;
  yAxisLabel = 'Saldo (en miles)';
  colorScheme: any;
  themeSubscription: any;
  autoScale = true;
  showGridLines = true;

  constructor(private theme: NbThemeService) {
    this.themeSubscription = this.theme.getJsTheme().subscribe(config => {
      const colors: any = config.variables;
      this.colorScheme = {
        domain: [colors.primaryLight, colors.infoLight, colors.successLight, colors.warningLight, colors.dangerLight],
      };
    });
  }
```

Figure 64: Código de gráfico de tendencia 1era sección
*Fuente.* Elaboración propia

```
@Component({
  selector: 'sig-chart-panel-resumen',
  styleUrls: ['./chart-panel-resumen.component.scss'],
  template: `
    <div class="summary-container">
      <div *ngFor="let item of resumen">
        <div>{{ item.titulo }}</div>
        <div class="h6">{{ item.valor | sigNumeroConComas }}</div>
      </div>
    </div>
  `
})
export class ChartPanelResumenComponent {
  @Input() resumen: { titulo: string; valor: number }[];
}
```

Figure 65: Código de resumen de saldos 1era sección
*Fuente.* Elaboración propia

**Componente 2da Sección:** esta sección sigue el mismo enfoque, se tiene un componente principal que carga los dos paneles donde se presentan los gráficos, y dentro de cada

panel se incluye el componente que contiene el gráfico, se usa el mismo componente de gráfico pero se le envía distintos datos tanto para captaciones como colocaciones. A continuación el código empleado:

```
export class EstructuraPanelComponent implements OnInit {

  fecha = '2019-05-31';
  estructuraPasivas: any[];
  estructuraActivas: any[];
  posicionActivas: PosicionActivas[];
  posicionPasivas: PosicionPasivas[];

  constructor(private pasivasService: PasivasService,
              private activasService: ActivasService) { }

  ngOnInit() {
    const tipoInstitucion = localStorage.getItem('tipoInstitucion');
    this.pasivasService.getPosicionPasivas(this.fecha, tipoInstitucion)
      .subscribe(res => {
        this.posicionPasivas = res;
        this.calcularEstructuraPasivas();
      }, error => {
        console.log('Error al llamar al servicio');
      });

    this.activasService.getPosicionActivas(this.fecha, tipoInstitucion)
      .subscribe(res => {
        this.posicionActivas = res;
        this.calcularEstructuraActivas();
      }, error => {
        console.log('Error al llamar al servicio');
      });
  }

  calcularEstructuraPasivas() {
    this.estructuraPasivas = d3.nest()
      .key(dc.pluck('producto'))
      .rollup((d): any => {
        return d3.sum(d, (v: any) => v.fecha === this.fecha ? v.saldo / 1000 : 0);
      }).entries(this.posicionPasivas)
      .map((d): any => {
        return {
          name: d.key,
          value: d.value
        };
      });
  }

  calcularEstructuraActivas() {
    this.estructuraActivas = d3.nest()
      .key(dc.pluck('producto'))
      .rollup((d): any => {
        return d3.sum(d, (v: any) => v.fecha === this.fecha ? v.saldo / 1000 : 0);
      }).entries(this.posicionActivas)
      .map((d): any => {
        return {
          name: d.key,
          value: d.value
        };
      });
  }
}
```

Figure 66: Código de componente de paneles 2da sección
*Fuente.* Elaboración propia

```
<div class="row">
  <div class="col-md-6">
    <nb-card>
      <nb-card-header>
        <span>Estructura captaciones</span>
      </nb-card-header>
      <nb-card-body>
        <sig-chart-estructura [datos]="estructuraPasivas"></sig-chart-estructura>
      </nb-card-body>
    </nb-card>
  </div>
  <div class="col-md-6">
    <nb-card>
      <nb-card-header>
        <span>Estructura colocaciones</span>
      </nb-card-header>
      <nb-card-body>
        <sig-chart-estructura [datos]="estructuraActivas"></sig-chart-estructura>
      </nb-card-body>
    </nb-card>
  </div>
</div>
```

Figure 67: Código de componente de paneles página web 2da sección
*Fuente.* Elaboración propia

```
@Component({
  selector: 'sig-chart-estructura',
  template: `
    <ngx-charts-bar-vertical
      [scheme]="colorScheme"
      [results]="datos"
      [xAxis]="showXAxis"
      [yAxis]="showYAxis"
      [legend]="showLegend"
      [xAxisLabel]="xAxisLabel"
      [yAxisLabel]="yAxisLabel"
      [showDataLabel]="showDataLabel"
      [showGridLines]="showGridLines">
    </ngx-charts-bar-vertical>
  `
})
export class ChartEstructuraComponent implements OnDestroy {

  @Input() datos: any;

  colorScheme: any;
  themeSubscription: any;
  showLegend = false;
  showXAxis = true;
  showYAxis = false;
  showDataLabel = true;
  showGridLines = false;
  xAxisLabel = 'Producto';
  yAxisLabel = 'Saldo (en miles)';

  constructor(private theme: NbThemeService) {
    this.themeSubscription = this.theme.getJsTheme().subscribe(config => {
      const colors: any = config.variables;
      this.colorScheme = {
        domain: [colors.primaryLight, colors.infoLight, colors.successLight, colors.warningLight, colors.dangerLight],
      };
    });
  }

  ngOnDestroy() {
    this.themeSubscription.unsubscribe();
  }

}
```

Figure 68: Código de componente de gráfico 2da sección
*Fuente.* Elaboración propia

**Componente 3ra Sección:** mantiene la misma estructura de trabajo que la 2da sección, carga los dos paneles donde se presentan los gráficos, y dentro de cada panel se incluye el componente que contiene el gráfico de variaciones, aquí se agregan dos combos adicionales que permiten consultar variaciones mensuales, acumuladas y anuales, se usa el mismo com-

ponente de gráfico pero se le envía distintos datos tanto para captaciones como colocaciones.

A continuación el código empleado:

```
<div class="row">
  <div class="col-md-6">
    <nb-card class="chart-card">
      <nb-card-header>
        <span class="stats">Variaciones Captaciones</span>
        <nb-select class="type-select" [selected]="tipoVariacionPasivas" (selectedChange)="changeVariacionPasivas($event)">
          <nb-option *ngFor="let variacion of tiposVariaciones" [value]="variacion">
            {{ variacion }}
          </nb-option>
        </nb-select>
      </nb-card-header>
      <nb-card-body>
        <sig-chart-variacion [datos]="barrasPasivas"></sig-chart-variacion>
      </nb-card-body>
    </nb-card>
  </div>
  <div class="col-md-6">
    <nb-card class="chart-card">
      <nb-card-header>
        <span class="stats">Variaciones Colocaciones</span>
        <nb-select class="type-select" [selected]="tipoVariacionActivas" (selectedChange)="changeVariacionActivas($event)">
          <nb-option *ngFor="let variacion of tiposVariaciones" [value]="variacion">
            {{ variacion }}
          </nb-option>
        </nb-select>
      </nb-card-header>
      <nb-card-body>
        <sig-chart-variacion [datos]="barrasActivas"></sig-chart-variacion>
      </nb-card-body>
    </nb-card>
  </div>
</div>
```

Figure 69: Código de componente de paneles página web 3ra sección
*Fuente.* Elaboración propia

```
@Component({
  selector: 'sig-chart-variacion',
  template: `
    <ngx-charts-bar-horizontal
      [scheme]="colorScheme"
      [animations]="false"
      [results]="datos"
      [xAxis]="showXAxis"
      [yAxis]="showYAxis"
      [legend]="showLegend"
      [xAxisLabel]="xAxisLabel"
      [yAxisLabel]="yAxisLabel"
      [showDataLabel]="showDataLabel"
      [showGridLines]="showGridLines">
    </ngx-charts-bar-horizontal>
  `,
})
export class ChartVariacionComponent implements OnChanges, OnDestroy {
  @Input() datos: any;

  colorScheme: any;
  themeSubscription: any;
  showLegend = false;
  showXAxis = true;
  showYAxis = true;
  showDataLabel = true;
  showGridLines = false;
  xAxisLabel = 'Country';
  yAxisLabel = 'Population';

  constructor(private theme: NbThemeService) {
    this.themeSubscription = this.theme.getJsTheme().subscribe(config => {
      const colors: any = config.variables;
      this.colorScheme = {
        domain: [colors.primaryLight, colors.infoLight, colors.successLight, colors.warningLight, colors.dangerLight],
      };
    });
  }

  ngOnChanges() {
    if (!this.datos) {
      return;
    }

    this.colorScheme = {
      domain: this.datos.map((d: any) => d.color),
    };
```

Figure 70: Código de componente de gráfico 3ra sección
*Fuente.* Elaboración propia

```
export class VariacionPanelComponent implements OnInit {

  tipoVariacionPasivas = 'MENSUAL';
  tipoVariacionActivas = 'MENSUAL';
  tiposVariaciones: string[] = ['MENSUAL', 'ACUMULADA', 'ANUAL'];

  fecha = '2019-05-31';
  barrasPasivas: any[];
  barrasActivas: any[];
  variacionActivas: VariacionActivas[];
  variacionPasivas: VariacionPasivas[];

  constructor(private pasivasService: PasivasService,
              private activasService: ActivasService) { }

  ngOnInit() {
    const tipoInstitucion = localStorage.getItem('tipoInstitucion');
    this.pasivasService.getVariacionPasivas(this.fecha, tipoInstitucion)
      .subscribe(res => {
        this.variacionPasivas = res;
        this.calcularVariacionPasivas(this.tipoVariacionPasivas);
      }, error => {
        console.log('Error al llamar al servicio');
      });

    this.activasService.getVariacionActivas(this.fecha, tipoInstitucion)
      .subscribe(res => {
        this.variacionActivas = res;
        this.calcularVariacionActivas(this.tipoVariacionActivas);
      }, error => {
        console.log('Error al llamar al servicio');
      });
  }

  calcularVariacionPasivas(tipo: string) {
    this.barrasPasivas = d3.nest()
      .key(dc.pluck('producto'))
      .rollup((d): any => {
        return d3.sum(d, (v: any): any => (v.valor_final - v.valor_inicial) / 1000);
      }).entries(this.variacionPasivas.filter(d => d.tipo === tipo))
      .map((d: any): any => {
        return {
          name: d.key,
          value: d.value,
          color: (d.value >= 0) ? '#2ce69b' : '#ff708d'
        };
      });
  }

  calcularVariacionActivas(tipo: string) {
    this.barrasActivas = d3.nest()
      .key(dc.pluck('producto'))
      .rollup((d): any => {
        return d3.sum(d, (v: any): any => (v.valor_final - v.valor_inicial) / 1000);
      }).entries(this.variacionActivas.filter(d => d.tipo === tipo))
      .map((d: any): any => {
        return {
          name: d.key,
          value: d.value,
          color: (d.value >= 0) ? '#2ce69b' : '#ff708d'
        };
      });
  }

  changeVariacionPasivas(tipoVariacion: string): void {
    this.tipoVariacionPasivas = tipoVariacion;
    this.calcularVariacionPasivas(tipoVariacion);
  }

  changeVariacionActivas(tipoVariacion: string): void {
    this.tipoVariacionActivas = tipoVariacion;
    this.calcularVariacionActivas(tipoVariacion);
  }
}
```

Figure 71: Código de componente de paneles 3ra sección
*Fuente.* Elaboración propia

Estos elementos de código construidos definen la guía que se siguió para la construcción del resto de módulos como son las captaciones y colocaciones, cada uno mantiene su propio paquete de código generado.

Es importante mostrar como es el proceso de llamado de los datos a las API's construidas, para ello se definieron unas interfaces que luego son implementadas en servicios angular, cada uno de estos son importados posteriormente dentro de cada componente en donde se desee obtener los datos, a continuación se muestra el servicio que obtiene los datos de las captaciones:

```typescript
import { Observable } from 'rxjs';

export interface PosicionPasivas {
  _id: any;
  fecha: string;
  tipo_institucion: string;
  nombre_corto: string;
  nombre_largo: string;
  producto: string;
  saldo: number;
}

export interface VariacionPasivas {
  _id: any;
  fecha: string;
  tipo_institucion: string;
  nombre_corto: string;
  nombre_largo: string;
  producto: string;
  valor_inicial: number;
  valor_final: number;
  tipo: string;
}

export abstract class PasivasData {
  abstract getPosicionPasivas(fecha: string, tipoInstitucion: string): Observable<PosicionPasivas[]>;
  abstract getVariacionPasivas(fecha: string, tipoInstitucion: string): Observable<VariacionPasivas[]>;
}
```

Figure 72: Código de interfaz de las captaciones
*Fuente.* Elaboración propia

```typescript
import { VariacionPasivas } from './../data/pasivas';
import { Constantes } from './../shared/constantes';
import { Injectable } from '@angular/core';
import { PasivasData, PosicionPasivas } from '../data/pasivas';
import { CustomHttp } from '../shared/customHttp';
import { Observable } from 'rxjs';
import { map, catchError } from 'rxjs/operators';

@Injectable()
export class PasivasService extends PasivasData {

  private baseUrl = Constantes.API_PASIVAS;

  constructor(private http: CustomHttp) {
    super();
  }

  getPosicionPasivas(fecha: string, tipoInstitucion: string): Observable<PosicionPasivas[]> {
    const url = `${this.baseUrl}/posicion/buscar/fecha/${fecha}/${tipoInstitucion}`;
    return this.http.get(url).pipe(
      map((response: any) => {
        return response as PosicionPasivas;
      }), catchError(this.handleError));
  }

  getVariacionPasivas(fecha: string, tipoInstitucion: string): Observable<VariacionPasivas[]> {
    const url = `${this.baseUrl}/variacion/buscar/fecha/${fecha}/${tipoInstitucion}`;
    return this.http.get(url).pipe(
      map((response: any) => {
        return response as VariacionPasivas;
      }), catchError(this.handleError));
  }

  private handleError(error: any): Promise<any> {
    console.error('Ha ocurrido un error.', error); // para propositos de desarrollo
    return Promise.reject(error.message || error);
  }
}
```

Figure 73: Código de servicio de las captaciones
*Fuente.* Elaboración propia

Vemos en la imagen 73 que se hacen las peticiones *GET* que se definen en la API, haciendo uso de un paquete de angular denominado **http**, de esta forma es como se realizan las peticiones de datos al servidor, esta data es mapeada al objeto *PosicionPasivas* o *VariacionPasivas* que se muestran en la imagen 72. Se sigue el mismo proceso para el resto de métodos http que se tienen para cada operación del sistema como son las colocaciones y la administración del SIG.

## 5.4. Transición

Esta sección se refiere a verificar en base a pruebas realizadas al producto final sobre si se cumplió o no los requerimientos del cliente, esto se lo realiza analizando cada uno de los requerimientos funcionales con los componentes desarrollados y detallados en la sección de construcción, en la siguiente tabla se visualiza este análisis:

| Requerimiento | Procedimiento | Resultado |
|---|---|---|

| | | |
|---|---|---|
| **RF001. Extracción balances generales** | Este proceso se lo realiza con los proceso batch que fueron construidos en la herramienta R y parametrizados en la base de datos, estos son configurados únicamente por el administrador de sistema.<br>1. Parametrizar el scheduler del sistema operativo para indicarle la hora del día en que se ejecutará el proceso.<br>2. Verificar que los procesos se encuentren subidos al servidor que realizará la ejecución diaria de procesos.<br>3. El proceso automáticamente reprocesara la información en caso de fallo hasta un número máximo de días<br>4. En caso de fallo, verificar el estado del proceso en la base de datos correspondiente al perído de ejecución y si se encuentra con estado fallido se procede a ejecutar manualmente el proceso, si aún así persiste el problema quiere indicar que no existe los datos para la extracción | El proceso batch realiza correctamente la extracción de información, por lo general la extracción es mensual, se ve que al inicio de cada mes extrae datos, pero siempre las fuentes demoran en cargarlos por lo que entran al batch de reproceso que mantiene consultando diariamente hasta obtener los datos. |
| **RF002. Administración de procesos automáticos** | Este proceso se lo realiza en las bases de datos de los procesos batch, por lo que existe el mantenimiento de los procesos automáticos.<br>1. Acceder a la base de datos procesos_batch<br>2. Acceder a la colección origen en donde se puede parametrizar el nombre del proceso, la frecuencia de ejecución, el numero de reintentos, la url de extracción de los datos, el nombre del documento en caso de existir, el formato del documento, la hoja del documento en caso de tratarse de un archivo de excel, esta parametrización es variable y depende del origen, por esa razón se lo implementa en mongo por la flexibilidad que ofrece en los datos. | La gesión de la base de datos corre por parte del administrador de la base, pero la configuración de los procesos esta a cargo del administrador del sistema, por lo que el usuario debe tener conocimientos sobre la gestión de una base mongo, en este caso se da por cumplido este requierimiento. |

| | | |
|---|---|---|
| **RF003. Administración cluster de procesamiento** | Esta administración corre a cargo del administrador del clúster, por lo que tiene que conocer los comandos de sistema operativo linux y de archivos xml, ya que la administración se la realiza en este tipo de documentos.<br>1. Accede al servidor master donde esta ejecutandose Apache Spark<br>2. Se redirije al directorio en donde esta almacenado Apache Spark, por lo general esta en /opt/spark<br>3. Realizar las configuraciónes que vea necesarias.<br>4. Detener el servicio de Apache Spark e inicializarlo nuevamente.<br>5. Verificar que se este ejecutando correctamente apuntando a la dirección http://master-spark:8080<br>6. Para ejecutar una aplicación spark el administrador debe ejecutar el comando spark-submit con los parametros que se especifiquen en la aplicación. | Se creó un clúster spark, con dos servidores un máster y un esclavo, el administrador puede seguir añadiendo mas nodos si lo ve necesario, además puede ejecutar diferentes aplicaciones spark para llevar a cabo procesos de datos masivos, por lo que se cumple este proceso. |

| | | |
|---|---|---|
| **RF004. Sistema basado en roles** | Se creó una aplicación web capaz de permitir tres tipos de acceso de usuarios.<br>Usuario Administrador.<br>1. Ingresa las credenciales de usuario<br>2. Accede a toda las información de todos las instituciones y a todos los módulos<br>3. Puede verificar los saldos de captaciones y colocaciones en un período de un año.<br>4. Puede verificar las variaciones actuales con respecto al mes anterior, año anterior.<br>5. Puede acceder a la interfaz de administración de usuarios.<br>Usuario Institución<br>1. Ingresa las credenciales del usuario.<br>2. Accede a consultar las instituciones a la cuál pertenece pudiendo ser Banco, Cooperativa o Mutualista<br>3. Tiene acceso a todos los módulos pero no al módulo de administración.<br>Usuario Invitado<br>1. Ingresa con las credenciales de invitado.<br>2. Tiene acceso a la información de todas las instituciones pero no tiene acceso a todos los módulos solo al dashboard principal. | El sistema realiza las validaciones correspondientes para que no se le presente las opciones a las cuales no tiene acceso un usuario, de esta manera se consigue el sistema basado en roles, cumpliendo con el requerimiento funcional. |
| **RF005. Visualización de dashboard** | En realidad el sistema tiene 3 dashboard que estan atádos a cada módulo, dependiendo del rol de usuario podrá acceder a todos o solamente a uno, y de igual forma visualizar todos las instituciones o solo la que le corresponda.<br>1. Ingresar al sistema por medio de sus credenciales.<br>2. Por default, el sistema le redirigue al dashboar principal que contiene un resumen de las captaciones y colocaciones. | El sistema contiene 3 módulos, en donde el principal muestra un resumen de las captaciones y colocaciones del sistema, por lo que se cumple con el requerimiento de permitir un dashboard general. |

| | | |
|---|---|---|
| **RF006. Consulta de captaciones** | 1. Ingresar con las credenciales de usuario<br>2. Si tiene los permisos correspondientes, en el menu izquierdo podrá visualizar la opción de Captaciones<br>3. Al dar clic, le redirije a la nueva pantalla, en donde se tiene datos mas detallados por institución y su estructura y variación, así como una tabla de los datos que no se pueden visualizar en los gráficos. | El sistema incluye un módulo que permite la visualización de las captaciones, en donde se tiene un gráfico de barras horizontal de todas las instituciones, si tiene acceso a todas puede filtrar y consultar solo bancos, cooperativas o mutualistas, tambien un resumen en forma tabular de los saldos y sus variaciones, ya sea del tipo de institucion a la cuál tiene acceso. Se cumple el requerimiento. |
| **RF007. Consulta de colocaciones** | 1. Ingresar con las credenciales de usuario<br>2. Si tiene los permisos correspondientes, en el menu izquierdo podrá visualizar la opción de Colocaciones<br>3. Al dar clic, le redirije a la nueva pantalla, en donde se tiene datos mas detallados por institución y su estructura y variación, así como una tabla de los datos que no se pueden visualizar en los gráficos. | El sistema incluye un módulo que permite la visualización de las colocaciones, en donde se tiene un gráfico de barras horizontal de todas las instituciones, si tiene acceso a todas puede filtrar y consultar solo bancos, cooperativas o mutualistas, tambien un resumen en forma tabular de los saldos y sus variaciones, ya sea del tipo de institucion a la cuál tiene acceso. Se cumple el requerimiento. |
| **RF008. Consulta de variaciones** | El procedimiento que se sigue es acceder a cualquier módulo del sistema dependiendo del permiso del usuario, y existe un combo en la parte superior derecha que facilita el análisis de las variaciones mensuales, acumuladas y anuales. | Para cumplir con este requerimiento en cada módulo se agrego una gráfica que brinda la posibilidad al usuario de poder comparar los productos o las instituciones ya sea en captaciones o colocaciones con datos del mes anterior, año anterior y fin de año anterior. |

| | | |
|---|---|---|
| **RF009. Comparación entre instituciones** | La comparación se la puede realizar con el módulo de captaciones y colocaciones.<br>1. Ingresar al sistema con las credenciales otorgadas.<br>2. Acceder al módulo de captaciones o colocaciones.<br>3. En el primer gráfico se tiene el ranking de las agencias y se marca la institución a la cuál pertenece.<br>4. De esta manera, el usuario puede comparar sus saldos con las otras instituciones y visualizar su posición dentro del mercado financiero ecuatoriano. | El sistema ofrece la posibilidad de listar todas las instituciones y marcar la institución a la que pertenece el usuario de esta manera pueda compararse con otras instituciones y ver su posicion dentro del mercado financiero. |
| **RF010. Registro de usuarios** | | Este requerimiento se descarto puesto que el usuario final indicó que los usuarios no pueden registrarse ellos directamente, sino que lo realiza el administrador del sistema, el cuál se cumple en el RF011 |
| **RF011. Administración de usuarios** | Usuario Administrador.<br>1. Ingresar las credenciales del usuario administrador.<br>2. En la cabecera del aplicativo junto al nombre de usuario se muestra un icono que permite ingresar a la administración de usuarios.<br>3. Puede visualizar el listado de usuarios presentes en el sistema, y puede agregar nuevos usuarios asignandoles los permisos respectivos. | El sistema incluye un módulo que no se visualiza en el menú del sistema sino que se le incluye como una opción dentro de la cabecera como que indicando que existe una interfaz de configuración del aplicativo. El sistema cumple con la administración de usuarios. |
| **RF012. Filtros cruzados** | Usuario Administrador y Usuario Institución.<br>1. Ingresar al sistema con las credenciales otorgadas.<br>2. Ingresar al módulo de captaciones o colocaciones.<br>3. En el gráfico donde se lista las instituciones financieras se puede seleccionar una barra, e inmediatamente se renderiza el resto de gráficas para poder analizar la variación y la estructura de las captaciones o colocaciones | El sistema ofrece una interacción directa en el gráfico afectando el filtro a todas las demás gráficas cumpliendo de esta manera un filtro cruzado. |

| | | |
|---|---|---|
| **RF013. Administración datos de balances.** | Usuario Administrador<br>1. El usuario ingresa a la base de datos repositorio<br>2. Consulta la colección balances<br>3. Puede realizar updates a los datos que vea necesario cambiar<br>4. Puede subir archivos csv o json en caso de que necesite cargar nuevos datos de forma manual | La administración de los balances se los realiza directamente en la base de datos, no esta especificado que este tratamiento se lo haga en la aplicación ya que al tratarse de mucha información no resulta adecuado. |
| **RF014. Administración base de datos** | Usuario Administrador Base de Datos<br>1. Acceder con las credenciales de administrador de base de datos.<br><br>En caso de creación de bases de datos, usar un gestor gráfico o por medio de comandos mongo.<br>1. Clic derecho en servidor mongo<br>2. Seleccionar la opción de crear base de datos<br><br>En caso de creación de colecciones, usar un gestor gráfico o por medio de comandos mongo.<br>1. Clic derecho sobre la base de datos en donde se creará la colección.<br>2. Ingresar el nombre de la colección.<br><br>En caso de actividades ingreso, modificación y eliminación.<br>1. Crear una nueva ventana de script mongo.<br>2. Ingresar las sentencias para la insercción, modificación y eliminación. | La administración de la base de datos es una tarea completamente ajeno a la aplicación web, pero es funcional porque este permitira incrementar las capacidades de seguridad y almacenamiento del servidor a medida que aumente los datos, esto se cumple ya que se han creado las respectivas credenciales de administrador de aplicación y administrador de clúster mongo. |

Table 10: Transición Requerimientos Funcionales. *Fuente:* Elaboración Propia

# 6. Evaluación

Para el proyecto desarrollado se realizaron una serie de preguntas que se entregaron a los usuarios que probaron el sistema y en base a su criterio responderlas. Estas preguntas tienen un rango de respuestas que van desde el valor 1 que significa que el usuario no esta de acuerdo al 5 que siginifica que esta completamente de acuerdo, de esta manera se podrá evaluar el software.

La creación del cuestionario se baso en la plantilla creada por la *Digital Equipment Corportation* que creó un cuestionario denominado SUS *(System Usability Scale)* el cuál se centra en la evaluación de la facilidad y funcionalidad sobre cualquier sistema.

El cuestionario contiene 11 preguntas que se encuentran detalladas en la sección de anexos (Calvo-Fernández Rodríguez, Ortega Santamaría, & Valls Saenz, 2004), distribuido en tres secciones, permitiendo a través de la escala de Likert determinar la facilidad (preguntas del 1 al 5) y la funcionalidad (preguntas del 6 al 10), la pregunta 11 ayudará al administrador de la empresa Logical Consulting a determinar si continuará para trabajos futuros potenciando el proyecto realizado en el TFM.

## 6.1. Aplicación del cuestionario

El cuestionario se aplico a un número de 10 usuarios que fueron seleccionados de manera aleatoria por parte del administrador de la empresa Logical Consulting puesto que son ellos los que usarán la demo para determinar si es necesario una actualización del software. Basándose en los distintos roles que tienen los usuarios, se llegaron a las siguientes puntuaciones.

| Persona | Puntuación |
|---|---|
| Persona 1 | **4.8** |
| Persona 2 | **4.4** |
| Persona 3 | **4.6** |
| Persona 4 | **5.0** |
| Persona 5 | **4.6** |
| Persona 6 | **5.0** |
| Persona 7 | **5.0** |
| Persona 8 | **5.0** |
| Persona 9 | **4.2** |
| Persona 10 | **4.2** |
| **Media** | **4.7** |
| **Mediana** | **4.7** |
| **Moda** | **5.0** |
| **Desviación Estándar** | **0.3** |

Table 11: Tabla de puntuación de la facilidad de uso del SIG. *Fuente.* Elaboración Propia.

| Persona | Puntuación |
|---|---|
| Persona 1 | **4.4** |
| Persona 2 | **4.6** |
| Persona 3 | **4.6** |
| Persona 4 | **5.0** |
| Persona 5 | **4.8** |
| Persona 6 | **5.0** |
| Persona 7 | **5.0** |
| Persona 8 | **5.0** |
| Persona 9 | **5.0** |
| Persona 10 | **4.8** |
| **Media** | **4.8** |
| **Mediana** | **4.9** |
| **Moda** | **5.0** |
| **Desviación Estándar** | **0.2** |

Table 12: Tabla de puntuación de la funcionalidad del SIG. *Fuente.* Elaboración Propia.

## 6.2. Análisis de los resultados

El análisis de los resultados se basan en las dos secciones mencionadas con anterioridad de estas se tomo 4 preguntas relevantes 2 de la facilidad y 2 de la funcionalidad para evaluar a partir de la desviación estándar que tan dispersos se encuentran los usuarios en relación al promedio. (Calvo-Fernández Rodríguez et al., 2004)

### 6.2.1. Puntuaciones a las preguntas claves

Para la facilidad se evaluó las siguientes preguntas clave:

En la **pregunta 2**, se observa en la siguiente gráfica claramente que el 50% piensan que no necesitan tener conocimientos previos para hacer uso del sistema de información gerencial, sin embargo existe un 50% de personas que consideran que es necesario tener un conocimiento previo para el uso del sistema.

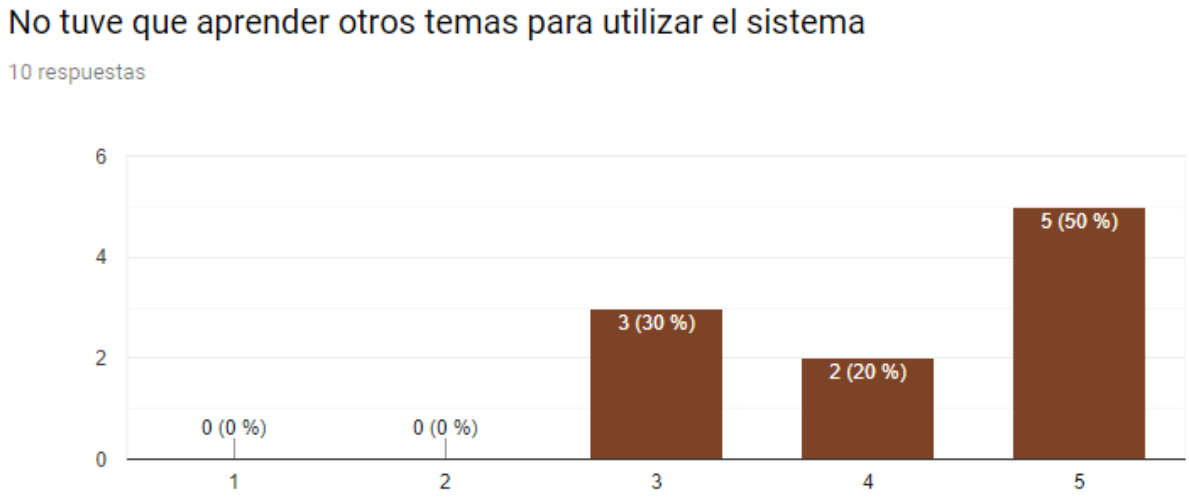



Figure 74: Gráfico de evaluación de la 2da pregunta
*Fuente.* Elaboración propia

Otras de las preguntas que a mi parecer necesitan revisarse es la **pregunta 4**, de igual forma si analizamos la gráfica se observa que el 60% de usuarios indican que no necesitan apoyo de algún soporte para usar el sistema, un 30% piensa que necesita en raras ocasiones apoyo, y el 10% de usuarios considera que frecuentemente necesitará de soporte para manipular el sistema.

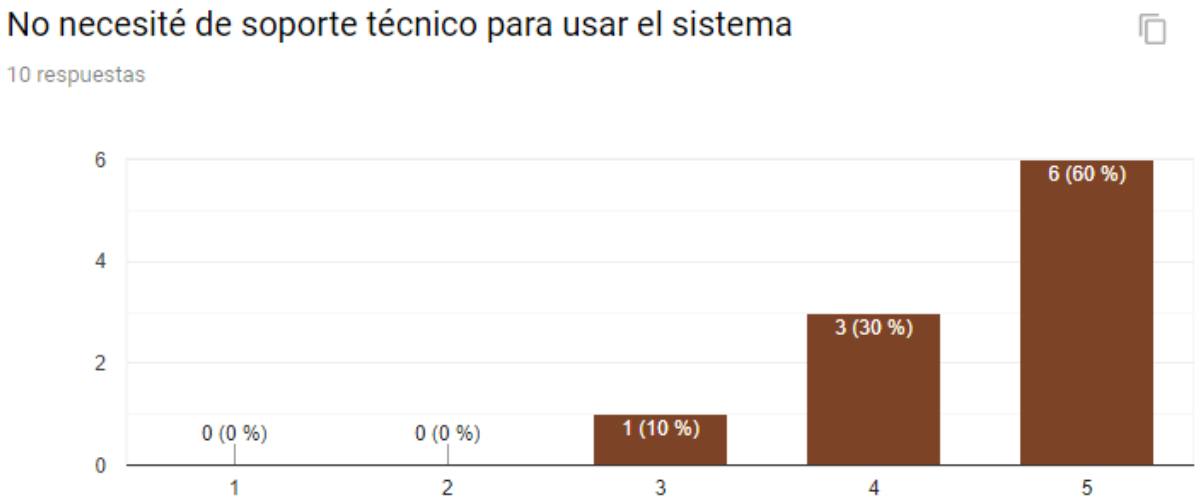



Figure 75: Gráfico de evaluación de la 4ta pregunta
*Fuente.* Elaboración propia

Para la funcionalidad se evaluó las siguientes preguntas clave:

Se tomo en consideración la **pregunta 8** en donde se visualiza en la gráfica siguiente

que el 80% de usuarios consideran las funcionalidades del sistema correctamente integradas, en comparación de un 20% que considera que falta realizar algunos ajustes en las opciones presentes.

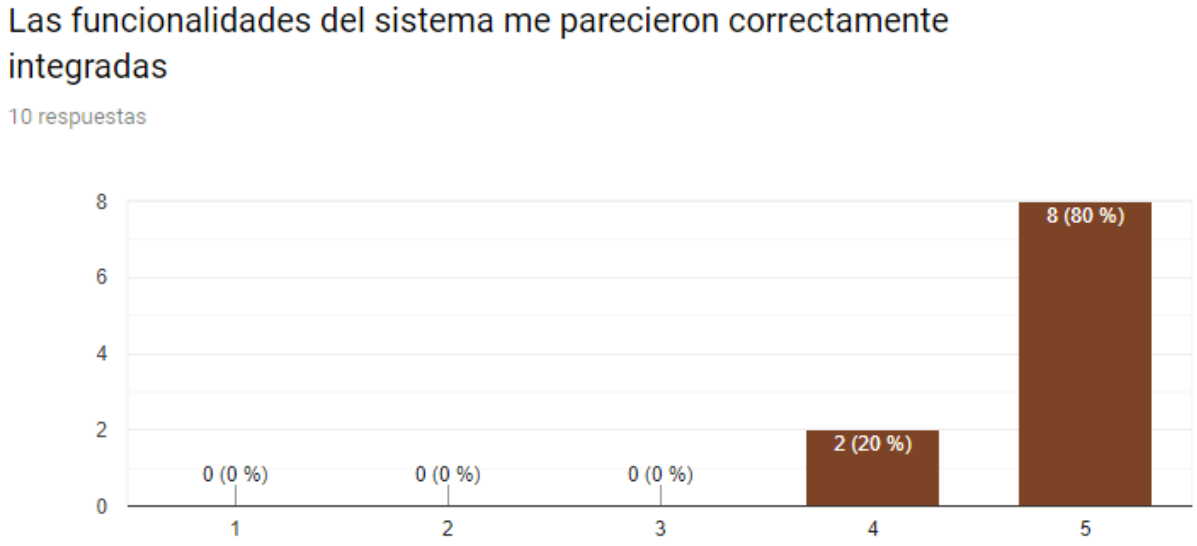



Figure 76: Gráfico de evaluación de la 8va pregunta
*Fuente.* Elaboración propia

La **pregunta 9** que es la que le sigue, permite evaluar si el software cumple con la finalidad para la que fue creada, si observamos la puntuación obtenida vemos que el 90% asegura que el sistema funciona como un software para la toma de decisiones, únicamente una persona piensa que el software le falta cierta funcionalidad para complementar lo desarrollado.

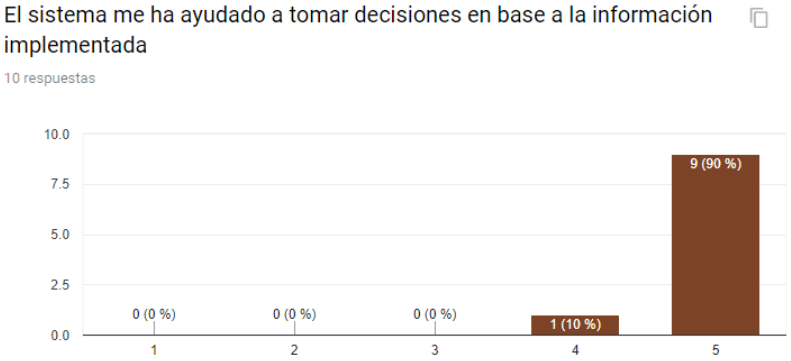



Figure 77: Gráfico de evaluación de la 9na pregunta
*Fuente.* Elaboración propia

### 6.2.2. Puntuaciones globales obtenidas

A continuación se levanto un gráfico global de las puntuaciones tanto de la funcionalidad y facilidad de uso del SIG. El primer gráfico nos muestra las puntuaciones de la facilidad de uso y observamos que el 80% de usuarios coincide que el sistema es fácil de usar, además que las puntuaciones están entre un rango del 4.4 y 5.0, por lo que se da por entendido que el sistema cumple con el objetivo de crear un software que sea de fácil uso.

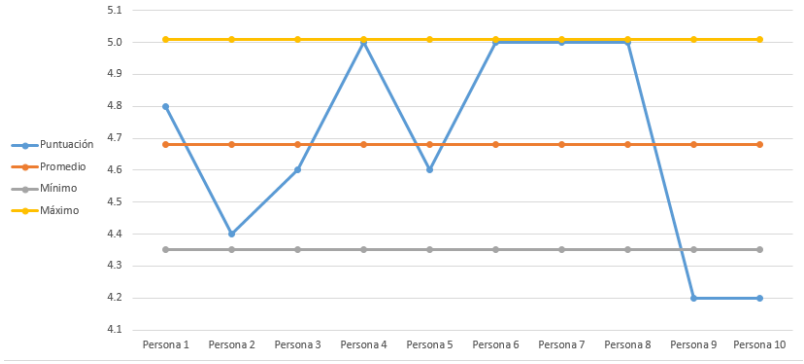



Figure 78: Evaluación facilidad
*Fuente.* Elaboración propia

En el siguiente gráfico tenemos las puntuaciones de la funcionalidad, en este podemos observar que el 90% de usuarios coinciden que el sistema cumple con las funcionalidades planteadas en las requerimientos y objetivos del proyecto, además que las puntuaciones oscilan en un rango del 4.6 y 5.0.

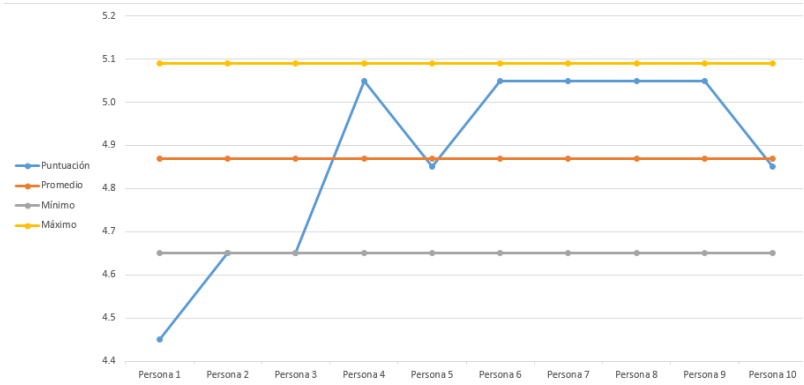



Figure 79: Evaluación funcionalidad
*Fuente.* Elaboración propia

### 6.2.3. Conclusiones

Después de haber evaluado los cuestionarios respondidos por cada uno de los usuarios participantes, puedo concluir que en cuanto a la facilidad al parecer los usuarios ven en el diseño de la aplicación con respecto al anterior software un avance, puesto que se incorporan nuevos gráficos, nuevas formas de navegar entre los módulos y sobre todo que los gráficos se adaptan a la pantalla en la que estén visualizando, estos temas de diseño fueron propuestos por mi persona al administrador y al parecer ofrecerles una interfaz simple sin necesidad de recorrer muchos pasos para realizar una tarea fue de agradecer por parte de los usuarios.

El software tiene la aceptación en cuanto a la funcionalidad por parte de los usuarios, puesto que los encuestados conocen, de un sistema previo, la información que presenta y no encontraron dificultad en este aspecto, mas bien se trato de presentar gráficamente información, de tal forma que a simple vista puedan visualizar su situación actual. Sin embargo, para usuarios que no conocen de temas financieros o no han trabajado con información de captaciones y colocaciones no encontrarán funcional al aplicativo, por lo que se entiende que el software esta orientado a usuarios que necesiten tomar decisiones en base a información que manejan diariamente y saben como tratar.

Con el antecedente en cuanto a la experticia de los usuarios no se ve la necesidad de llevar a cabo un plan de capacitación a los usuarios puesto que con solo navegar en el software y manipular los gráficos y tablas podrán rápidamente encontrar la información que buscan.

# 7. Conclusiones

Los objetivos que se plantearon para este trabajo de investigación y desarrollo de software fueron cumplidos en su totalidad y por tanto puedo argumentar lo siguiente:

- Se construyó un software desde cero que abarca información de todas las instituciones financieras del Ecuador, presentando información que pueda servir a los usuarios a tomar decisiones basándose en el comportamiento que han seguido la mayoría.
- Se levantó un repositorio central que contiene información de distintas instituciones, obtenidas mediante el uso de herramientas libres, cumpliendo así el objetivo de construir un almacén de datos que contenga información actual e histórica. Esta se irá alimentando dependiendo con la frecuencia en que se obtenga la información desde los

orígenes.

- Se elaboraron procesos automáticos que faciliten al administrador del sistema la descarga de los datos que alimentan al sistema de información gerencial, este era una de las grandes falencias del sistema anterior puesto que los procesos eran manuales, además se creó una base de datos en donde se puede parametrizar los atributos necesarios para los procesos de extracción actuales y de igual forma crear nuevos procesos.

- Se creó una base de datos adicional que permita la configuraciones de los sistemas de información gerencial en cuanto a las instituciones financieras, cuentas contables atadas a los productos de captaciones y colocaciones, módulos, usuarios y permisos.

- Se realizó un análisis de las herramientas que puedan ajustarse a las necesidades planteadas por el administrador del sistema de información gerencial en cuanto a términos de licencias y costos, para ello se decidió trabajar con software libre y tenemos el caso del lenguaje R, Python, Angular, Apache Spark y MongoDB, de tal forma brindo una solución libre que pueda seguir siendo trabajada por otras personas interesadas en potenciar al proyecto desarrollado.

- Puedo concluir que se cumplen altos criterios en cuanto a la usabilidad y facilidad del sistema por parte de los usuarios, aclarando que estos conocen de la información y saben que consultar y para que.

- En base a los cuestionarios, se revisamos la última pregunta que indica si los usuarios desean continuar con el nuevo software o mantenerse con el anterior, la mayoría de usuarios concuerdan que se necesita una actualización al software con nuevas tecnologías, de esta manera se abren las puertas para seguir añadiendo nuevas funcionalidades que como indique en un inicio quedaron fuera de este tema de TFM, pero se proponen como trabajos a futuro.

- Se realizó una investigación en cuanto a implementar Apache Hadoop para que sea el repositorio centralizado de datos, la mayoría de artículos concuerdan que la plataforma de Hadoop es óptima cuando se desea trabajar con billones de datos, los cuáles no tenemos en este proyecto de TFM, por lo que mi recomendación fue utilizar MongoDB como la base de datos del repositorio y Apache Spark como el motor de procesamiento, debido a que si en un futuro se desea integrar esta plataforma se lo pueda hacer ya que Spark se adapta al clúster de Hadoop.

- El software de almacenamiento y procesamiento usados brindan la capacidad de crecer tanto verticalmente como horizontalmente (recomendable), las configuraciones no resultan complejas ya que consiste añadir un nuevo nodo de trabajo y apuntar al servidor máster, modificando los archivos de configuración, además de agregar otros con la finalidad de optimizarlos para ambientes productivos.

Para concluir, de manera general este proyecto resulto ambicioso en cuanto a tiempos de trabajo, ya que se han dejado por fuera el uso de componentes que también dan su valor agregado al proyecto como es Apache Hadoop, Apache HBase, etc. pero me sirvió para poder entender que existe un gran abanico de soluciones tanto de pago como libres que a una empresa, no únicamente del ámbito financiero, puedan servir para que puedan tomar decisiones de manera oportuna y basándose en datos integrados y consistentes, a mi parecer juntar Business Intelligence mas Big Data y sumarle Business Analytics se pueden construir soluciones muy potentes que beneficiará a empresas y apoyarán a sus actividades cotidianas y a predecir comportamientos futuros.

# 8. Trabajo futuro

Es necesario tener claro que el sistema de información gerencial parte de un sistema anterior que estaba quedándose caduco en cuanto a las herramientas usadas y que los tiempos de extracción de la información eran mayores puesto que se los realizaba manualmente, sin embargo este sistema contiene muchos módulos adicionales que no fueron abarcados dentro del alcance del proyecto, y es ahí donde se puede ir implementando como trabajos a futuro, de esta manera se puede considerar lo siguiente:

- Construir nuevos módulos que permitan por ejemplo visualizar la información de balances de las instituciones financieras.
- Realizar predicciones de las captaciones o colocaciones mediante técnicas estadísticas como regresiones lineales a partir de información histórica.
- Con la misma información de balances que se extraen de las entidades de control, se pueden definir indicadores financieros como son la rentabilidad de las instituciones, la cobertura, el rendimiento, el retorno de la inversión, control de gastos, etc.
- Plantearse una actualización en cuanto a la infraestructura de los servidores de produc-

ción en donde se instalarán los componentes que forman parte del sistema de Información gerencial.

- Realizar una migración de la información histórica del sistema de información gerencial anterior al nuevo.
- Investigar y crear nuevos procesos de extracción de información que actualmente no se tienen.
- Se recomienda seguir con el proceso de desarrollo del sistema de información para implementar nuevas funcionalidades, esta línea de trabajo futuro va atada a la anterior.
- Como todo software se debe proceder con la actualización de los componentes a medida que surgen nuevas actualizaciones que resuelven *bugs* de los componentes actuales.
- Plantearse una nueva solución en cuanto a la implementación de Apache Hadoop en caso de que la información crezca con el tiempo y el repositorio Mongo empiece a quedarse, aunque Mongo tiene la capacidad de usar *replica sets* y *sharding* para optimizar su procesamiento e integridad de los datos.
- Se hace uso del motor de procesamiento Apache Spark, la razón de usarlo es que, muy aparte de que mejoran los tiempos de procesamiento de MapReduce, este se puede integrar, realizando las configuraciones correspondientes, a un clúster Hadoop, y de esta manera usar a Hadoop como repositorio y aprovechar el procesamiento paralelo del clúster, esto puede tratarse como una línea de trabajo futura.
- Al elaborar API's de consulta de los datos, estos vienen a ser independientes de la aplicación web por lo que se los puede hacer uso en cualquier otro sistema, e inclusive exponerlo a la internet para apoyar a otras empresas a la consulta de datos de balances, captaciones y colocaciones de forma libre, sin necesidad de que tengan acceso directo a la base de datos o tengan que nuevamente extraer la dato, el cuál es un proceso ya realizado.
- Optimizar la base de datos MongoDB para ofrecer *sharding* y *replica sets*.
- Crear un sistema de administración para facilitar la configuración de los procesos automáticos al administrador del sistema.
- Se debe implementar un punto de entrada seguro para las API's de consultas de los datos del repositorio en caso de exponerlos a la web, brindando soluciones de seguridad como pueden ser API Keys, Token de seguridad, etc.

# Bibliografía

Le Roy Miller, R. (1992). Moneda y Banca.

Zuleta, H. & Carvajal, A. (1997). Desarrollo del Sistema Financiero y Crecimiento Económico. *67*. Retrieved from http://www.banrep.gov.co/sites/default/files/publicaciones/pdfs/borra067.pdf

Calvo-Fernández Rodríguez, A., Ortega Santamaría, S., & Valls Saenz, A. (2004). Métodos de evaluación con usuarios, 94. Retrieved from http://openaccess.uoc.edu/webapps/o2/bitstream/10609/9861/4/PID_00176614.pdf

Rodríguez, H. A. S. (2009). ¿Cómo iniciar los proyectos de sistemas de información? *Sistemas de Información*, 17. Retrieved from https://bv.unir.net:2056/lib/univunirsp/detail.action?docID=3181174

CAF, B. d. D. d. A. L. (2011). Servicios Financieros para el desarrollo: Promoviendo el acceso américa latina. *Reporte de economía y desarrollo*.

Alicante, U. (2012). Introducción al Lenguaje Java. Retrieved, from http://www.jtech.ua.es/dadm/restringido/java/sesion01-apuntes.pdf

Schneider, R. D. (2012). Hadoop for dummies, 66–67. Retrieved from https://piazza-resources.s3.amazonaws.com/hqecl11rq5m6e5/hrnuizxzu4e20m/Hadoop.PDF

Rueda, N. (2013). Bancarización, profundización y densidad financiera del sistema financiero ecuatoriano. http://repositorio.puce.edu.ec/bitstream/handle/22000/12620/Disertaci%C3%B3n%20Neyla%20Rueda.pdf?sequence=1&isAllowed=y.

Asobanca. (2014). Boletín Informativo de la asociación de bancos privados del ecuador. http://www.asobanca.org.ec/file/588/download?token=MwpxpFRW.

Python. (2014). https://entrenamiento-python-basico.readthedocs.io/es/latest/index.html.

COSEDE, el guardián de tus depósitos. (2015). http://www.cosede.gob.ec/wp-content/uploads/2015/05/Diapositivas-COSEDE-para-p%C3%A1gina-web.pdf.

Gadi, M. (2015). Las nuevas finanzas se llaman 'big data', 5. Retrieved from http://www.espaciotv.es:2048/referer/secretcode/docview/1703055032?accountid=142712

Novatrix. (2015). Sistemas de información agilizan procesos y mejoran finanzas. https://bv.unir.net:2257/docview/1680139053?pq-origsite=summon.

HDFS Architecture. (2018). https://www.ibm.com/downloads/cas/WEB4XBOR.

Tejeda Yépez, S. (2018). Sistemas de Información Gerencial. *Sistemas de Información*, 11. doi:https://doi.org/10.35195/ob.v9i4.204

Angular.IO. (2019). Angular. Retrieved, from https://angular.io/docs

Apache Spark. (2019). https://spark.apache.org/docs/latest/index.html.

Asobanca. (2019). Boletín Macroeconómico. https://www.asobanca.org.ec/publicaciones/boletin-macroeconomico.

Bostock, M. (2019). D3.js. Retrieved, from https://d3js.org/

Historia de la Superintendencia. (2019). https://www.superbancos.gob.ec/bancos/historia-de-la-superintendencia/.

MongoDB. (2019). Manual de MongoDB. Retrieved, from https://docs.mongodb.com/manual/tutorial/getting-started/

Mozilla. (2019). Javascript. Retrieved, from https://developer.mozilla.org/es/docs/Web/JavaScript

Otero, M. F. & Huerga, M. N. (2019). Sistema de Información en la Empresa, 438–439. Retrieved from https://bv.unir.net:2056/lib/univunirsp/reader.action?docID=3226757

¿Qué es la SEPS? (2019). https://www.seps.gob.ec/interna?-que-es-la-seps-.

Schneider, R. & Karmiol, J. (2019). Spark for dummies, 80. Retrieved from https://piazza-resources.s3.amazonaws.com/hqecl11rq5m6e5/hrnuizxzu4e20m/Hadoop.PDF

Superintendencia de bancos: Misión y Visión. (2019). https://www.superbancos.gob.ec/bancos/mision-y-vision/.

Superintendencia de Economía Popular y Solidaria: Misión y Visión. (2019). https://www.seps.gob.ec/interna?vision-mision-atribuciones.

YARN Architecture. (2019). https://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html.

# Anexos

## Anexo I. Listado de Instituciones Financieras

A continuación se tiene un listado de los distintos bancos y sociedades financieras que se encuentran cubiertas por el seguro de depósitos COSEDE, como una breve descripción, todas las instituciones financieras privadas son controladas por la Superintendencia de Bancos y Seguros, por lo que todos los depositantes de dichas instituciones cuentan con la protección de forma automática y gratuita, el COSEDE es la corporación que administra estos seguros, por lo que el listado que se presenta a continuación es de una fuente confiable. («COSEDE, el guardián de tus depósitos», 2015).

| Razón Social | Subsistema | Estado Jurídico |
|---|---|---|
| Banco Amazonas S.A. | Bancos Privados | Activa |
| Banco Bolivariano S.A. | Bancos Privados | Activa |
| Citibank N.A. Sucursal Ecuador | Bancos Privados | Activa |
| Banco Capital S.A. | Bancos Privados | Activa |
| Banco Comercial de Manabí S.A. | Bancos Privados | Activa |
| Banco Coopnacional S.A. | Bancos Privados | Activa |
| Banco D-Miro S.A. | Bancos Privados | Activa |
| Banco Diners | Bancos Privados | Activa |
| Banco Guayaquil S.A. | Bancos Privados | Activa |
| Banco de Loja S.A. | Bancos Privados | Activa |
| Banco de Machala S.A. | Bancos Privados | Activa |
| Banco de la Producción | Bancos Privados | Activa |
| Produbanco S.A. | Bancos Privados | Activa |
| Banco del Austro S.A. | Bancos Privados | Activa |
| Banco del Litoral S.A. | Bancos Privados | Activa |
| Banco del Pacífico S.A. | Bancos Privados | Activa |
| Banco Delbank S.A. | Bancos Privados | Activa |
| Banco Desarrollo de los Pueblos S.A. | Bancos Privados | Activa |
| Banco General Rumiñahui S.A. | Bancos Privados | Activa |
| Banco Internacional S.A. | Bancos Privados | Activa |

| | | |
|---|---|---|
| Banco para la Asistencia Comunitaria Finca S.A. | Bancos Privados | Activa |
| Banco Pichincha S.A. | Bancos Privados | Activa |
| Banco Procredit S.A. | Bancos Privados | Activa |
| Banco Solidario S.A. | Bancos Privados | Activa |
| Banco VisionFund Ecuador S.A. | Bancos Privados | Activa |

Table 13: Listado Bancos Privados

De igual forma tenemos el listado de cooperativas y mutualistas, los cuales se agrupan en distintos segmentos del 1 al 5, y estos indican el volumnes de captaciones y colocaciones que manejan, para el proyecto se procederá a extraer la información de las cooperativas hasta el tercer segmento, que son las que por lo general la empresa Logical Consulting se enfoca a analizar.

| Segmento | Razón Social | Estado Jurídico |
|---|---|---|
| SEGMENTO 1 | Juventud Ecuatoriana Progresista | Organización Activa |
| | Jardín Azuayo LTDA. | Organización Activa |
| | San José LTDA. | Organización Activa |
| | De la Pequeña Empresa Biblián LTDA. | Organización Activa |
| | Riobamba LTDA. | Organización Activa |
| | Santa Rosa LTDA. | Organización Activa |
| | Atuntaqui LTDA. | Organización Activa |
| | Pilahiun LTDA. | Organización Activa |
| | Vicentina Manuel Estéban Godoy Ortega LTDA. | Organización Activa |
| | De la pequeña empresa de Pastaza LTDA. | Organización Activa |
| | 23 de Julio LTDA. | Organización Activa |
| | Andalucía LTDA. | Organización Activa |
| | Cooprogreso LTDA. | Organización Activa |
| | Alianza del Valle LTDA. | Organización Activa |
| | 29 de Octubre LTDA. | Organización Activa |
| | Policía Nacional LTDA. | Organización Activa |

| | | |
|---|---|---|
| | De los servidores públicos del Ministerio de Educación y Cultura | Organización Activa |
| | Oscus LTDA. | Organización Activa |
| | San Francisco LTDA. | Organización Activa |
| | El Sagrario LTDA. | Organización Activa |
| | Cámara de Comercio de Ambato LTDA. | Organización Activa |
| | Mushuc Runa LTDA. | Organización Activa |
| | Asociación Mutualista de Ahorro y Crédito para la Vivienda Pichincha | Organización Activa |
| | Asociación Mutualista de Ahorro y Crédito para la Vivienda Ambato | Organización Activa |
| | Asociación Mutualista de Ahorro y Crédito para la Vivienda Azuay | Organización Activa |
| | Asociación Mutualista de Ahorro y Crédito para la Vivienda Imbabura | Organización Activa |
| | Educadores del Azuay LTDA. | Organización Activa |
| | La Merced LTDA. | Organización Activa |
| | ERCO LTDA. | Organización Activa |
| | Alfonso Jaramillo León LTDA. | Organización Activa |
| | COOPAC Austro LTDA. | Organización Activa |
| | CREA LTDA. | Organización Activa |
| | Santa Isabel LTDA. | Organización Activa |
| | Guaranda LTDA. | Organización Activa |
| | Juan Pío De Mora LTDA. | Organización Activa |
| | Virgen del Cisne | Organización Activa |
| | Lucha Campesina | Organización Activa |
| | Fernando Daquilema | Organización Activa |
| | Once de Junio LTDA. | Organización Activa |
| | Armanda Nacional LTDA. | Organización Activa |
| | San Antonio LTDA. - Imbabura | Organización Activa |
| | Artesanos LTDA. | Organización Activa |
| | Mujeres Unidas Tantanakushka Warmikunapac | Organización Activa |

SEGMENTO 2

| | | |
|---|---|---|
| | Padre Julián Lorente LTDA. | Organización Activa |
| | Educadores de Loja LTDA. | Organización Activa |
| | De la pequeña empresa CACPE Loja LTDA. | Organización Activa |
| | Calceta LTDA. | Organización Activa |
| | Chone LTDA. | Organización Activa |
| | 15 de Abril LTDA. | Organización Activa |
| | Comercio LTDA. | Organización Activa |
| | De la Pequeña Empresa Gualaquiza | Organización Activa |
| | Cotocollao LTDA. | Organización Activa |
| | San Francisco de ASIS LTDA. | Organización Activa |
| | Construcción Comercio y Producción LTDA. | Organización Activa |
| | Textil 14 de Mayo | Organización Activa |
| | Previsión Ahorro y Desarrollo LTDA. | Organización Activa |
| | Mquita Cushunchic LTDA. | Organización Activa |
| | Luz del Valle | Organización Activa |
| | Indígena SAC LTDA. | Organización Activa |
| | Ambato LTDA. | Organización Activa |
| | Kullki Wasi LTDA. | Organización Activa |
| | Chibuleo LTDA. | Organización Activa |
| | De la pequeña empresa CAPCE Zamora LTDA. | Organización Activa |
| | Mas Ahorro Solidario MASCOOP | Organización Activa |
| | Multiempresarial | Organización Activa |
| | Fasayñan LTDA. | Organización Activa |
| | Señor de Girón | Organización Activa |
| | Gañansol LTDA. | Organización Activa |
| | Baños LTDA. | Organización Activa |
| | Promoción de Vida Asociada LTDA. PROVIDA | Organización Activa |
| | San Pedro LTDA. | Organización Activa |
| | Salinas LTDA. | Organización Activa |
| | Cañar LTDA. | Organización Activa |
| | Mushuk-Yuyai | Organización Activa |
| | Educadores Tulcán LTDA. | Organización Activa |

| | |
|---|---|
| San Gabriel LTDA. | Organización Activa |
| Educadores Primarios del Cotopaxi | Organización Activa |
| 9 de Octubre LTDA. | Organización Activa |
| Futuro Lamanense | Organización Activa |
| Sumak Kawsay LTDA. | Organización Activa |
| Andina LTDA. | Organización Activa |
| Sierra Centro LTDA. | Organización Activa |
| Visión de los Andes VISANDES | Organización Activa |
| Educadores de Chimborazo LTDA. | Organización Activa |
| San Jorge LTDA. | Organización Activa |
| San Miguel de Patallanga LTDA. | Organización Activa |
| Minga LTDA. | Organización Activa |
| 4 de Octubre | Organización Activa |
| La Doloresa LTDA. | Organización Activa |
| Universidad de Guayaquil LTDA. | Organización Activa |
| Saltre LTDA. | Organización Activa |
| Metropolitana LTDA. | Organización Activa |
| Base de Taura | Organización Activa |
| Nueva Huancavilca | Organización Activa |
| Unión El Ejido | Organización Activa |
| De Indígenas Chuchuqui LTDA. | Organización Activa |
| De Imbabura Amazonas | Organización Activa |
| Santa Anita LTDA. | Organización Activa |
| Acción Imbaburapak LTDA. | Organización Activa |
| Ecuacreditos LTDA. | Organización Activa |
| Cristo Rey | Organización Activa |
| De la Microempresa Fortuna | Organización Activa |
| Crediamigo LTDA. | Organización Activa |
| 13 de Abril | Organización Activa |
| San Antonio LTDA. | Organización Activa |
| Magisterio Manabita Limitada | Organización Activa |
| Santa Ana LTDA. | Organización Activa |

| | |
|---|---|
| La Benéfica LTDA. | Organización Activa |
| Microempresarial Sucre | Organización Activa |
| Tena LTDA. | Organización Activa |
| Coca LTDA. | Organización Activa |
| Educadores de Pastaza LTDA. | Organización Activa |
| San Cristóbal LTDA. | Organización Activa |
| Ciudad de Quito LTDA. | Organización Activa |
| Del Magisterio de Pichincha | Organización Activa |
| 16 de Julio LTDA. | Organización Activa |
| Alianza Minas LTDA. | Organización Activa |
| Politénica LTDA. | Organización Activa |
| Puellaro LTDA. | Organización Activa |
| Pedro Moncayo LTDA. | Organización Activa |
| San Juan de Cotogchoa | Organización Activa |
| Santa Ana de Nayon | Organización Activa |
| Juan de Salinas | Organización Activa |
| Del Distrito Metropolitano de Quito Amazonas | Organización Activa |
| Corporación Centro LTDA. | Organización Activa |
| San Miguel de los Bancos LTDA. | Organización Activa |
| De los Empleados Jubilados y Ex-empleados del Banco Central | Organización Activa |
| Unidad y Progreso | Organización Activa |
| San Vicente del Sur | Organización Activa |
| Hermes Gaibor Verdesoto | Organización Activa |
| Universidad Católica del Ecuador | Organización Activa |
| Fondo para el Desarrollo y la Vida | Organización Activa |
| Huaicana LTDA | Organización Activa |
| Suboficiales de la Policía Nacional | Organización Activa |
| Choco Tungurahua Runa LTDA. | Organización Activa |
| Cámara de Comercio de Santo Domingo | Organización Activa |
| Manantial de Oro LTDA. | Organización Activa |
| Coop Catar LTDA. | Organización Activa |

| | | |
|---|---|---|
| | Educadores de Tungurahua LTDA. | Organización Activa |
| | Maquita Cushun LTDA. | Organización Activa |
| | Crediambato LTDA. | Organización Activa |
| | Campesina COOPAC | Organización Activa |
| | Sembrando Un Nuevo País | Organización Activa |
| | Crecer Wiñari LTDA. | Organización Activa |
| | Indígena SAC Pelileo LTDA. | Organización Activa |
| | De la Pequeña Empresa CACPE Yantza LTDA. | Organización Activa |
| | Educadores de Zamora Chinchipe LTDA. | Organización Activa |

Table 14: Listado Cooperativas y Mutualistas

## Anexo II. Recolección de los requisitos

- **Necesidad de un nuevo Sistema de Información Gerencial**

  Actualmente, la empresa Logical Consulting busca mantener la fidelización actual de sus clientes y la adquisición de nuevos, pero existen factores externos a la empresa que le obligan a plantearse nuevas soluciones, una de estas es la competencia, ya que estos ofrecen nuevas tecnologías, y la versión actual del sistema se está quedando rezagado. El plus de la empresa además del software, es que tiene la experiencia y el conocimiento para no solo ofrecer un producto software sino de brindarles una solución y asesoría a nivel gerencial y enfocado a temas financieros. La necesidad por lo tanto está enfocada a construir un nuevo sistema libre como muestra al público, para que puedan consultar información limitada, y sirva como enganche para nuevos clientes, además de informar a los actuales de que se viene una nueva actualización del sistema.

- **Tareas a realizarse**

  - Automatización de la extracción de la información de bancos, cooperativas y mutualistas.
  - Implementación de tecnologías big data para el nuevo sistema.
  - Construcción de un nuevo software web donde se tendrá un dashboard principal de los datos.

- **Descripción del sistema actual**

Actualmente la empresa tiene un software cuya base de datos es Microsoft Excel, y un sistema web que lee de este archivo, y presenta gráficas y análisis de los datos, el proceso de carga de los datos a esta base es manual, extrae la información de las fuentes, realiza los cambios y el tratamiento de la data en un archivo de Excel y lo procede a insertar a la base el cual como menciono es un mismo excel.

- **Descripción de la información a recolectar**
  - Obtener datos de los balances generales de bancos de la página de la superintendencia de bancos.
  - Obtener los boletines financieros de las cooperativas y mutualistas de la página de la superintendencia de economía popular y solidaria.
- **Definición del acuerdo legal entre la empresa y el desarrollador**

  Como requisito de la UNIR y parte del TFM y para evitar tener un problema legal con la empresa por basarse el desarrollo en un proyecto anterior, se llegó a un acuerdo en donde la empresa se compromete a facilitar el conocimiento necesario y el requerimiento de la aplicación para construir un nuevo Sistema de Información Gerencial en donde únicamente se visualice información de captaciones, colocaciones de los bancos, cooperativas y mutualistas comparándola con el mismo mes del año anterior. Del lado del gestor del desarrollo y configuración, el compromiso consiste en construir el software libre de tal forma que cumpla el requerimiento de la empresa y se cumpla las definiciones de la UNIR, entregarle la versión final del código fuente a la empresa, y cualquier nuevo desarrollo adicional al aplicativo corre por cuenta de la empresa y queda fuera del proyecto TFM y por ende es código privado, la empresa decidirá si se contacta con el mismo desarrollador o con un nuevo.
- **Detalles Generales**

  Se tiene un bosquejo de lo que la empresa desea visualizar en el sistema de información gerencial.

SETEAR QUÉ INSTITUCIONES DE ANALISIS

TIPO DE INSTITUCION

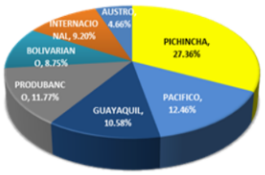
PRODUCTO O INDICADOR

**CAPTACIONES**

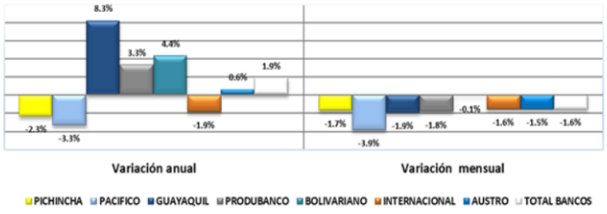
**(en miles de dólares)**

| INSTITUCION | 20180131 | | | 20190131 | | | Variación anual | |
|---|---|---|---|---|---|---|---|---|
| | VALOR | PART. | RK | VALOR | PART. | RK | Valor | % |
| PICHINCHA | 8,619,067 | 28.54% | 1 | 8,417,939 | 27.36% | 1 | -201,128 | -2.3% |
| PACIFICO | 3,965,328 | 13.13% | 2 | 3,834,057 | 12.46% | 2 | -131,270 | -3.3% |
| GUAYAQUIL | 3,007,844 | 9.96% | 4 | 3,256,363 | 10.58% | 4 | 248,519 | 8.3% |
| PRODUBANCO | 3,505,147 | 11.61% | 3 | 3,621,210 | 11.77% | 3 | 116,063 | 3.3% |
| BOLIVARIANO | 2,580,042 | 8.54% | 6 | 2,692,316 | 8.75% | 6 | 112,275 | 4.4% |
| INTERNACIONAL | 2,885,930 | 9.56% | 5 | 2,832,116 | 9.20% | 5 | -53,814 | -1.9% |
| AUSTRO | 1,424,851 | 4.72% | 7 | 1,433,429 | 4.66% | 7 | 8,578 | 0.6% |
| DINERS | 853,546 | 2.83% | 8 | 1,109,474 | 3.61% | 8 | 255,928 | 30.0% |
| SOLIDARIO | 408,813 | 1.35% | 12 | 421,409 | 1.37% | 12 | 12,596 | 3.1% |
| MACHALA | 625,021 | 2.07% | 10 | 675,018 | 2.19% | 10 | 49,997 | 8.0% |
| RUMINAHUI | 673,304 | 2.23% | 9 | 678,007 | 2.20% | 9 | 4,703 | 0.7% |
| CITIBANK | 534,840 | 1.77% | 11 | 672,358 | 2.19% | 11 | 137,518 | 25.7% |
| LOJA | 364,813 | 1.21% | 13 | 379,130 | 1.23% | 13 | 14,317 | 3.9% |
| PROCREDIT | 169,043 | 0.56% | 14 | 145,961 | 0.47% | 16 | -23,081 | -13.7% |
| **TOTAL BANCOS** | **30,200,740** | **79.04%** | | **30,771,061** | **77.47%** | | **570,321** | **1.9%** |
| **TOTAL COOPERATIVA** | **7,205,315** | **18.86%** | | **8,117,059** | **20.44%** | | **911,743** | **12.7%** |
| **TOTAL MUTUALISTAS** | **788,842** | **2.06%** | | **829,832** | **2.09%** | | **40,990** | **5.2%** |
| **TOTAL FINANCIERAS** | **13,350** | **0.03%** | | **0** | **0.00%** | | **-13,350** | **-100.0%** |
| **SIST. FINANCIERO** | **38,208,248** | **100.00%** | | **39,717,951** | **100.00%** | | **1,509,703** | **4.0%** |

**Market share**




**Crecimiento Bancos**

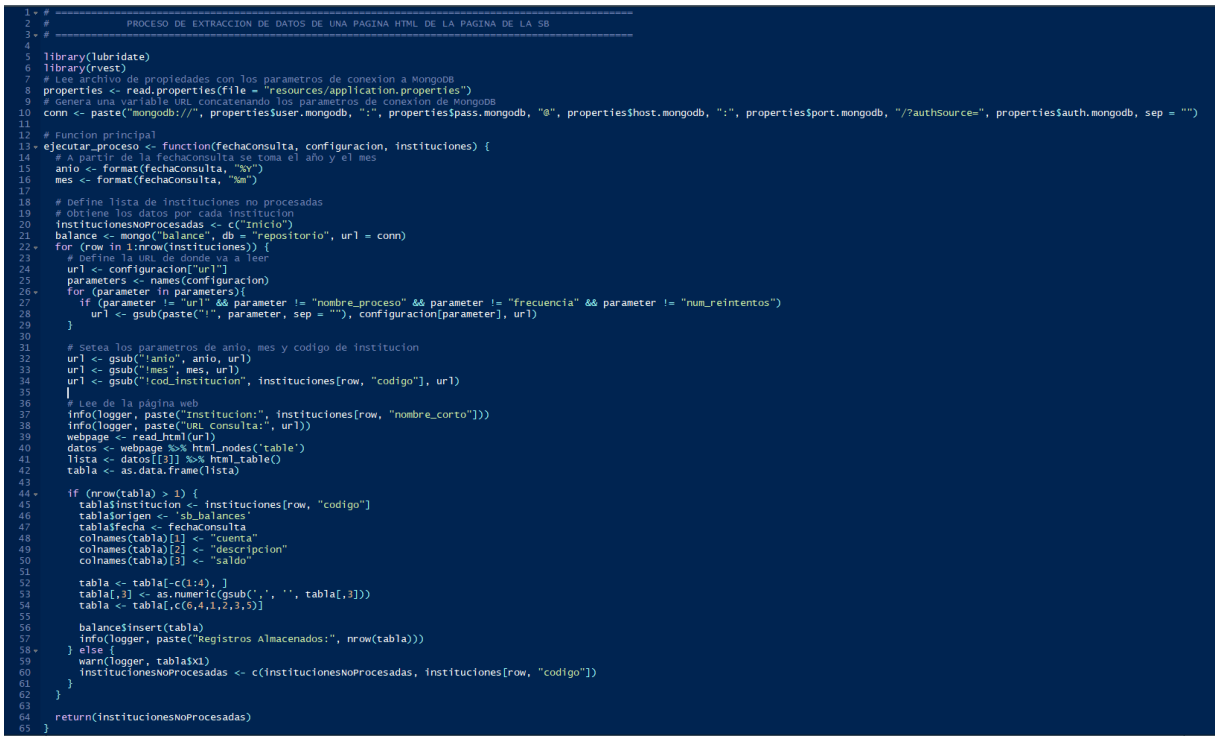


## Anexo III. Procesos R: Extracción Fuentes de Información

A continuación se presenta el código empleado para la extracción de información de las fuentes definidas en este proyecto.

### Proceso Balances Bancos (process_balance_html_sb)

Este permite extraer los balances de las instituciones bancarias del ecuador, que se encuentran en una tabla dentro de una página web, es decir, se realiza un web scrapping.



```r
# ================================================================================================
#             PROCESO DE EXTRACCION DE DATOS DE UNA PAGINA HTML DE LA PAGINA DE LA SB
# ================================================================================================

library(lubridate)
library(rvest)
# Lee archivo de propiedades con los parametros de conexion a MongoDB
properties <- read.properties(file = "resources/application.properties")
# Genera una variable URL concatenando los parametros de conexion de MongoDB
conn <- paste("mongodb://", properties$user.mongodb, ":", properties$pass.mongodb, "@", properties$host.mongodb, ":", properties$port.mongodb, "/?authSource=", properties$auth.mongodb, sep = "")

# Funcion principal
ejecutar_proceso <- function(fechaConsulta, configuracion, instituciones) {
  # A partir de la fechaConsulta se toma el año y el mes
  anio <- format(fechaConsulta, "%Y")
  mes <- format(fechaConsulta, "%m")

  # Define lista de instituciones no procesadas
  # Obtiene los datos por cada institucion
  institucionesNoProcesadas <- c("Inicio")
  balance <- mongo("balance", db = "repositorio", url = conn)
  for (row in 1:nrow(instituciones)) {
    # Define la URL de donde va a leer
    url <- configuracion["url"]
    parameters <- names(configuracion)
    for (parameter in parameters){
      if (parameter != "url" && parameter != "nombre_proceso" && parameter != "frecuencia" && parameter != "num_reintentos")
        url <- gsub(paste("!", parameter, sep = ""), configuracion[parameter], url)
    }

    # Setea los parametros de anio, mes y codigo de institucion
    url <- gsub("!anio", anio, url)
    url <- gsub("!mes", mes, url)
    url <- gsub("!cod_institucion", instituciones[row, "codigo"], url)

    # Lee de la página web
    info(logger, paste("Institucion:", instituciones[row, "nombre_corto"]))
    info(logger, paste("URL Consulta:", url))
    webpage <- read_html(url)
    datos <- webpage %>% html_nodes('table')
    lista <- datos[[3]] %>% html_table()
    tabla <- as.data.frame(lista)

    if (nrow(tabla) > 1) {
      tabla$institucion <- instituciones[row, "codigo"]
      tabla$origen <- 'sb_balances'
      tabla$fecha <- fechaConsulta
      colnames(tabla)[1] <- "cuenta"
      colnames(tabla)[2] <- "descripcion"
      colnames(tabla)[3] <- "saldo"

      tabla <- tabla[-c(1:4), ]
      tabla[,3] <- as.numeric(gsub(',', '', tabla[,3]))
      tabla <- tabla[,c(6,4,1,2,3,5)]

      balance$insert(tabla)
      info(logger, paste("Registros Almacenados:", nrow(tabla)))
    } else {
      warn(logger, tabla$X1)
      institucionesNoProcesadas <- c(institucionesNoProcesadas, instituciones[row, "codigo"])
    }
  }

  return(institucionesNoProcesadas)
}
```

### Proceso Balances Cooperativas (process_balance_xls_seps)

Este permite extraer los balances de las cooperativas de los segmentos 1, 2 y mutualistas del ecuador, que se encuentran en un documento de excel dentro de una página web, es decir, se realiza una descarga del archivo y su posterior lectura.

```r
library(mongolite)
library(httr)
library(readxl)
library(rvest)

# Lee archivo de propiedades con los parametros de conexion a MongoDB
properties <- read.properties(file = "resources/application.properties")
# Genera una variable URL concatenando los parametros de conexion de MongoDB
conn <- paste("mongodb://", properties$user.mongodb, ":", properties$pass.mongodb, "@", properties$host.mongodb, ":", properties$port.mongodb, "/?authSource=", properties$auth.mongodb, sep = "")
meses <- c("Ene","Feb","Mar","Abr","May","Jun","Jul","Ago","Sep","Oct","Nov","Dic")

ejecutar_proceso <- function(fechaConsulta, configuracion, instituciones) {
  # Obtiene el año y mes a partir de la fecha de consulta
  mes <- meses[month(as.POSIXlt(fechaConsulta))]
  anio <- format(fechaConsulta, "%y")
  dateFile <- paste(mes, anio, sep = "")

  # Genera ubicacion de descarga del archivo
  destFile <- paste("files/", configuracion["prefijo_documento"], "_", dateFile, "_", configuracion["segmento"], ".", configuracion["formato"], sep = "")

  # Consulta el enlace de descarga del archivo directamente en la página web
  url <- as.character(configuracion["url"])
  webpage <- read_html(url)
  enlaces <- webpage %>% html_nodes('a') %>% html_attr('href')

  # Selecciona los enlaces que contienen la palabra Boletin_financiero
  # Selecciona los enlaces que corresponden a la fecha de consulta
  # Selecciona el enlace que corresponde al segmento 1
  boletines <- enlaces[grep("Boletin_financiero", enlaces)]
  boletinesAnioActual <- boletines[grep(anio, boletines)]
  boletinesMesActual <- boletinesAnioActual[grep(mes, boletinesAnioActual)]
  linkSegmento <- boletinesMesActual[grep(configuracion["segmento"], boletinesMesActual)]

  info(logger, paste("Pagina web del archivo:", url))

  # Esta variable permite almacenar en la base de datos si existio o no archivo para descargar y volver a reprocesar
  institucionesNoProcesadas <- c("Inicio")
  if (length(linkSegmento) > 0) {
    link <- paste(substr(url, 0, 23), linkSegmento[1], sep = "")
    info(logger, paste("Enlace de desacarga del archivo:", link))
    info(logger, paste("Ubicacion de descarga:", destFile))

    # Valida si existe el archivo, sino existe lo descarga
    if (!file.exists(destFile))
      download.file(link, destFile, method = "curl")

    # Lee del archivo descargado la hoja 3 que corresponde a los balances
    tabla <- as.data.frame(read_excel(destFile, sheet = as.integer(configuracion["sheet"])))

    # Realiza un formateo a la hoja eliminado filas y columnas innecesarias
    tabla <- tabla[-c(1:6), ]
    tabla <- tabla[, -c(3:4)]
    tabla <- tabla[,colSums(is.na(tabla)) < nrow(tabla)]
    tabla <- tabla[1:(length(tabla)-2)] # Para el archivo S1 se elimina las dos ultimas columnas ya que son totales
    tabla <- na.omit(tabla) # Omite las filas con NA

    columnaCuenta <- tabla[2:nrow(tabla), 1]
    columnaDescripcion <- tabla[2:nrow(tabla), 2]

    balance <- mongo("balance", db = "repositorio", url = conn) # Cambiar a balance en ves de test
    institucionDB <- mongo("institucion_financiera", db = "administracion_sig", url = conn)

    for (column in 3:length(tabla)) {
      #TODO: se debe buscar el elemento del archivo y compararlo con el elemento que envia el reproceso o el proceso normal
      institucion <- institucionDB$find(paste('{ "nombre_largo": "', tabla[1, column], '", "estatus": true }', sep = ""))
      if (length(institucion)) {
        info(logger, paste("Institucion:", institucion$nombre_corto))

        newTabla <- data.frame("fecha" = fechaConsulta,
                               "institucion" = institucion$codigo,
                               "cuenta" = columnaCuenta,
                               "descripcion" = columnaDescripcion,
                               "saldo" = as.numeric(gsub(",","",tabla[2:nrow(tabla), column])),
                               "origen" = "seps_balances")

        balance$insert(newTabla)
        info(logger, paste("Registros almacenados:", nrow(tabla) - 1))
      } else {
        if (institucion$codigo) {
          warn(logger, paste("La institucion ", tabla[1, column], ", existe en el archivo pero no en la base de datos."))
          institucionesNoProcesadas <- c(institucionesNoProcesadas, institucion$codigo)
        } else {
          info(logger, paste("La institucion ", tabla[1, column], ", esta deshabilitada."))
        }
      }
    }
  } else {
    error(logger, "No existe documento disponible todavia.")
    institucionesNoProcesadas <- c(institucionesNoProcesadas, "todas")
  }

  return(institucionesNoProcesadas)
}
```

### Proceso Balances Cooperativas Segmento 3 (process_balance_xls_seps_s2)

Este permite extraer los balances de las cooperativas del segmentos 3, que se encuentran en un documento de excel dentro de una página web, es decir, se realiza una descarga del archivo y su posterior lectura, pero difiere en ciertos campos del archivo anterior.

```r
library(mongolite)
library(httr)
library(rvest)
library(readxl)

# Lee archivo de propiedades con los parametros de conexion a MongoDB
properties <- read.properties(file = "resources/application.properties")
# Genera una variable URL concatenando los parametros de conexion de MongoDB
conn <- paste("mongodb://", properties$user.mongodb, ":", properties$pass.mongodb, "@", properties$host.mongodb, ":", properties$port.mongodb, "/?authSource=", properties$auth.mongodb, sep = "")
meses <- c("Ene","Feb","Mar","Abr","May","Jun","Jul","Ago","Sep","Oct","Nov","Dic")

ejecutar_proceso <- function(fechaConsulta, configuracion, instituciones) {
  # Obtiene el año y mes a partir de la fecha de consulta
  mes <- meses[month(as.POSIXlt(fechaConsulta))]
  anio <- format(fechaConsulta, "%y")
  dateFile <- paste(mes, anio, sep = "")

  # Genera ubicacion de descarga del archivo
  destFile <- paste("files/", configuracion["prefijo_documento"], "_", dateFile, "_", configuracion["segmento"], ".", configuracion["formato"], sep = "")

  # Consulta el enlace de descarga del archivo directamente en la página web
  url <- as.character(configuracion["url"])
  webpage <- read_html(url)
  enlaces <- webpage %>% html_nodes('a') %>% html_attr('href')

  # Selecciona los enlaces que contienen la palabra Boletin_financiero
  # Selecciona los enlaces que corresponden a la fecha de consulta
  # Selecciona el enlace que corresponde al segmento diferente de 1, pudiendo ser 2, 3, MUT
  boletines <- enlaces[grep("Boletin_financiero", enlaces)]
  boletinesAnioActual <- boletines[grep(anio, boletines)]
  boletinesMesActual <- boletinesAnioActual[grep(mes, boletinesAnioActual)]
  linkSegmento <- boletinesMesActual[grep(configuracion["segmento"], boletinesMesActual)]

  info(logger, paste("Pagina web del archivo:", url))

  # Esta variable permite almacenar en la base de datos si existio o no archivo para descargar y volver a reprocesar
  institucionesNoProcesadas <- c("Inicio")
  if (length(linkSegmento) > 0) {
    link <- paste(substr(url, 0, 23), linkSegmento[1], sep = "")
    info(logger, paste("Enlace de descarga del archivo:", link))
    info(logger, paste("Ubicacion de descarga:", destFile))

    # Valida si existe el archivo, sino existe lo descarga
    if (!file.exists(destFile))
      download.file(link, destFile, method = "curl")
    print(configuracion["sheet"])
    # Lee del archivo descargado la hoja 3 que corresponde a los balances
    tabla <- as.data.frame(read_excel(destFile, sheet = as.integer(configuracion["sheet"])))

    # Realiza un formateo a la hoja eliminado filas y columnas innecesarias
    tabla <- tabla[-c(1:6), ]
    tabla <- tabla[, -c(3:4)]
    tabla <- tabla[,colSums(is.na(tabla)) < nrow(tabla)]
    tabla <- tabla[1:(length(tabla)-1)] # Para el archivo S2, S3 se elimina las ultima columna ya que es total
    tabla <- na.omit(tabla) # Omite las filas con NA

    columnaCuenta <- tabla[2:nrow(tabla), 1]
    columnaDescripcion <- tabla[2:nrow(tabla), 2]

    balance <- mongo("balance", db = "repositorio", url = conn) # Cambiar a balance en ves de test
    institucionDB <- mongo("institucion_financiera", db = "administracion_sig", url = conn)

    for (column in 3:length(tabla)) {
      #TODO: se debe buscar el elemento del archivo y compararlo con el elemento que envia el reproceso o el proceso normal
      institucion <- institucionDB$find(paste('{ "nombre_largo": "', tabla[1, column], '", "estatus": true }', sep = ""))
      if (length(institucion)) {
        info(logger, paste("Institucion:", institucion$nombre_corto))

        newTabla <- data.frame("fecha" = fechaConsulta,
                               "institucion" = institucion$codigo,
                               "cuenta" = columnaCuenta,
                               "descripcion" = columnaDescripcion,
                               "saldo" = as.numeric(gsub(",","",tabla[2:nrow(tabla), column])),
                               "origen" = paste("seps_balances_", tolower(configuracion["segmento"]), sep = ""))

        balance$insert(newTabla)
        info(logger, paste("Registros almacenados:", nrow(tabla) - 1))
      } else {
        if (institucion$codigo) {
          warn(logger, paste("La institucion ", tabla[1, column], ", existe en el archivo pero no en la base de datos."))
          institucionesNoProcesadas <- c(institucionesNoProcesadas, institucion$codigo)
        } else {
          info(logger, paste("La institucion ", tabla[1, column], ", esta deshabilitada."))
        }
      }
    }
  } else {
    error(logger, "No existe documento disponible todavia.")
    institucionesNoProcesadas <- c(institucionesNoProcesadas, "todas")
  }

  return(institucionesNoProcesadas)
}
```

Como podemos observar en esta ocasión se ha creado tres procesos de extracción de datos, pero se pueden desarrollar nuevos procesos e incluirlos dentro del mismo proyecto para que puedan ser ejecutados por el proceso batch normal como el de reprocesos.

## Anexo IV. Aplicaciones Spark

En este anexo se puede observar el código fuente empleado para la clase principal __main__.py, consolidador_posicion.py y consolidador_variacion.py, estos conforman el proceso de consolidación de los datos que son la entrada para el sistema de información gerencial.

### Clase Principal (__main__)

Esta clase python se encarga de ser el archivo inicial para una aplicación Spark, recibe como entrada dos parámetros que son la fecha y el proceso a realizar (posición o variación), esta clase ***"main"*** valida que se mande los parámetros, caso contrario no continua el proceso.

```python
from data.consolidador_posicion import ConsolidadorPosicion
from data.consolidador_variacion import ConsolidadorVariacion

import sys

def main():
    if sys.argv.__len__() != 3:
        print("Falta el argumento <fecha> <tipo>")
        return

    if sys.argv[2] == "posicion":
        src = ConsolidadorPosicion(sys.argv[1])
        src.run()
    else:
        src = ConsolidadorVariacion(sys.argv[1])
        src.run()

if __name__ == "__main__":
    main()
```

### Clase Consolidador Posición (consolidador_posicion)

Genera la estructura de posición de datos tanto para las colocaciones y captaciones de las instituciones financieras.

```python
# coding=utf-8
from pyspark import SparkConf, SparkContext
from pyspark.sql import SparkSession

from configparser import ConfigParser # se debe cambiar de configparse a ConfigParser en produccion
import os


class ConsolidadorPosicion:
    def __init__(self, fecha):
        ROOT_DIR = os.path.dirname(os.path.abspath(__file__))

        config = ConfigParser()
        config.read(os.path.join(ROOT_DIR, 'data_source.ini'))
        master_url = config.get('spark', 'master_url')
        app_name = config.get('spark', 'app_name')
        mongo_jar = config.get('spark', 'mongo_jar')

        self.mongodb = config.get('spark', 'mongodb_url')
        self.fecha = fecha

        conf = SparkConf().setAppName(app_name).setMaster(master_url).set("spark.jars.packages", mongo_jar)

        self.sc = SparkContext(conf=conf)
        self.spark = SparkSession.builder.config(conf=conf).config("spark.mongodb.output.uri", self.mongodb).getOrCreate()

    def run(self):
        self.spark.read.format("com.mongodb.spark.sql.DefaultSource").option(
            "uri", self.mongodb + "administracion_sig.producto_cuenta").load().createOrReplaceTempView("productoCuentas")

        self.spark.read.format("com.mongodb.spark.sql.DefaultSource").option(
            "uri", self.mongodb + "repositorio.balance").load().filter("fecha = '" + self.fecha + "'").createOrReplaceTempView("balance")

        self.spark.read.format("com.mongodb.spark.sql.DefaultSource").option(
            "uri", self.mongodb + "administracion_sig.institucion_financiera").load().createOrReplaceTempView("instituciones")

        self.spark.sql("SELECT b.fecha, i.tipo as tipo_institucion, i.nombre_corto, i.nombre_largo, pc.producto, pc.tipo, pc.segmento, sum(b.saldo) as saldo \
            FROM balance b INNER JOIN productoCuentas pc ON b.cuenta = pc.Cuenta  \
            LEFT JOIN instituciones i on i.codigo = b.institucion \
            GROUP BY b.fecha, i.tipo, i.nombre_corto, i.nombre_largo, pc.producto, pc.tipo, pc.segmento").createOrReplaceTempView("productos")

        pasivas = self.spark.sql("SELECT fecha, tipo_institucion, nombre_corto, nombre_largo, producto, sum(saldo) as saldo FROM productos \
            WHERE producto IN ('Ahorros', 'Monetarios', 'Plazo', 'Otros Depositos') \
            GROUP BY fecha, tipo_institucion, nombre_corto, nombre_largo, producto")

        activas = self.spark.sql("SELECT fecha, tipo_institucion, nombre_corto, nombre_largo, producto, segmento, tipo as segmento_cartera, sum(saldo) as saldo FROM productos \
            WHERE producto IN ('Comercial', 'Consumo', 'Microcredito', 'Vivienda') \
            GROUP BY fecha, tipo_institucion, nombre_corto, nombre_largo, producto, segmento, tipo")

        pasivas.write.format("com.mongodb.spark.sql.DefaultSource").mode("append").option("database", "repositorio").option("collection", "posicion_pasivas").save()
        activas.write.format("com.mongodb.spark.sql.DefaultSource").mode("append").option("database", "repositorio").option("collection", "posicion_activas").save()
```

### Clase Consolidador Variación (consolidador_variacion)

Genera la estructura de variación de los datos de una fecha actual correspondiente a una fecha del mes anterior, fin de año anterior y del mismo mes año anterior.

```python
# coding=utf-8
from pyspark import SparkConf, SparkContext
from pyspark.sql import SparkSession
from pyspark.sql import functions as F

from configparser import ConfigParser # se debe cambiar de configparse a ConfigParser en produccion
import datetime
import os
class ConsolidadorVariacion:
    def __init__(self, fecha):
        ROOT_DIR = os.path.dirname(os.path.abspath(__file__))

        config = ConfigParser()
        config.read(os.path.join(ROOT_DIR, 'data_source.ini'))
        master_url = config.get('spark', 'master_url')
        app_name = config.get('spark', 'app_name')
        mongo_jar = config.get('spark', 'mongo_jar')

        self.mongodb = config.get('spark', 'mongodb_url')
        self.fecha = fecha

        conf = SparkConf().setAppName(app_name).setMaster(master_url).set("spark.jars.packages", mongo_jar)

        self.sc = SparkContext(conf=conf)
        self.spark = SparkSession.builder.config(conf=conf).config("spark.mongodb.output.uri", self.mongodb).getOrCreate()

    def run(self):
        d = datetime.datetime.strptime(self.fecha, "%Y-%m-%d")
        anio = d.year
        mes = d.month

        anioAnterior = d.year - 1
        mesAnterior = mes - 1

        if mesAnterior == 0:
            mesAnterior = 12
            anio = anioAnterior

        fechaFinMesAnterior = ConsolidadorVariacion.last_day_of_month(datetime.date(anio, mesAnterior, 1))
        fechaFinAnioAnterior = datetime.date(anioAnterior, 12, 31)
        fechaFinMesAnioAnterior = ConsolidadorVariacion.last_day_of_month(datetime.date(anioAnterior, mes, 1))

        self.spark.read.format("com.mongodb.spark.sql.DefaultSource").option("uri", self.mongodb + "repositorio.posicion_activas").load() \
            .filter("fecha in ('" + self.fecha + "', '" + str(fechaFinMesAnioAnterior) + "', '" + str(fechaFinMesAnterior) + "', '" + str(fechaFinAnioAnterior) + "')") \
            .createOrReplaceTempView("activas")

        self.spark.read.format("com.mongodb.spark.sql.DefaultSource").option("uri", self.mongodb + "repositorio.posicion_pasivas").load() \
            .filter("fecha in ('" + self.fecha + "', '" + str(fechaFinMesAnioAnterior) + "', '" + str(fechaFinMesAnterior) + "', '" + str(fechaFinAnioAnterior) + "')") \
            .createOrReplaceTempView("pasivas")

        # PASIVAS
        actualPasivas = self.spark.sql("SELECT * FROM pasivas WHERE fecha = '" + self.fecha + "'").withColumnRenamed("saldo", "valor_final").withColumn("valor_inicial", F.lit(0))
        finMesAnteriorPasivas = self.spark.sql("SELECT * FROM pasivas WHERE fecha = '" + str(fechaFinMesAnterior) + "'").withColumnRenamed("saldo", "valor_inicial").withColumn("valor_final",
F.lit(0)).withColumn("fecha", F.lit(self.fecha))
        fechaFinAnioAnteriorPasivas = self.spark.sql("SELECT * FROM pasivas WHERE fecha = '" + str(fechaFinAnioAnterior) + "'").withColumnRenamed("saldo", "valor_inicial").withColumn("valor_final",
F.lit(0)).withColumn("fecha", F.lit(self.fecha))
        fechaFinMesAnioAnteriorPasivas = self.spark.sql("SELECT * FROM pasivas WHERE fecha = '" + str(fechaFinMesAnioAnterior) + "'").withColumnRenamed("saldo", "valor_inicial").withColumn("valor_final",
F.lit(0)).withColumn("fecha", F.lit(self.fecha))

        actualPasivas.select("fecha", "tipo_institucion", "nombre_corto", "nombre_largo", "producto", "valor_inicial", "valor_final") \
            .union(finMesAnteriorPasivas.select("fecha", "tipo_institucion", "nombre_corto", "nombre_largo", "producto", "valor_inicial", "valor_final")) \
            .groupBy("fecha", "tipo_institucion", "nombre_corto", "nombre_largo", "producto") \
            .agg(F.sum("valor_inicial").alias("valor_inicial"), F.sum("valor_final").alias("valor_final")) \
            .withColumn("tipo", F.lit("MENSUAL")).createOrReplaceTempView("variacionesPasivasMensual")

        actualPasivas.select("fecha", "tipo_institucion", "nombre_corto", "nombre_largo", "producto", "valor_inicial", "valor_final") \
            .union(fechaFinAnioAnteriorPasivas.select("fecha", "tipo_institucion", "nombre_corto", "nombre_largo", "producto", "valor_inicial", "valor_final")) \
            .groupBy("fecha", "tipo_institucion", "nombre_corto", "nombre_largo", "producto") \
            .agg(F.sum("valor_inicial").alias("valor_inicial"), F.sum("valor_final").alias("valor_final")) \
            .withColumn("tipo", F.lit("ACUMULADA")).createOrReplaceTempView("variacionesPasivasAcumulada")

        actualPasivas.select("fecha", "tipo_institucion", "nombre_corto", "nombre_largo", "producto", "valor_inicial", "valor_final") \
            .union(fechaFinMesAnioAnteriorPasivas.select("fecha", "tipo_institucion", "nombre_corto", "nombre_largo", "producto", "valor_inicial", "valor_final")) \
            .groupBy("fecha", "tipo_institucion", "nombre_corto", "nombre_largo", "producto") \
            .agg(F.sum("valor_inicial").alias("valor_inicial"), F.sum("valor_final").alias("valor_final")) \
            .withColumn("tipo", F.lit("ANUAL")).createOrReplaceTempView("variacionesPasivasAnual")

        # ACTIVAS
        actualActivas = self.spark.sql("SELECT * FROM activas WHERE fecha = '" + self.fecha + "'").withColumnRenamed("saldo", "valor_final").withColumn("valor_inicial", F.lit(0))
        finMesAnteriorActivas = self.spark.sql("SELECT * FROM activas WHERE fecha = '" + str(fechaFinMesAnterior) + "'").withColumnRenamed("saldo", "valor_inicial").withColumn("valor_final",
F.lit(0)).withColumn("fecha", F.lit(self.fecha))
        fechaFinAnioAnteriorActivas = self.spark.sql("SELECT * FROM activas WHERE fecha = '" + str(fechaFinAnioAnterior) + "'").withColumnRenamed("saldo", "valor_inicial").withColumn("valor_final",
F.lit(0)).withColumn("fecha", F.lit(self.fecha))
        fechaFinMesAnioAnteriorActivas = self.spark.sql("SELECT * FROM activas WHERE fecha = '" + str(fechaFinMesAnioAnterior) + "'").withColumnRenamed("saldo", "valor_inicial").withColumn("valor_final",
F.lit(0)).withColumn("fecha", F.lit(self.fecha))

        actualActivas.select("fecha", "tipo_institucion", "nombre_corto", "nombre_largo", "producto", "segmento", "segmento_cartera", "valor_inicial", "valor_final") \
            .union(finMesAnteriorActivas.select("fecha", "tipo_institucion", "nombre_corto", "nombre_largo", "producto", "segmento", "segmento_cartera", "valor_inicial", "valor_final")) \
            .groupBy("fecha", "tipo_institucion", "nombre_corto", "nombre_largo", "producto", "segmento", "segmento_cartera") \
            .agg(F.sum("valor_inicial").alias("valor_inicial"), F.sum("valor_final").alias("valor_final")) \
            .withColumn("tipo", F.lit("MENSUAL")).createOrReplaceTempView("variacionesActivasMensual")

        actualActivas.select("fecha", "tipo_institucion", "nombre_corto", "nombre_largo", "producto", "segmento", "segmento_cartera", "valor_inicial", "valor_final") \
            .union(fechaFinAnioAnteriorActivas.select("fecha", "tipo_institucion", "nombre_corto", "nombre_largo", "producto", "segmento", "segmento_cartera", "valor_inicial", "valor_final")) \
            .groupBy("fecha", "tipo_institucion", "nombre_corto", "nombre_largo", "producto", "segmento", "segmento_cartera") \
            .agg(F.sum("valor_inicial").alias("valor_inicial"), F.sum("valor_final").alias("valor_final")) \
            .withColumn("tipo", F.lit("ACUMULADA")).createOrReplaceTempView("variacionesActivasAcumulada")

        actualActivas.select("fecha", "tipo_institucion", "nombre_corto", "nombre_largo", "producto", "segmento", "segmento_cartera", "valor_inicial", "valor_final") \
            .union(fechaFinMesAnioAnteriorActivas.select("fecha", "tipo_institucion", "nombre_corto", "nombre_largo", "producto", "segmento", "segmento_cartera", "valor_inicial", "valor_final")) \
            .groupBy("fecha", "tipo_institucion", "nombre_corto", "nombre_largo", "producto", "segmento", "segmento_cartera") \
            .agg(F.sum("valor_inicial").alias("valor_inicial"), F.sum("valor_final").alias("valor_final")) \
            .withColumn("tipo", F.lit("ANUAL")).createOrReplaceTempView("variacionesActivasAnual")

        self.spark.sql("SELECT * FROM variacionesPasivasMensual UNION ALL SELECT * FROM variacionesPasivasAcumulada UNION ALL SELECT * FROM variacionesPasivasAnual") \
            .write.format("com.mongodb.spark.sql.DefaultSource").mode("append").option("database", "repositorio").option("collection", "variacion_pasivas").save()

        self.spark.sql("SELECT * FROM variacionesActivasMensual UNION ALL SELECT * FROM variacionesActivasAcumulada UNION ALL SELECT * FROM variacionesActivasAnual") \
            .write.format("com.mongodb.spark.sql.DefaultSource").mode("append").option("database", "repositorio").option("collection", "variacion_activas").save()

    def last_day_of_month(any_day):
        next_month = any_day.replace(day=28) + datetime.timedelta(days=4)  # this will never fail
        return next_month - datetime.timedelta(days=next_month.day)
```

## Anexo V. Microservicio "administracion"

### Clase CredencialApi

Esta API es usada para autenticar las credenciales de un usuario y retornar los permisos, datos personales y módulos correspondientes, también es usada para poder crear credenciales de nuevos usuarios.

```java
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import net.unir.mongoDemo.api.dto.ApiResponse;
import net.unir.mongoDemo.api.dto.EstadoApiResponseEnum;
import net.unir.mongoDemo.modelo.Credencial;
import net.unir.mongoDemo.servicios.CredencialService;

@RestController
@RequestMapping(value= "/sig/administracion/credencial")
public class CredencialApi{

    private final Logger logger = LoggerFactory.getLogger(this.getClass());

    @Autowired
    CredencialService serv;

    @CrossOrigin("*")
    @GetMapping(value= "/autenticar/{credencial_id}/{password}")
    public ApiResponse autenticar(@PathVariable(value= "credencial_id") String _id, @PathVariable(value= "password") String password) {
        logger.debug("Autenticando...");
        return new ApiResponse(EstadoApiResponseEnum.OK.getCode(), serv.autenticar(_id, password));
    }

    @CrossOrigin("*")
    @PostMapping(value= "/crear")
    public ApiResponse crear(@RequestBody Credencial obj) {
        logger.debug("Creando...");
        return new ApiResponse(EstadoApiResponseEnum.OK.getCode(), serv.createCredencial(obj) );
    }

}
```

### Clase Institucion_FinancieraApi

Esta API es usada para realizar la consulta de todas las instituciones presentes en el sistema de información gerencial, este permitirá en la administración del sistema asignarle una institución a un usuario.

```java
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import net.unir.mongoDemo.api.dto.ApiResponse;
import net.unir.mongoDemo.api.dto.EstadoApiResponseEnum;
import net.unir.mongoDemo.servicios.Institucion_FinancieraService;

@RestController
@RequestMapping(value= "/sig/administracion/institucion_financiera")
public class Institucion_FinancieraApi{

    private final Logger logger = LoggerFactory.getLogger(this.getClass());

    @Autowired
    Institucion_FinancieraService serv;

    @CrossOrigin("*")
    @GetMapping(value= "/obtenerInstitucionesFinancieras")
    public ApiResponse getAll() {
        logger.debug("Obteniendo Instituciones...");
        return new ApiResponse(EstadoApiResponseEnum.OK.getCode(), serv.getAllInstitucion_Financiera());
    }
}
```

### Clase PersonaApi

Esta API permite obtener el listado de todos los usuarios existentes en el sistema con la finalidad de que el administrador pueda conocer que usuarios acceden al sistema, también se tiene la entrada para la creación de nuevos usuarios.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import net.unir.mongoDemo.api.dto.ApiResponse;
import net.unir.mongoDemo.api.dto.EstadoApiResponseEnum;
import net.unir.mongoDemo.modelo.Persona;
import net.unir.mongoDemo.servicios.PersonaService;

@RestController
@RequestMapping(value= "/sig/administracion/persona")
public class PersonaApi{

    private final Logger logger = LoggerFactory.getLogger(this.getClass());

    @Autowired
    PersonaService serv;

    @CrossOrigin("*")
    @GetMapping(value= "/obtenerPersonas")
    public ApiResponse getAll() {
        logger.debug("Obteniendo Personas...");
        return new ApiResponse(EstadoApiResponseEnum.OK.getCode(), serv.getAllPersona());
    }

    @CrossOrigin("*")
    @PostMapping(value= "/crear")
    public ApiResponse crear(@RequestBody Persona obj) {
        logger.debug("Creando...");
        return new ApiResponse(EstadoApiResponseEnum.OK.getCode(), serv.createPersona(obj) );
    }
}
```

### Clase RolApi

Esta API permite obtener el listado de todos los roles existentes en el sistema de información gerencial.

```java
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import net.unir.mongoDemo.api.dto.ApiResponse;
import net.unir.mongoDemo.api.dto.EstadoApiResponseEnum;
import net.unir.mongoDemo.servicios.RolService;

@RestController
@RequestMapping(value= "/sig/administracion/rol")
public class RolApi{

    private final Logger logger = LoggerFactory.getLogger(this.getClass());

    @Autowired
    RolService serv;

    @CrossOrigin("*")
    @GetMapping(value= "/obtenerRoles")
    public ApiResponse getAll() {
        logger.debug("Obteniendo Roles...");
        return new ApiResponse(EstadoApiResponseEnum.OK.getCode(), serv.getAllRol());
    }

}
```

Cada uno de estos API's deben conectarse a la base de datos de administración del sistema, para ello se tienen los respectivos CRUD de acceso a los datos, la configuración de la base de datos al tratarse de una aplicación Spring Boot se los realiza en el archivo **application.properties**.

```
# Application configuration.
server.port=9091

# Local mongodb configuration.
spring.data.mongodb.host=localhost
spring.data.mongodb.port=27017
spring.data.mongodb.database=administracion_sig

# Logging configuration.
logging.level.com.assignment.springboot.mongo=DEBUG
logging.pattern.console= %d{yyyy-MM-dd HH:mm:ss} - %msg%n
```

Como vemos se tiene las credenciales a la base de datos y el puerto en donde se ejecutará el servidor en donde se expondrán los servicios.

En las siguientes imágenes se tienen las clases de que realizan las operaciones de CRUD de la base de datos mongodb.

```java
import org.springframework.data.mongodb.repository.MongoRepository;
import org.springframework.stereotype.Repository;

import net.unir.mongoDemo.modelo.Credencial;

@Repository
public interface CredencialDao extends MongoRepository<Credencial, String> {

}
```

```java
import org.springframework.data.mongodb.repository.MongoRepository;
import org.springframework.stereotype.Repository;

import net.unir.mongoDemo.modelo.Institucion_Financiera;

@Repository
public interface Institucion_FinancieraDao extends MongoRepository<Institucion_Financiera, String> {

}
```

```java
import org.springframework.data.mongodb.repository.MongoRepository;
import org.springframework.stereotype.Repository;

import net.unir.mongoDemo.modelo.Modulo;

@Repository
public interface ModuloDao extends MongoRepository<Modulo, String> {

}
```

```java
import org.springframework.data.mongodb.repository.MongoRepository;
import org.springframework.stereotype.Repository;

import net.unir.mongoDemo.modelo.Persona;

@Repository
public interface PersonaDao extends MongoRepository<Persona, String> {

}
```

```java
import org.springframework.data.mongodb.repository.MongoRepository;
import org.springframework.stereotype.Repository;

import net.unir.mongoDemo.modelo.Rol;

@Repository
public interface RolDao extends MongoRepository<Rol, String> {

}
```

Como podemos ver en Spring Boot se definen las interfaces que extienden de la clase **MongoRepository** que indica a Spring que se conecta a una base de datos de tipo Mongo y contiene todas las operaciones típicas de un CRUD como son la creación, modificación,

eliminación, buscar todos los elementos o buscar por el identificador, adicionalmente se puede necesitar un CRUD distinto a los que se tienen en la clase, por esa razón se implementan nuevos servicios y se realizan las operaciones que sean necesarias, precisamente esto se muestra a continuación:

```java
@Override
public void createCredencial(List<Credencial> objs) {
    credencialDao.saveAll(objs);
}

@Override
public CredencialDto autenticar(String _id, String password) {

    CredencialDto res = new CredencialDto();
    try
    {
        Optional<Credencial> credencial = credencialDao.findById(_id);
        if (credencial.get().getPassword().equals(password)){
            res.set_id(_id);

            Optional<Persona> persona = personaDao.findById(credencial.get().getIdentificacion());
            res.setNombre(persona.get().getNombre());
            res.setApellido(persona.get().getApellido());
            res.setInstitucion(persona.get().getInstitucion());
            res.setEmail(persona.get().getEmail());
            res.setRol(credencial.get().getRol());

            Optional<Rol> rol = rolDao.findById(credencial.get().getRol());
            res.setTipo_institucion(rol.get().getTipo_institucion());

            Collection<Modulo> lstmodulos = new ArrayList<Modulo>();
            for (String modulo_id : rol.get().getModulos()) {
                Optional<Modulo> modulo = moduloDao.findById(modulo_id);
                lstmodulos.add(modulo.get());
            }
            res.setModulos(lstmodulos);
        }
    }catch(Exception e){
        System.out.println(e.getMessage());
    }

    return res;
}
```

```java
import java.util.Collection;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import net.unir.mongoDemo.dao.Institucion_FinancieraDao;
import net.unir.mongoDemo.modelo.Institucion_Financiera;
import net.unir.mongoDemo.servicios.Institucion_FinancieraService;

@Service
public class Institucion_FinancieraServiceImpl implements Institucion_FinancieraService{

    @Autowired
    Institucion_FinancieraDao institucion_FinancieraDao;

    @Override
    public Collection<Institucion_Financiera> getAllInstitucion_Financiera() {
        return institucion_FinancieraDao.findAll();
    }


}
```

```java
import java.util.Collection;
import java.util.Optional;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import net.unir.mongoDemo.dao.PersonaDao;
import net.unir.mongoDemo.modelo.Persona;
import net.unir.mongoDemo.servicios.PersonaService;

@Service
public class PersonaServiceImpl implements PersonaService {

    @Autowired
    PersonaDao personaDao;

    @Override
    public Collection<Persona> getAllPersona() {
        return personaDao.findAll();
    }

    @Override
    public Persona createPersona(Persona obj) {
        Optional<Persona> credencial = personaDao.findById(obj.get_id());
        if (credencial.isPresent())
            return null;
        return personaDao.save(obj);
    }


}
```

```java
import java.util.Collection;

import org.springframework.stereotype.Service;
import org.springframework.beans.factory.annotation.Autowired;

import net.unir.mongoDemo.dao.RolDao;
import net.unir.mongoDemo.modelo.Rol;
import net.unir.mongoDemo.servicios.RolService;

@Service
public class RolServiceImpl implements RolService {

    @Autowired
    RolDao rolDao;

    @Override
    public Collection<Rol> getAllRol() {
        return rolDao.findAll();
    }


}
```

## Anexo VI. Cuestionario de evaluación de usabilidad y funcionalidad



# Evaluación Sistema de Información Gerencial - SIG

Esta encuesta permite evaluar la usabilidad y funcionalidad del nuevo sistema para determinar si la nueva propuesta de actualizar la plataforma es necesaria.

*Obligatorio

1. **Me sentí seguro utilizando el sistema** *
   *Marca solo un óvalo.*

| | 1 | 2 | 3 | 4 | 5 | |
|---|---|---|---|---|---|---|
| En completo desacuerdo | ◯ | ◯ | ◯ | ◯ | ◯ | En completo acuerdo |

2. **No tuve que aprender otros temas para utilizar el sistema** *
   *Marca solo un óvalo.*

| | 1 | 2 | 3 | 4 | 5 | |
|---|---|---|---|---|---|---|
| En completo desacuerdo | ◯ | ◯ | ◯ | ◯ | ◯ | En completo acuerdo |

3. **El sistema me pareció fácil de usar** *
   *Marca solo un óvalo.*

| | 1 | 2 | 3 | 4 | 5 | |
|---|---|---|---|---|---|---|
| En completo desacuerdo | ◯ | ◯ | ◯ | ◯ | ◯ | En completo acuerdo |

4. **No necesité de soporte técnico para usar el sistema** *
   *Marca solo un óvalo.*

| | 1 | 2 | 3 | 4 | 5 | |
|---|---|---|---|---|---|---|
| En completo desacuerdo | ◯ | ◯ | ◯ | ◯ | ◯ | En completo acuerdo |

5. **Pienso que los demás usuarios aprenderán rápidamente a usar el sistema** *
   *Marca solo un óvalo.*

| | 1 | 2 | 3 | 4 | 5 | |
|---|---|---|---|---|---|---|
| En completo desacuerdo | ◯ | ◯ | ◯ | ◯ | ◯ | En completo acuerdo |

6. **Me gustaría usar el sistema de información gerencial frecuentemente** *
   *Marca solo un óvalo.*

| | 1 | 2 | 3 | 4 | 5 | |
|---|---|---|---|---|---|---|
| En completo desacuerdo | ◯ | ◯ | ◯ | ◯ | ◯ | En completo acuerdo |

7. **Me resulta fácil hacer que el sistema haga lo que necesito** *
*Marca solo un óvalo.*

| | 1 | 2 | 3 | 4 | 5 | |
|---|---|---|---|---|---|---|
| En completo desacuerdo | ◯ | ◯ | ◯ | ◯ | ◯ | En completo acuerdo |

8. **Las funcionalidades del sistema me parecieron correctamente integradas** *
*Marca solo un óvalo.*

| | 1 | 2 | 3 | 4 | 5 | |
|---|---|---|---|---|---|---|
| En completo desacuerdo | ◯ | ◯ | ◯ | ◯ | ◯ | En completo acuerdo |

9. **El sistema me ha ayudado a tomar decisiones en base a la información implementada** *
*Marca solo un óvalo.*

| | 1 | 2 | 3 | 4 | 5 | |
|---|---|---|---|---|---|---|
| En completo desacuerdo | ◯ | ◯ | ◯ | ◯ | ◯ | En completo acuerdo |

10. **Pienso que no se requiere de demasiados pasos para realizar una tarea** *
*Marca solo un óvalo.*

| | 1 | 2 | 3 | 4 | 5 | |
|---|---|---|---|---|---|---|
| En completo desacuerdo | ◯ | ◯ | ◯ | ◯ | ◯ | En completo acuerdo |

11. **Recomiendo continuar con la actualización del software** *
*Marca solo un óvalo.*

| | 1 | 2 | 3 | 4 | 5 | |
|---|---|---|---|---|---|---|
| En completo desacuerdo | ◯ | ◯ | ◯ | ◯ | ◯ | En completo acuerdo |

| | FACILIDAD | | | | | USABILIDAD | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Persona | P1 | P2 | P3 | P4 | P5 | P6 | P7 | P8 | P9 | P10 | P11 | Promedio Facilidad | Promedio Usabilidad | Promedio Total |
| Persona 1 | 5 | 4 | 5 | 5 | 5 | 4 | 5 | 5 | 4 | 4 | 5 | **4.8** | **4.4** | **4.6** |
| Persona 2 | 5 | 4 | 4 | 4 | 5 | 5 | 4 | 4 | 5 | 5 | 5 | **4.4** | **4.6** | **4.5** |
| Persona 3 | 5 | 5 | 5 | 4 | 4 | 5 | 4 | 4 | 5 | 5 | 5 | **4.6** | **4.6** | **4.6** |
| Persona 4 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | **5.0** | **5.0** | **5.0** |
| Persona 5 | 5 | 3 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 4 | 5 | **4.6** | **4.8** | **4.7** |
| Persona 6 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | **5.0** | **5.0** | **5.0** |
| Persona 7 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | **5.0** | **5.0** | **5.0** |
| Persona 8 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | **5.0** | **5.0** | **5.0** |
| Persona 9 | 5 | 3 | 5 | 4 | 4 | 5 | 5 | 5 | 5 | 5 | 5 | **4.2** | **5.0** | **4.6** |
| Persona 10 | 5 | 3 | 5 | 3 | 5 | 4 | 5 | 5 | 5 | 5 | 5 | **4.2** | **4.8** | **4.5** |
| **TOTAL** | **5.0** | **4.2** | **4.9** | **4.5** | **4.8** | **4.8** | **4.8** | **4.8** | **4.9** | **4.8** | **5.0** | **4.7** | **4.8** | **4.8** |

Figure 80: Tabla de puntuaciones global.

| Nº | Preguntas | 1 En completo desacuerdo | 2 | 3 | 4 | 5 En completo acuerdo | TOTAL |
|---|---|---|---|---|---|---|---|
| 1 | Me sentí seguro utilizando el sistema | 0 | 0 | 0 | 0 | 10 | 10 |
| 2 | No tuve que aprender otros temas para utilizar el sistema | 0 | 0 | 3 | 2 | 5 | 10 |
| 3 | El sistema me pareció fácil de usar | 0 | 0 | 0 | 1 | 9 | 10 |
| 4 | No necesité de soporte técnico para usar el sistema | 0 | 0 | 1 | 3 | 6 | 10 |
| 5 | Pienso que los demás usuarios aprenderán rápidamente a usar el sistema | 0 | 0 | 0 | 2 | 8 | 10 |
| 6 | Me gustaría usar el sistema de información gerencial frecuentemente | 0 | 0 | 0 | 2 | 8 | 10 |
| 7 | Me resulta fácil hacer que el sistema haga lo que necesito | 0 | 0 | 0 | 2 | 8 | 10 |
| 8 | Las funcionalidades del sistema me parecieron correctamente integradas | 0 | 0 | 0 | 2 | 8 | 10 |
| 9 | El sistema me ha ayudado a tomar decisiones en base a la información implementada | 0 | 0 | 0 | 1 | 9 | 10 |
| 10 | Pienso que no se requiere de demasiados pasos para realizar una tarea | 0 | 0 | 0 | 2 | 8 | 10 |
| 11 | Recomiendo continuar con la actualización del software | 0 | 0 | 0 | 0 | 0 | 10 |

Figure 81: Número de preguntas contestadas.