IMPLEMENTACIÓN DE PROTOTIPO DE SOFTWARE DE GESTIÓN DE INFORMACIÓN Y PROCESOS PARA OFICINA ASESORA DE SEGUROS BASADA EN TECNOLOGÍAS CLOUD

Presentado por:

DANILO ALEJANDRO ARIZA QUIROGA
HÉCTOR MANUEL RODRÍGUEZ MARTIN

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS
FACULTAD DE INGENIERÍA
INGENIERÍA DE SISTEMAS
BOGOTÁ D.C.
2016

IMPLEMENTACIÓN DE PROTOTIPO DE SOFTWARE DE GESTIÓN DE INFORMACIÓN Y PROCESOS PARA OFICINA ASESORA DE SEGUROS BASADA EN TECNOLOGÍAS CLOUD

Presentado Por:

DANILO ALEJANDRO ARIZA QUIROGA cod. 20092020006
HÉCTOR MANUEL RODRÍGUEZ MARTIN cod. 20092020074

Proyecto de Grado

JHON FREDDY PARRA PEÑA
Director

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS
FACULTAD DE INGENIERÍA
INGENIERÍA DE SISTEMAS
BOGOTÁ D.C.
2016

CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

INTRODUCCIÓN

En el presente proyecto se aborda la importancia del aprovechamiento de las tecnologías en la nube (cloud) por parte de las empresas, grandes o pequeñas, para apalancar su actividad económica; se tiene en cuenta el uso de dichas tecnologías desde etapas de diseño y desarrollo, debido a las facilidades que estas presentan.

Entre las facilidades para las empresas, está la delegación al proveedor de servicios, del mantenimiento de infraestructura tecnológica y para los desarrolladores, facilidades para trabajo colaborativo, pruebas y despliegue de desarrollos.

Se tiene en cuenta, como forma de trabajo, el uso de metodologías ágiles; que más que ser metodologías, son vistas como una forma de pensar las cuales buscan orientar el desarrollo más al sentir del cliente que a la solución en sí de forma técnica.

El trabajo en torno a estos temas se desarrolla por interés de los integrantes en profundizar formas de trabajo diferentes a las tradicionales, por la importancia de identificar y explotar las ventajas que cada uno de ellas ofrece. En el ámbito profesional, es de gran importancia el conocimiento de diversas formas de trabajo, ya que grandes empresas a nivel mundial están adaptándolas; además, un profesional con estos conocimientos, se puede adaptar fácilmente cualquier entorno de trabajo.

El presente proyecto presenta una propuesta de aplicación de los temas mencionados anteriormente a un negocio específico (Nubia Pinto Asesores de seguros e inversiones), con el fin de facilitar y mejorar la forma en la que actualmente se desarrolla su actividad económica, además de permitir a futuro su crecimiento sin necesidad de preocuparse por la adaptabilidad de la solución.

1. PLANTEAMIENTO DEL PROBLEMA

1.1. DESCRIPCIÓN DEL PROBLEMA

El manejo de información de los clientes en un negocio es el fundamental para mantener, controlar y expandir el mismo. Cuando se tienen diferentes servicios específicamente: venta y control de productos como: SOAT, pólizas de seguro contra todo riesgo y pólizas de cumplimiento; son varios los factores y condiciones que se deben controlar al mismo tiempo; a continuación, se describe la forma en la que se desarrolla la actividad económica de la oficina Nubia Pinto Asesores de seguros e inversiones**("Nubia Pinto - Asesores de seguros e inversiones," n.d.)**, la cual es caso de estudio en el presente proyecto.

La oficina presta servicios de asesoramiento y venta de pólizas de seguros de diferentes tipos (SOAT, pólizas de cumplimiento, seguro contra todo riesgo); los cuales son gestionados directamente en su ubicación física. Para cada uno de los anteriores existe un comportamiento diferente en cuanto a vigencias, formas de pago, trámite requerido, entre otros.

El control de los productos y servicios vendidos, la vigencia de cada uno de ellos, el número de cuotas restantes, se realiza mediante hojas de cálculo, en las cuales se registra la información básica, tanto de los clientes, como de los productos adquiridos. Por lo general, dicho registro se lleva en archivos separados para cada tipo de producto y servicio. En cuanto al control de los productos y servicios que se encuentran en proceso de tramitación, este se realiza, al igual que lo descrito anteriormente, mediante hojas de cálculo, en las cuales se lleva registro de la información necesaria del cliente; además, cuando el proceso de tramitación requiere de documentación adicional, ya sea física o digital, en las hojas se registra la entrega o falta de la misma; dicha documentación se almacena en folios, por aparte. Algunos de los productos o servicios requieren, para su adquisición, que el cliente se acerque en más de una ocasión durante el proceso.

En cuanto a la información de los productos y servicios ofrecidos, esta se entrega únicamente en la oficina. Los procesos de consulta de los estados de los productos por usuario se realizan de forma manual en cada una de las hojas de cálculo. Dado que la actividad principal del negocio es prestar la asesoría y facilitar el proceso de adquisición de los productos y servicios ofrecidos por diversas compañías aseguradoras, es necesario llevar un control de cada uno de los dineros que es recogido teniendo en cuenta cada una de las compañías proveedoras de los mismos; está tareas (control de dineros por compañía y consulta de estados de productos por usuarios) se vuelven tediosas debido a la falta de un sistema centralizado que permita la gestión de la información.

Al manejar variedad de productos y servicios, la empresa recolecta una cantidad significativa de información de clientes y del manejo interno del negocio, pero no cuenta con una infraestructura tecnológica propia por el costo y la complejidad que esta conlleva.

Al no ser una empresa especializada en el sector tecnológico, los recursos de hardware y software no son prioridad para la ejecución diaria de la actividad económica, pero este hecho estanca la mejora de los servicios prestados y el crecimiento en general en todas las áreas de la empresa.

1.2. FORMULACIÓN DEL PROBLEMA

¿Cómo proveer una solución de bajo costo que permita un desarrollo óptimo de las actividades de la oficina, facilitando un manejo eficiente de la información y que además pueda integrar soluciones enfocadas en los clientes finales de forma rápida y sin cambios drásticos y costosos?

2. OBJETIVOS

2.1. OBJETIVO GENERAL

Implementar una solución de software prototipo que centralice la información y los procesos realizados por parte de la oficina utilizando tecnologías en la nube, permitiéndole desarrollar de forma eficiente su actividad económica.

2.2. OBJETIVOS ESPECÍFICOS

- Desarrollar una solución de software basada en cloud computing implementando la metodología SCRUM, que permitan al cliente (oficina) involucrarse de forma más directa con el desarrollo de la solución y prevengan de forma oportuna posibles desvíos en cuanto al objetivo durante el proceso.

- Establecer las características de los diferentes escenarios que se presentan durante el desarrollo de la actividad económica por parte de los empleados de la oficina, con el fin de realizar un modelamiento correcto de estos en la plataforma.

- Desarrollar una aplicación móvil enfocada en los clientes finales, que permita establecer un canal de comunicación alterno con la oficina, mediante el cual podrá realizar consultas de sus productos, cotizaciones y adquisiciones de nuevos productos.

- Diseñar un modelo de datos que permita, a futuro, manejar la información correspondiente a nuevas actividades desarrolladas por parte de la oficina.

- Presentar un esquema de costos de operación de la solución en cuanto a los entornos de ejecución y servicios teniendo en cuenta el flujo estimado de la información durante el desarrollo de la actividad económica de la oficina.

3. JUSTIFICACIÓN

En la actualidad, proveer y operar servicios digitales disponibles en todo momento es materia esencial para todo negocio, sea una gran compañía u oficinas pequeñas. Con la expansión acelerada de tecnologías y servicios como Web 2.0, cloud computing, redes sociales, Smartphones, entre otras; los usuarios cada vez buscan suplir sus necesidades, sea cual sea, a través de estos servicios; por otro lado, las empresas buscan facilitar canales a sus clientes, por estos medios, de tal forma que estos puedan acceder a los productos y servicios ofrecidos.

En el 2015, el estudio anual "Global Contact Center Survey Results" realizado por Deloitte **(Deloitte Consulting LLP, 2015)** reveló que los canales propensos a experimentar mayor crecimiento para consultas sencillas son web, email y móvil presentando un 83%, 80%, y 77% de crecimiento respectivamente. Es de gran importancia que todo negocio enfoque la atención a sus usuarios, a cualquier nivel, en los medios digitales.

Tomando el caso de empresas pequeños, se suele escuchar que no se realiza inversión para implementar una plataforma digital, aunque esta sea vital en el desarrollo de los procesos de estas, debido al alto costo que conlleva la adquisición de hardware, para montar una infraestructura tecnológica, y de licencias de software para operar.

En marzo de 2015, la revista Forbes **(Kepes, 2015)** publicó un artículo en el cual se menciona que la nube (cloud), dada la visión y el progreso que tienen las empresas hacia ella, se está convirtiendo en la forma por defecto de ejecutar las aplicaciones. En el artículo se presenta algunas estadísticas del estudio realizado, entre las cuales se encuentra que el 93% de las empresas entrevistadas está ejecutando aplicaciones en la nube o están experimentando con IaaS (Infrastructure as a service). En cuanto a las oportunidades de expansión a través de la nube, el estudio encontró que el 68% de las empresas encuestadas están ejecutando al menos la quinta parte de su portafolio de aplicaciones en la nube; en cuanto a la totalidad del portafolio de aplicaciones, el 55% de las empresas manifiesta que gran parte de él no se encuentra en la nube, pero que está construido con arquitecturas amigables con la nube (cloud-friendly).

El uso de cualquiera, o de los 3 pilares del cloud computing por parte de cualquier empresa o negocio (IaaS, PaaS, SaaS), mejorará en gran medida la gestión de sus procesos internos, al delegar en el proveedor la responsabilidad de gestionar sus recursos de IT, con esto permite que la empresa se enfoque de mejor manera en su actividad económica; además mejorará la relación con sus clientes al proveer diversos canales de comunicación, facilitándoles la interacción con ellas.

4. MARCO TEÓRICO

4.1. MARCO REFERENCIAL

Últimamente se escucha hablar mucho de un concepto bastante característico de la época, “Cloud” (nube); pero, cuales son los orígenes de dicho concepto y porque ha tenido tanta trascendencia desde su aparición. En general, cuando se habla de cloud computing (computación en la nube) se suele pensar que es un concepto reciente, muchas veces atribuido al siglo XXI y al impacto y evolución en el ámbito tecnológico presente desde inicios de este.

INICIOS CLOUD COMPUTING

Las primeras nociones de este concepto se le atribuye al matemático y científico computacional estadounidense John McCarthy quien en los años 60 concibió la idea de la computación de tiempo compartido o en red, la cual permitiera a los usuarios compartir información, mediante la vinculación a una computadora central, lo cual en última instancia reduciría los costos de uso de las computadoras **(Childs, 2011)**; sin embargo, debido a que las capacidades de software y de hardware de la época eran muy limitadas, la popularidad de esta noción se desvaneció de a poco.

En los años 70, con la aparición del concepto de virtualización y su aplicación, las máquinas virtuales (VMs), las cuales permiten la ejecución de uno o varios sistemas operativos de forma simultánea dentro de un ambiente aislado **(Neto, 2014)**; la noción de poder computacional compartido volvió a tomar fuerza y llevó a la transformación de la computadora central (Mainframe, 1950) permitiéndole alojar diversos entornos computacionales independientes dentro de un único entorno físico; a su vez, proporcionó mejoras en cuanto al rendimiento de las aplicaciones ya que al ejecutarse en entornos aislados, tenían acceso exclusivo a los recursos del sistema **(Rose, 2004)**. IBM desarrolló este concepto como una forma de aplicar tiempo compartido sobre mainframe costosos. IBM define a una máquina virtual como una copia completamente protegida y aislada del hardware de la máquina subyacente **(Creasy, 1981)**.

Para los años 90, y gracias a la irrupción del sistema operativo Linux y sus estándares abiertos, se facilitó la implementación de *clusters*, conjuntos de servidores que, con la ayuda de conexiones de alta velocidad, trabajan en conjunto como si fueran una sola computadora. “Estos clusters sufrieron un proceso de especialización para proporcionar servicios de cálculo y almacenamiento, fundamentalmente en centros de investigación y universidades. Estos centros comenzaron a ofrecer sus servicios a terceros a través de protocolos estándar, constituyendo la denominada arquitectura de computación *grid*, orientada al procesamiento en paralelo o al almacenamiento de gran cantidad de información” **(Urueña, Ferrari, Blanco, & Valdecasa, 2010)**.

Esta arquitectura tuvo poca popularidad más allá de los centros de investigación debido a que presentaba problemas de integración con dispositivos heterogéneos, portabilidad entre diferentes sistemas "*grid*", mantenimiento complejo al igual que la utilización de la infraestructura. Debido a esto, las máquinas virtuales tomaron fuerza al ver que con ellas se podía subsanar los problemas mencionados. El uso de esta arquitectura (granjas de servidores ejecutando máquinas virtuales de diferentes especificaciones) tomó mucha fuerza y es la que actualmente le da vida a la computación en la nube.

En la actualidad grandes empresas líderes en servicios de cloud computing como Google, IBM, Amazon, se basan en esta arquitectura para prestar sus servicios.

IMPLICACIONES MANUTENCIÓN INFRAESTRUCTURA PROPIA

La preferencia principal de la mayoría de las empresas, de cualquier tamaño, es tener el control total (físico y operacional) de la información trabajada en su actividad económica, la primera opción es poseer una infraestructura propia y adquisición de licencias de software para apalancar el desarrollo de su actividad, pero en muchas ocasiones, en especial en negocios pequeños, la adquisición y mantenimiento de una infraestructura genera costos demasiado elevados, ya que se debe tener en cuenta factores como cantidad de procesadores, capacidad en memoria RAM, fuentes de poder (principales y de respaldo), capacidad en discos duros (principales y de respaldo), sistemas de refrigeración, infraestructura de red, adicionalmente y dependiendo del tamaño de la empresa, se hace necesario la inclusión de personal que realice soporte a dicha infraestructura, lo que se conoce como gastos operativos. Por tal razón, en ocasiones las empresas prefieren desligarse del compromiso monetario y operacional que acarrea dicha infraestructura y enfocar sus esfuerzos en la esencia de sus negocios. una de las opciones de mayor beneficio son los servicios cloud.

PILARES DEL CLOUD COMPUTING

Se entiende como cloud a una infraestructura IT, diseñada con el propósito de proveer, de forma remota, recursos tecnológicos escalables y moderados originalmente el término era metafóricamente usado para describir a internet, la cual en esencia es una red de redes. Previamente a que cloud se convirtiera en una industria IT formalizada, este término era utilizado para representar a internet a través de múltiples especificaciones **(WhatIsCloud.com, n.d.)**. En la actualidad hace referencia a aquellos servicios prestados por compañías a través de internet.

El ideal principal que apoya la computación en la nube es Xaas (anything as a service) o todo como servicio el cual se refiere a la entrega de servicios a través de la red; es decir, proveer a sus clientes en cualquier parte del mundo una solución en la cual no requieran el uso de computadores extremadamente poderosos para ejecutar tareas de gran procesamiento, esas tareas se le delegan a los servidores del proveedor, dejando al cliente la única necesidad de poseer una conexión a internet para acceder a sus servicios mediante el uso de rentas (se paga por lo que se usa).
Al delegar toda la parte de procesamiento, mantenimiento y almacenamiento al proveedor, el cliente tiene una reducción de costos considerable ya que se libera del mantenimiento de infraestructura, adquisición de licencias, entre otros; beneficiando tanto

a grandes compañías como usuarios pequeños al proveer soluciones de infraestructura, plataforma y software a la medida de sus necesidades.

COMPONENTES

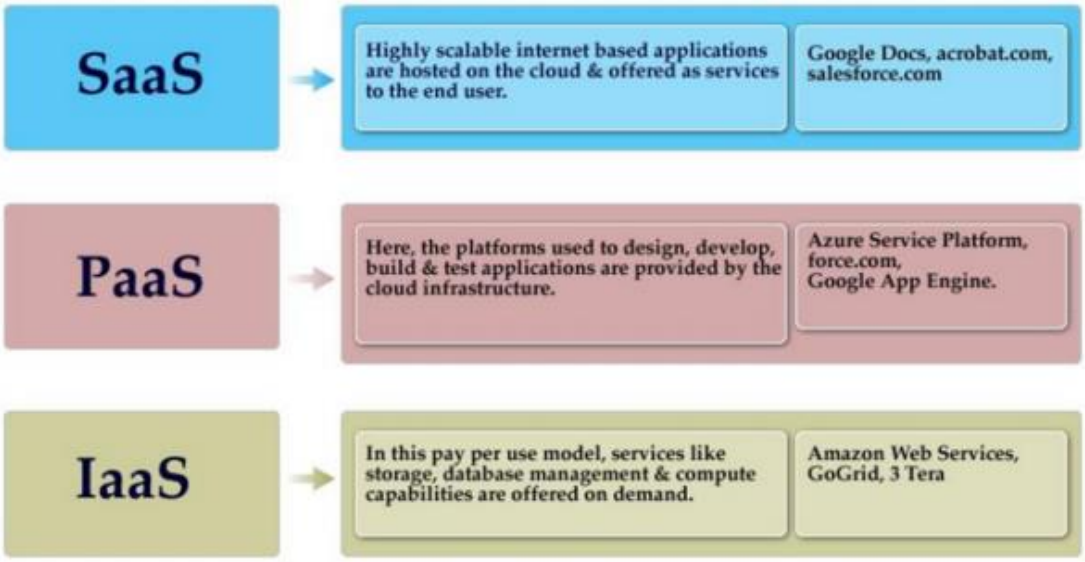

Figura 1. Cloud Models **(Wu et al., 2009)**.

Los 3 principales componentes en los que se basa la computación en la nube en la actualidad son:

- IaaS (Infrastructure as a service) infraestructura como servicio: Principalmente busca poner a disposición del cliente una infraestructura informática (capacidad de procesamiento, espacio en disco, bases de datos, etc) como servicios estandarizados en la red; es decir, los clientes en lugar de adquirir infraestructura propia, acceden a la infraestructura que es puesta en disposición como servicio por demanda por parte de los proveedores. Las principales características de IaaS son **(Kepes, 2013)**:

    - Distribución de recursos como servicio
    - Permite escalamiento dinámico
    - Costo variable (pago por uso)
    - Por lo general incluye a múltiples usuarios en una misma pieza de hardware

- PaaS (Platform as a service) plataforma como servicio: funciona como una capa de software o entorno de desarrollo que se encapsula y se ofrece como servicio. en esta, el cliente tiene la libertad de construir sus propias aplicaciones, las cuales se ejecutarán en la infraestructura del proveedor **(Wu et al., 2009)**. En cuanto a los

desarrolladores, es un conjunto de servicios dirigido que les ayuda a desarrollar y aplicaciones de prueba sin tener que preocuparse por la infraestructura subyacente. Proporciona facilidades a los desarrolladores en cuanto a escribir código, probar la aplicación, iniciar la aplicación, y ser capaz de hacer cambios en él continuamente para corregir errores **(Butler, 2013)**. Las principales características de PaaS son **(Kepes, 2013)**:

- Servicios para desarrollar probar desplegar alojar y mantener aplicaciones del cliente en el mismo entorno de desarrollo integrado
- Arquitectura de múltiples anfitriones, donde múltiples usuarios concurrentes utilizan la misma aplicación de desarrollo
- incorpora escalabilidad al software desplegado teniendo en cuenta aspectos como balanceo de carga y conmutación por error (failover).
- Integración con servicios web y bases de datos mediante estándares comunes
- Soporte para colaboración de equipos de desarrollo; algunas soluciones incluyen planeación del proyecto y herramientas de comunicación.

- SaaS (Software as a service) software como servicio: Es un modelo de entrega de aplicaciones por demanda por medio del cual se le proporciona a los usuarios, soluciones de software a través de internet. el modelo de ingresos de SaaS por lo general es basado en suscripción **(Amazon Web Services, 2010)**; con SaaS, múltiples usuarios pueden estar accediendo a la misma instancia del software al mismo tiempo. Las principales características de este modelo son:

  - Pago por uso mas no por licencia de software
  - Por parte del usuario no es necesario aplicar parches mejoras o actualizaciones al software, de esto se encarga el proveedor de servicios.
  - Acceso web a las aplicaciones ofrecidas en cualquier parte

TIPOS DE NUBES

Existen 3 tipos de nubes; públicas, privadas e híbridas, las cuales se escogen por sus clientes de acuerdo a sus necesidades de negocio. Las siguientes definiciones son tomadas del sitio de IBM Cloud **(IBM Cloud, n.d.)**.

- Nube pública: Son propiedad y además son operadas por las empresas encargadas de ofrecer, mediante un rápido acceso a través de una red pública, acceso a recursos informáticos asequibles. Con los servicios de nube pública, los usuarios no tienen que comprar hardware, software o infraestructura de apoyo, ya que de esto se encargan los proveedores.

- Nube privada: Hace referencia a la infraestructura que, operada exclusivamente para una sola organización, ya sea que se gestione internamente por la misma o por un tercero, en cuanto a su alojamiento, puede ser interno o externo. Las nubes privadas pueden aprovechar las ventajas de la nube, mientras que proporciona un

mayor control de los recursos y una clara dirección en cuanto a múltiples arrendamientos.

- Nube híbrida: Hace referencia a la infraestructura que utiliza una base de nube privada combinada con la integración estratégica y el uso de los servicios de nube pública. La realidad es que una nube privada no puede existir de forma aislada al resto de los recursos de TI de una empresa y la nube pública. La mayoría de las empresas con nubes privadas de una u otra forma evolucionarán para gestionar cargas de trabajo a través de centros de datos (data centers); la implementación de nubes en esta forma es lo que le da vida al concepto de nubes híbridas. es de gran utilidad para empresas que quieren mantener un control más estricto de su información sensible pero que a la vez quieren apalancar su actividad económica mediante el uso de los servicios ofrecidos por las nubes públicas.

PROVEEDORES DE PAAS

Actualmente existe una gran cantidad de proveedores de servicios en el modelo PaaS, entre ellos se encuentran App Engine (Google), Amazon Elastic Beanstalk (Amazon) y Bluemix (IBM); refiriéndonos a los sitios de cada una de estas plataformas, podemos notar cierta similitud en cuanto a escalamiento, entornos de ejecución, modelo de despliegue, entre otras. A continuación, se muestra una tabla comparativa entre las 3 plataformas basada en las métricas utilizadas por el proyecto *paas-profiles* **(Stefan Kolb, 2013)** y complementada con la información encontrada en los sitios de cada proveedor

Tabla 1. Comparativa proveedores de PaaS.

ÍTEMS	**Google App Engine**	**IBM Bluemix**	**Amazon Elastic Beanstalk**
Hosting	Público	Público, privado, híbrido	Público
Escalamiento	Horizontal, vertical, automatizado	Horizontal, vertical, automatizado	Horizontal, vertical, automatizado
Estado	En producción	En producción	En producción
Runtimes	Go, Java, PHP, Python	Go, Java, Node, PHP, .NET, Ruby, Python, XSP, (Posibilidades de extender)	Go, Java, Node, PHP, .NET, Ruby, Python
Frameworks	Django, Webapp2	Rails, Sinatra	
# Servicios y Add-ons	174	104	0

Infraestructura	Europa, Norte América	Reino Unido, Norteamérica, Australia	*Singapore* Japon Korea del sur Irlanda Estados Unidos (California, Virginia, Oregon) Australia Brasil
DevOps Services	Si	Si	Si

El presente proyecto se desarrollará sobre la plataforma Bluemix de IBM la cual cuenta, entre sus DevOps services con Rational Team Concert, una herramienta de seguimiento y control del trabajo realizado bajo la metodología SCRUM.

Se dispone, para el desarrollo de todo el proyecto, de una cuenta de prueba extendida en esta plataforma.

BASES DE DATOS NO SQL

Con la gran acogida que han tenido la computación en la nube y las tecnologías web 2.0 muchas tecnologías y formas de trabajo se han visto obligadas a cambiar, al igual que otras han surgido a causa de esto; entre ellos se encuentran los sistemas de gestión de bases de datos (DBMS). Debido a la gran cantidad y al crecimiento acelerado de la información de diversos tipos, estructurada, semi-estructurada y sin estructura, alrededor del mundo, fenómeno al cual se le conoce como *Big Data*. Dada esta situación se presenta la necesidad de sistemas de almacenamiento que puedan proveer un modelo fácilmente programable y además que sean altamente escalables, con el fin de poder soportar, procesar y almacenar de forma eficiente dicha información.

Los sistemas de gestión de bases de datos relacionales (RDBMS) fueron diseñados en una era en la que el hardware disponible, al igual que los requerimientos de almacenamiento y procesamiento eran muy diferentes a los presentados en la actualidad, por lo tanto, estas soluciones han tenido bastantes retos en cuanto a los requerimientos de rendimiento y escalamiento necesarios para *Big Data* **(Grolinger, Higashino, Tiwari, & Capretz, 2013)**.

Google, Amazon, Facebook y LinkedIn fueron de las primeras compañías en percatarse de esta situación y al no encontrar alternativas comerciales en el momento, se encargaron de plantear sus propias aproximaciones; el trabajo realizado por estas compañías, generó gran interés en la industria ya que muchas otras empresas estaban teniendo problemas por la misma situación. De aquí surge una de las alternativas existentes en la actualidad, las bases de datos No SQL **(Couchbase, 2014)**.

No SQL (No sólo SQL) hace referencia a aquellos sistemas de gestión de bases de datos que, por lo general, cuya forma para acceder, almacenar y gestionar los datos, no utiliza el lenguaje SQL.

En la actualidad existen alrededor de 150 sistemas gestores de base de datos no relacionales. Al evaluar sistemas No SQL y otros, e común basarse en el teorema CAP. (Consistency, Availability and Partition tolerance) para clasificarlos; según este teorema, ningún sistema distribuido puede cumplir a la vez con estas propiedades **(Gilbert & Lynch, 2012)**:

- Consistency (Consistencia): Hace referencia a que todos los nodos del sistema puedan ver la misma información al mismo tiempo
- Availability (Disponibilidad): Hace referencia a garantizar que todas las peticiones realizadas al sistema tengan como retorno respuesta de éxito o fracaso de la misma.
- Partition tolerance (Tolerancia a particionado): Hace referencia a que el sistema continúe su funcionamiento normal a pesar haber sido particionado o dividido por fallas en la red.

Por lo general, este teorema es mal entendido como una simple decisión en tiempo de diseño de sistema en la que se deba "escoger 2 de las 3 propiedades para ser cumplidas"; cuando en realidad presenta un marco descriptivo en cuanto a la capacidad de los sistemas de compensarse en tiempo de ejecución con el fin de adaptarse a los requerimientos de la situación presente **(Burd, 2011)**.

Existen 4 tipos de bases de datos No SQL **(Sullivan, 2015)**:

- Orientadas a documentos: Consideradas los sistemas más versátiles debido a que tienen aplicabilidad en la gran cantidad de proyectos y sistemas que tradicionalmente usan un sistema relacional. Gestionan datos semiestructurados; los formatos más utilizados para el almacenamiento de los datos son XML, JSON y BSON.
- Orientadas a columnas: Son sistemas pensados para realizar consultas y agregaciones sobre grandes cantidades de datos; mientras los sistemas relacionales guardan registros, estos guardan columnas.
- Clave valor: Guarda tuplas (clave, valor) de datos. Los datos son recuperados por medio de la clave.
- Grafos: Se basan en la teoría de grafos para la representación de los datos almacenados (nodos y aristas); de gran utilidad en modelos con muchas relaciones (redes, conexiones sociales, etc).

La elección del sistema No SQL depende de las necesidades del usuario.

NODEJS

Node.js es un entorno e intérprete JavaScript de lado de servidor que utiliza un modelo asíncrono y dirigido por eventos. Su meta es permitir a un programador construir

aplicaciones altamente escalables y escribir código que maneje decenas de miles de conexiones simultáneas en una sólo una máquina física **(Nodejs, n.d.)**.

Node usa el motor de JavaScript V8 de Google: El motor V8 JavaScript es el motor JavaScript subyacente que Google usa con su navegador Chrome. Un motor JavaScript en realidad interpreta el código y lo ejecuta. Con el V8, Google creó un intérprete ultra-rápido escrito en C++, con otro aspecto único: se puede descargar el motor e incorporarlo a cualquier aplicación que desee. No está restringido a ejecutarse en un navegador. Así, Node en realidad usa el motor V8 JavaScript escrito por Google y le da otro propósito para usarlo en el servidor. Node soporta protocolos TCP, DNS y HTTP **(Abernethy, 2011)**.

PROGRAMACIÓN ORIENTADA POR EVENTOS (Abernethy, 2011)

Node utiliza lo que se conoce como modelo de programación orientado por eventos. JavaScript es un gran lenguaje para programación orientada por eventos, porque permite funciones y cierres anónimos, y más importante, la sintaxis es similar para casi cualquier persona que haya codificado. Las funciones de devolución de llamado que se llaman cuando ocurre un evento pueden escribirse en el mismo punto en el que usted captura el evento. Fácil de codificar, fácil de mantener. No hay infraestructuras complicadas Orientadas a Objeto, no hay interfaces, no hay potencial para sobre-arquitectura de nada. Simplemente esperar por un evento, escribir una función de devolución de llamado.

Node está extremadamente bien diseñado para situaciones en que se esté esperando una gran cantidad de tráfico y donde la lógica del lado del servidor y el procesamiento requeridos, no sean necesariamente grandes antes de responder al cliente.

MÓDULOS NODE (Abernethy, 2011)

Node posee dos herramientas importantes para el uso de módulos: Node Modules y Node Package Manager. Se puede expandir la funcionalidad de Node instalando módulos.

Como una muestra rápida de las posibilidades, estos incluyen un módulo para escribir páginas creadas dinámicamente (como PHP), un módulo para facilitar el trabajo con MySQL, un módulo para ayudar con WebSockets, y un módulo para asistir en el análisis de texto y de parámetros, entre docenas de módulos disponibles.

Node presenta el Node Package Module, que es una forma integrada de instalar y administrar los módulos Node que esté usando. Este maneja automáticamente dependencias, de manera que se puede tener la seguridad de que cualquier módulo que se desee instalar se instalará correctamente con todas sus partes necesarias. También sirve como una forma para publicar módulos propios en la comunidad Node, si se opta por vincularse y escribir su propio módulo. El NPM es entonces una forma fácil para expandir la funcionalidad de Node sin tener que preocuparse por des configurar su instalación Node.

ANGULARJS (Prol, 2014)

AngularJS es un framework Javascript basado en MVC (Modelo-Vista-Controlador) y que se centra en intentar dinamizar documentos HTML, DHTML (Dynamic HTML) y ejecutarse como aplicación de una sola página.

AngularJS cambia el enfoque de "dinamización" de documentos HTML estáticos mediante la vinculación de elementos de nuestro documento HTML con nuestro modelo de datos (data binding). De este modo, definimos un modelo de datos que se corresponderá con determinadas partes de nuestro HTML y, siempre que haya cambios en una parte, automáticamente se verán reflejados en la otra, con independencia del árbol DOM de JavaScript.

LOS BLOQUES DE ANGULARJS (Prol, 2014)

Las aplicaciones AngularJS se organizan en: Modelos, Vistas, Controladores y, opcionalmente, Servicios.

Los modelos son objetos que representan los datos a los cuales puede acceder la aplicación y también se usan para representar el estado actual de la aplicación. Los controladores definen el comportamiento de la aplicación y conecta los modelos con las vistas.

Los servicios son objetos especializados que realizan el trabajo en nombre de otros objetos. Se busca que sean lo más reusables posible.

¿QUÉ OFRECE ANGULARJS? (Green & Seshadri, 2013)

Client-side template: El sistema de plantillas en AngularJS es diferente del utilizado en otros *frameworks*. Por lo general es el servidor el encargado de mezclar la plantilla con los datos y devolver el resultado al navegador. En AngularJS el servidor proporciona los contenidos estáticos (plantillas) y la información que se va a representar (modelo) y es el cliente el encargado de mezclar la información del modelo con la plantilla para generar la vista.

Controllers: Los *controllers* son los encargados de inicializar y modificar la información que contienen los *scopes* en función de las necesidades de la aplicación. También podemos declarar funciones en el *scope* que se podrán utilizar más tarde o ser llamadas desde la vista.

Services: Los *services* son los encargados de comunicarse con el servidor para enviar y obtener información que después será tratada por los *controllers* para mostrarla en las vistas.

Esta parte es más compleja de explicar con un ejemplo, por el momento nos basta con saber que los *services* se pueden dividir en tres categorías: *services*, *factories* y *providers*.

Uno de los *services* incluidos en el *framework* es $resource, el cual nos permite encapsular la interacción con servicios RESTful sin tener que tratar directamente con las llamadas http.

Otros services interesantes que incluye AngularJS son $q y las llamadas *promises*. Mediante este mecanismo podemos realizar acciones asíncronas y devolver valores que puede que aún no hayan sido resueltos. Cuando la acción ha finalizado el valor devuelto, llamado *promise*, se resuelve en función del resultado de la misma, mientras tanto la ejecución del programa sigue su curso.

Directives: Las *directives* son el plato fuerte de AngularJS. Mediante el uso de las mismas podemos extender la sintaxis de HTML y darle el comportamiento que deseemos. Podemos crear *directives* a nivel de elemento, de atributo, de clase y de comentario. Un ejemplo sería el siguiente, mediante nuestra *directive focusable* (una *directive* a nivel de atributo) podemos modificar el comportamiento de los elementos input.

Scopes: Los *scopes* son los distintos contextos de ejecución sobre los que trabajan las expresiones de AngularJS, por ejemplo, cuando referenciamos un atributo del modelo mediante la *directive ng-model*, no estamos sino apuntando a un atributo que contiene el *scope* sobre el que se está trabajando. En los *scopes* se guarda la información de los modelos que se representan en la vista y también atributos que se utilizan para manejar la lógica de la misma.

Los *scopes* se manejan principalmente desde los *controllers* y desde las *directives*.

Data Binding: El "*data binding"* es uno de los puntos fuertes de AngularJS. Mediante el "data *binding"*, la vista se actualizará automáticamente cada vez que el modelo cambie y viceversa. Esto es posible debido a que se ha eliminado la manipulación del DOM, con todos los problemas que ello conlleva**.**

¿QUÉ ES CLOUDANT? (MORALES, 2014)

Es una base de datos de como servicio (DBaaS) la cual permite centrarse en el desarrollo rápido de aplicaciones en Internet y aplicaciones móviles en lugar de preocuparse por la expansión y gestión de la base de datos por su cuenta. Tiene alta disponibilidad, es duradera y contiene amplias funciones. El almacén de datos se construye para la escalabilidad y está optimizada para lecturas y escritura de datos simultáneas. Maneja también una amplia variedad de tipos de datos estructurados y no estructurados entre ellos JSON, textos completos y geoespacial.
La base de datos Cloudant es la primera plataforma de gestión de datos para aprovechar la disponibilidad, escalabilidad, y el alcance de la Nube para crear una red de distribución global de datos (DDN) que permita a las aplicaciones estar disponibles para los usuarios donde quiera que se encuentre.

METODOLOGÍA DE DESARROLLO ÁGIL SCRUM

Scrum es un proceso en el que se aplican de manera regular un conjunto de buenas prácticas para trabajar colaborativamente, en equipo, y obtener el mejor resultado posible de un proyecto. Estas prácticas se apoyan unas a otras y su selección tiene origen en un estudio de la manera de trabajar de equipos altamente productivos.
La base fundamental de esta metodología consiste en la división del trabajo completo (Product Backlog) en distintos apartados o bloques que pueden ser abordados en periodos cortos de tiempo (1-4 semanas) que se denominan Sprints. Básicamente los principales términos de Scrum son **(Schwaber & Sutherland, 2013)**:

- Reuniones (Planificación del sprint, Seguimiento del sprint, Revisión del sprint)
- Elementos (Product Backlog, Sprint Backlog, Incremento)
- Roles (Product Owner, Equipo, Scrum master)

Scrum se basa en la teoría de control de procesos empírica o empirismo. El empirismo asegura que el conocimiento procede de la experiencia y de tomar decisiones basándose en lo que se conoce. Scrum emplea un enfoque iterativo e incremental para optimizar la predictibilidad y el control del riesgo. Tres pilares soportan toda la implementación del control de procesos empírico: transparencia, inspección y adaptación **(Juan & Claudia, 2011)**.

- Transparencia: Los aspectos significativos del proceso deben ser visibles para aquellos que son responsables del resultado. La transparencia requiere que dichos aspectos sean definidos por un estándar común, de tal modo que los observadores compartan un entendimiento común de lo que se está viendo.
- Inspección: Los usuarios de Scrum deben inspeccionar frecuentemente los artefactos de Scrum y el progreso hacia un objetivo, para detectar variaciones. Su inspección no debe ser tan frecuente como para que interfiera en el trabajo. Las inspecciones son más beneficiosas cuando se realizan de forma diligente por inspectores expertos, en el mismo lugar de trabajo.
- Adaptación: Si un inspector determina que uno o más aspectos de un proceso se desvían de límites aceptables, y que el producto resultante no será aceptable, el proceso o el material que está siendo procesado deben ser ajustados. Dicho ajuste debe realizarse cuanto antes para minimizar desviaciones mayores.

Scrum prescribe cuatro eventos formales, contenidos dentro del Sprint:

- Reunión de Planificación del Sprint (Sprint Planning Meeting)
- Scrum Diario (Daily Scrum)
- Revisión del Sprint (Sprint Review)
  Retrospectiva del Sprint (Sprint Retrospective)

5. ALCANCES Y LIMITACIONES

En esta sección se da claridad a los alcances y limitaciones identificados para el desarrollo del presente proyecto

5.1. ALCANCES

- Implementación de aplicación web para gestión de actividad de la oficina (venta y seguimiento productos y servicios).
- Implementación de aplicación móvil híbrida orientada a la interacción con los clientes.
- Definición del modelo de datos a utilizar para la base de datos no relacional.
- La solución de software se contempla como un prototipo funcional que resuelve los problemas de negocio, será implementado en una nube pública para su presentación y pruebas.

5.2. LIMITACIONES

- No se contempla la implementación de esquema de seguridad en el prototipo.
- No se contempla la distribución de la aplicación móvil mediante las tiendas de aplicaciones para cada sistema operativo móvil.
- No se desplegará aplicación móvil en IOS (no se posee equipos ni licencias necesarias).
- No se contempla la integración con las compañías proveedoras de pólizas. El aplicativo web para la oficina únicamente contempla el control interno de la actividad.
- No se contempla integración de la aplicación móvil con medios de pago, únicamente será un canal para informar al usuario final sobre sus productos, productos ofrecidos y formas de adquisición de nuevos productos.
- No se contempla el hecho de que el cliente implemente de forma inmediata la solución propuesta en el desarrollo del proyecto.
- No se contempla la compra de dominios ni espacios en la nube superiores a los ofrecidos gratuitamente por diferentes empresas en la web.

6. METODOLOGÍA

Para la definición de la metodología que se utilizará en este proyecto, se realizó una investigación de metodologías ágiles de proyectos aplicadas para la creación de software y se determinó el uso de SCRUM.

Para comenzar, se definió el backlog del producto dentro del cual se reúne las historias acordadas con el cliente y también incluye actividades propias del desarrollo del producto, divididas en 6 sprints programados en el cronograma.

Tabla 2. Backlog del producto

Descripción	Responsable	Estimado en horas	Estado	ID	0	1	2	3	4	5	6
Diseño de la arquitectura	HR, DA			1	X						
Definición historias de usuario	HR, DA			2	X						
Definición de modelo de datos de la base de datos	HR, DA			3							
Diseño de estructura de documentos a almacenar para cada entidad en el sistema (cliente, producto)	HR, DA			3.1							
Validación de diseño de documentos	HR, DA			3.2							
Diseño de pantallas aplicación web	HR, DA			4							
Diseño de pantallas aplicación móvil	HR, DA			5							
Pruebas de registros en la base de datos	HR, DA			6		X	X	X	X	X	X
Elección servicios y entornos de ejecución a utilizar	HR, DA			7							

Crear layout y estilos web	HR, DA			8							
Crear página principal	HR, DA			8.1							
Creación nodo de comunicación con base de datos	HR, DA			9							
Crear aplicación móvil	HR, DA			10							
Creación nodo de comunicación con aplicación web y móvil.	HR, DA			11							
Creación comunicación entre nodos front end y base de datos	HR, DA			12							
Pruebas de integridad	HR, DA			13		X	X	X	X	X	X
Documentación	HR, DA			14		X	X	X	X	X	X
Sprints planning y Sprints review	HR, DA			15		X	X	X	X	X	X

Se definieron los siguientes roles según SCRUM y necesidades del proyecto:

- Director del proyecto: Jhon Freddy Parra Peña
- Scrum Master: Danilo Ariza
- Product Owner: Representante de la empresa “Nubia Pinto asesores”. Eliana Quiroga Salamanca.
- Equipo de desarrolladores: Danilo Ariza, Héctor Rodríguez
- Tester: Héctor Rodríguez

7. RECURSOS

7.1. RECURSO HUMANO

Danilo Alejandro Ariza Quiroga, Estudiante de Ingeniería de Sistemas, Universidad Distrital, Décimo Semestre.

Héctor Manuel Rodríguez Martin, Estudiante de Ingeniería de Sistemas, Universidad Distrital, Décimo Semestre.

John Freddy Parra Peña. Docente de planta, Universidad Distrital. Director del proyecto

7.2. RECURSO TECNOLÓGICO

7.2.1. Herramientas de Software.

- Cuenta de prueba IBM Bluemix (PaaS).
- Node JS. Entorno de ejecución basado en javascript.
- AngularJS. Framework MVC para desarrollo Front-end.
- Ionic. Framework de desarrollo de aplicaciones móviles híbridas basadas en Apache Cordova.
- Cloudant. Implementación de CouchDB, base de datos NoSQL orientada a documentos.
- GIT. Sistema de control de versiones.
- Microsoft Office (Word, Excel, PowerPoint) 2013. Herramienta de ofimática
- Enterprise Architect – Herramienta diagramas
- Cacoo – Herramienta de diagramas online

7.2.2. Herramientas de Hardware.

- Computador Portátil Asus VivioBook
- HTC M8x. Dispositivo Android para pruebas aplicación móvil
- Samsung Galaxy Nexus. Dispositivo Android para pruebas aplicación móvil

8. PRESUPUESTO

Tabla 3. Presupuesto del proyecto

RUBROS	COSTO MENSUAL ($)	COSTO TOTAL ($) (3 MESES)
CAPITAL HUMANO		
Scrum Master	4,500,000	13,500,000
Desarrollador 1	3,000,000	9,000,000
Desarrollador 2	3,000,000	9,000,000
Tester	3,000,000	9,000,000
Director del proyecto	4,500,000	13,500,000
RECURSOS DE HARDWARE		
Sistema operativo Windows 8.1		1,000,000
Microsoft Office 2013 Enterprise (Word-Excel-Power Point)		600,000
Computador Portátil Asus VivioBook		800,000
Computador escritorio, Core i5, 500gb de disco duro, 4gb de Ram		700,000
Planta Física	300,000	900,000
Servicios públicos (Agua, luz, internet)	150,000	4,500,00
Bibliografía		100,000
Transporte	160,000	480,000
Gastos varios	200,000	600,000
	TOTAL	59,630,000

9. CRONOGRAMA

A continuación, se muestra el cronograma de Sprints y ceremonias programadas durante el desarrollo del proyecto.

Tabla 4. Cronograma del Proyecto (Tiempo por Sprint)

	Fecha Inicial	**Fecha Final**	**Tiempo total**
Sprint 0	Febrero 1 de 2016	Febrero 14 de 2016	40 horas por persona
Sprint 1	Febrero 15 de 2016	Febrero 28 de 2016	40 horas por persona
Sprint 2	Febrero 29 de 2016	Marzo 13 de 2016	40 horas por persona
Sprint 3	Marzo 14 de 2016	Marzo 27 de 2016	40 horas por persona
Sprint 4	Marzo 28 de 2016	Abril 10 de 2016	40 horas por persona
Sprint 5	Abril 11 de 2016	Abril 24 de 2016	40 horas por persona
Sprint 6	Abril 25 de 2016	Mayo 8 de 2016	40 horas por persona

Tabla 5. Cronograma de Sprints planning y Sprints review

Ceremonia	**Fecha**	**Tiempo Estimado**
Sprint Planning 1	Febrero 14 de 2016	1 hora
Sprint Review 1	Febrero 28 de 2016	1 hora
Sprint Planning 2	Febrero 29 de 2016	1 hora
Sprint Review 2	Marzo 13 de 2016	1 hora
Sprint Planning 3	Marzo 14 de 2016	1 hora
Sprint Review 3	Marzo 27 de 2016	1 hora
Sprint Planning 4	Marzo 28 de 2016	1 hora
Sprint Review 4	Abril 10 de 2016	1 hora
Sprint Planning 5	Abril 11 de 2016	1 hora
Sprint Review 5	Mayo 8 de 2016	1 hora
	Tiempo total	10 horas

10. SPRINT PLANNING

Tabla 6: Detalle Sprint Planning

Sprint	Historia de usuario	Puntos de historia	Tarea	Fecha inicio	Fecha final	Sprint Planning
1	Diseño de la arquitectura	13	Definición de flujo de información entre frontend y backend	15 de febrero	28 de febrero	14 de febrero
			Elección de servicios y entornos de ejecución en PaaS			
	Definición del modelo de datos	8	Definición estructura genérica para documentos JSON			
2	Creación módulo registro de cliente	20	Diseño y validación estructura de documento (Cliente) a almacenar	29 de febrero	13 de marzo	29 de febrero
			Creación recurso en servidor (Entidad cliente)			
			Creación componente registro (Cliente) en aplicación web			
	Creación módulo registro vehículo cliente	20	Diseño y validación estructura de documento (Vehículo) a almacenar			
			Creación recurso en servidor (Entidad vehículo)			
			Creación componente registro (Vehículo) en aplicación web			
3	Creación módulo registro venta SOAT	20	Diseño y validación estructura de documento (SOAT) a almacenar	14 de marzo	27 de marzo	14 de marzo
			Creación recurso en servidor (Entidad SOAT)			
			Creación componente registro (SOAT) en			

			aplicación web			
	Creación módulo registro póliza de cumplimiento	20	Diseño y validación estructura de documento (Póliza cumplimiento) a almacenar Creación recurso en servidor (Entidad Póliza cumplimiento) Creación componente registro (Póliza cumplimiento) en aplicación web			
4	Creación módulo registro venta seguro todo riesgo vehículos Creación módulo administrativo (Aplicación web)	20 20	Diseño y validación estructura de documento (Todo riesgo) a almacenar Creación recurso en servidor (Entidad Todo riesgo) Creación componente registro (Todo riesgo) en aplicación web Módulo registro valores SOAT Creación módulo empresas proveedoras	28 de marzo	10 de abril	28 de marzo
5	Creación módulo administración pólizas Creación módulo solicitud y adquisición pólizas (perfil cliente)	20 20	Módulo actualización estado póliza Módulo actualización información póliza Módulo registro pago realizado Adaptación métodos en backend para perfil cliente Diseño pantalla información pólizas ofrecidas Adaptación formularios solicitud pólizas	11 de abril	24 de abril	11 de abril
6	Creación módulo	13	Creación componentes	25 abril	8 mayo	25 de abril

	consulta productos cliente (mis productos)		frontend para módulo consulta productos cliente			
			Creación métodos backend para módulo consulta productos cliente			
			Integración componentes frontend y backend para módulo consulta productos cliente			
	Creación módulo información cliente	13	Implementación métodos en backend para edición de información de cliente y entidades relacionadas			
			Creación componentes frontend para módulo información cliente			
			Integración componentes frontend y backend para módulo información cliente			
	Creación módulo simulado de autenticación	13	Creación servicio simulado de autenticación			
			Creación componente de autenticación en aplicación web y móvil			
	Creación módulo información vehículo	13	Implementación métodos en backend para edición de información de vehículo y entidades relacionadas			
			Creación componentes frontend para módulo información vehículo			
7			Integración componentes frontend y backend para módulo información vehículo	9 de mayo	22 de mayo	9 de marzo
Total, puntos de historia: 233						

11. DIAGRAMAS

11.1. Diagrama de arquitectura

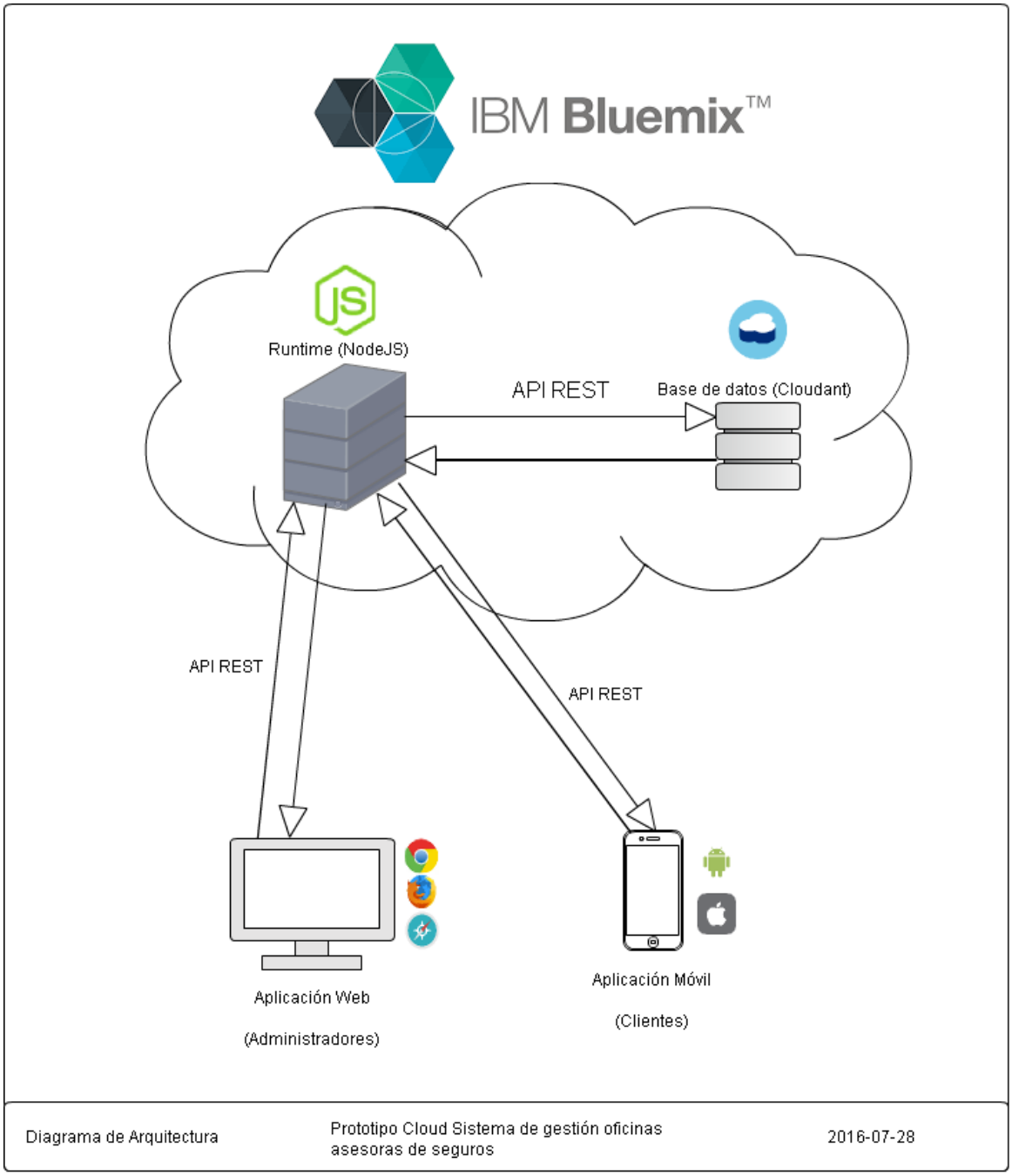

Figura 2: Diagrama arquitectura

11.2. Diagrama de componentes

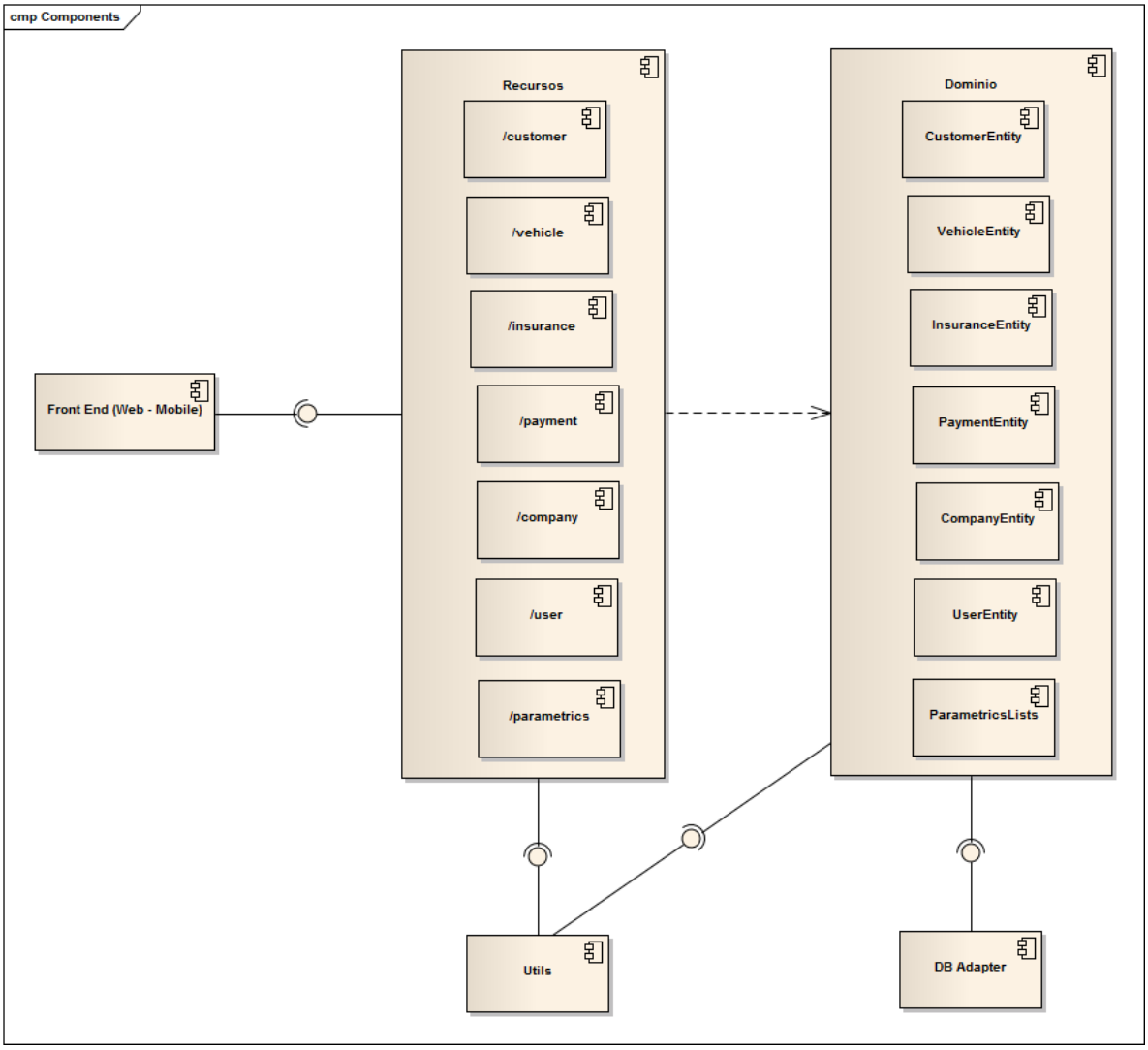

Figura 3: Diagrama componentes

12. MODELO DE NEGOCIO

Para el desarrollo del software se tomó como base la actividad productiva de la oficina asesora en la venta de seguros de diferentes tipos para vehículos (Nubia Pinto Asesores de seguros e inversiones), así mismo, para mostrar de una forma más clara las ventajas de usar tecnologías cloud en el desarrollo y despliegue se presenta una estimación aproximada de costos y comercialización de la aplicación utilizando el método de distribución Software as Service (SaaS).

Se realiza una estimación del número de peticiones que se realizan a la base de datos en cada uno de los procesos posee la aplicación hasta el momento, estas van a ser la base de la estimación total del costo mensual para la empresa ya que la plataforma de desarrollo IBM Bluemix (PaaS) también estima los costos de su servicio basados en el número de peticiones.

Según la estimación mensual hecha durante el desarrollo del proyecto con el número de peticiones realizadas en las pruebas de unidad (2500 peticiones) se estima que para una pequeña empresa como la que se tomó de referencia, los costos mensuales en la plataforma IBM Bluemix (PaaS) son:

Tabla 7: Costo mensual servicios IBM Bluemix

ITEM	**PRECIO US$**	**USO**	**MEMORIA**	**TOTAL (COP $)**
SDK for Node.js	0,0805 /GB hora	1 instancia	1GB	86,000
Cloudant NoSQL DB	1,15 /GB	1000 llamadas al API	20 GB almacenamiento	54,000
Soporte mensual				
			TOTAL	140,000

Se procede a realizar el estimado de peticiones que realiza cada uno de los procesos de la aplicación:

Tabla 8: Número de peticiones por proceso

PROCESO	**CANTIDAD DE PETICIONES A BD**	**¿Solo administrador?**
Crear compañía	2	Si
Actualizar compañía	2	Si
Obtener compañía	1	Si
Generar pagos	2	Si (Forma automática)
Registrar pago realizado	2	Si
Obtener pagos por periodo de tiempo	1	Si
Actualizar vehículo	4	Si
Actualizar estado póliza	2	Si
Eliminar póliza (No emitida)	2	Si
Registrar valores SOAT	2	Si
Crear cliente	2	No
Actualizar cliente	4	No
Obtener cliente	1	No
Obtener valor SOAT	1	No
Productos por cliente	1	No
Pagos por póliza	1	No
Iniciar sesión	1	No
Registrar usuario	3	No
Registrar vehículo	2	No
Obtener vehículo	1	No
Registrar póliza	4	No

Teniendo en cuenta el número de peticiones presentado en la tabla anterior se calcula un estimado de peticiones según el volumen de ventas mensuales que la oficina asesora está obteniendo.

Tabla 9: Costos mensuales estimados peticiones

Tipo de proceso	**Cantidad mensual**	**Peticiones/Registro (Peor caso)**	**Peticiones/Registro (Mejor caso)**	**Total Peticiones mensuales**	**Precio(COP$) mensual**
SOAT	144	22	10	3168 / 1440	285,000/71,112
Cumplimiento	5	19	10	95 / 50	6,650/7,800
Todo riesgo	5	37	10	185 / 50	26,600/31,200
Peticiones administrativas	-	98	64	98 / 64	6,650/7,800
TOTAL	154	176	94	3546/1604	324,900/117,912

Teniendo en cuenta el presupuesto presentado en el punto **8. Presupuesto** se tienen en cuenta los siguientes costos globales de desarrollo del proyecto:

Tabla 10: Costos capital humano desarrollo

CAPITAL HUMANO (3 meses)	
Desarrollador 1	9,000,000
Desarrollador 2	9,000,000
Costos de desarrollo	2,000,000
TOTAL	20,000,000

Como se está estimando un costo total mensual , es necesario incluir los costos del desarrollo dentro un plan de pagos que sea accesible para la empresa y al mismo tiempo

permita obtener ganancias para el equipo de trabajo, por este motivo, se propone un esquema de venta de la aplicación a varias empresas similares, es decir , a otras oficinas asesoras de seguros (mínimo 3 empresas) para que dichos costos de desarrollo sean divididos entre estas con el siguiente esquema básico de contratación:

- 5 años de contratación de los servicios de la plataforma IBM Bluemix y servicios desarrollados en la aplicación como tal.
- Configuración inicial sin costo.
- Soporte en línea mensual en caso de ser necesario.

Con el esquema de contratación propuesto, el costo mensual para cada empresa será aproximadamente:

Tabla 11:Total estimado mensual

ITEM	**PRECIO (COP$)**
Costo promedio mensual Servicios IBM Bluemix	221,406
Costo promedio mensual aplicación	112,000
TOTAL	333,406

13. CONCLUSIONES

- Mediante la aplicación de la metodología SCRUM se facilitó la definición de las necesidades del cliente; dichas necesidades fueron identificadas en reuniones con el cliente y redactadas como historias que a su vez permitieron formar el backlog del proyecto y dar inicio a la ejecución de la aplicación dentro de un tiempo razonable.

- El trabajo mediante la metodología scrum facilitó el trabajo durante el desarrollo del proyecto permitió corregir el rumbo del trabajo de manera temprana, lo cual ayudó al equipo de trabajo a estar siempre enfocados en las necesidades del cliente.

- Utilizar tecnologías de desarrollo y despliegue cloud permitió construir una aplicación web que no depende estrictamente de una infraestructura de hardware, a su vez el contar con un ambiente de desarrollo que unifica diversos frameworks y tecnologías facilitó construir casi paralelamente una aplicación móvil que reutiliza en gran parte lo desarrollado con anterioridad y permite ofrecer al cliente una experiencia más completa y productiva.
- Ofrecer al cliente un modelo de negocio de software como servicio permite que los gastos operacionales de sus recursos tecnológicos (Software y/o **hardware)** están estrechamente ligados a la evolución de su actividad económica, previniendo gastos innecesarios más asociados a la subutilización de dichos recursos, finalmente el cliente paga lo que consume.

BIBLIOGRAFÍA

Abernethy, M. (2011). ¿Simplemente qué es Node.js? Retrieved February 12, 2016, from https://www.ibm.com/developerworks/ssa/opensource/library/os-nodejs/

Amazon Web Services. (2010). Software- as- a- Service (SaaS) on AWS Business and Architecture Overview. *Amazon Web Services*, (September), 1–11.

Burd, G. (2011). NoSQL. *Login*, *36(5)*, 5–12.

Butler, B. (2013). PaaS primer: What is platform as a service and why does it matter? Retrieved February 7, 2016, from http://www.infoworld.com/article/2613027/paas/paas-primer--what-is-platform-as-a-service-and-why-does-it-matter-.html

Childs, M. (2011). John McCarthy: Computer scientist known as the father of AI. *Independent*. Retrieved from http://www.independent.co.uk/news/obituaries/john-mccarthy-computer-scientist-known-as-the-father-of-ai-6255307.html

Couchbase. (2014). Why NoSQL? *Couchbase*.

Creasy, R. J. (1981). The Origin of the VM / 370 Time-sharing System. *IBM Journal of Research and Development*, *25*(5), 483–790.

Deloitte Consulting LLP. (2015). 2015 Global Contact Center Survey Results. Top 10 Insights. *Deloitte*, (March), 1 – 45. http://doi.org/http://www2.deloitte.com/content/dam/Deloitte/us/Documents/process-and-operations/us-sdt-consulting-2013-global-contact-survey-051513.pdf

Gilbert, S., & Lynch, N. (2012). Perspectives on the CAP Theorem. *Computer*, *45*(2), 30–36. http://doi.org/10.1109/MC.2011.389

Green, B., & Seshadri, S. (2013). *AngularJS*. (S. St. Laurent & M. Blanchette, Eds.) (1st ed.). Sebastopol: O'Reilly Media.

Grolinger, K., Higashino, W. a, Tiwari, A., & Capretz, M. A. (2013). Data management in cloud environments: NoSQL and NewSQL data stores. *Journal of Cloud Computing: Advances, Systems and Applications*, *2*, 22. http://doi.org/10.1186/2192-113X-2-22

IBM Cloud. (n.d.). What is cloud computing? Retrieved January 30, 2016, from http://www.ibm.com/cloud-computing/what-is-cloud-computing.html

Juan, P., & Claudia, R. (2011). *Gestión de proyectos Scrum Manager* (2nd ed.). Scrum Manager. Retrieved from http://www.scrummanager.net/files/sm_proyecto.pdf

Kepes, B. (2013). Understanding the Cloud Computing Stack SaaS, Paas, IaaS. *Rackspace Support Network*, 1–20. Retrieved from https://www.rackspace.com/knowledge_center/sites/default/files/whitepaper_pdf/Understanding-the-Cloud-Computing-Stack.pdf

Kepes, B. (2015). New Stats From The State Of Cloud Report. *Forbes*. Retrieved from http://www.forbes.com/sites/benkepes/2015/03/04/new-stats-from-the-state-of-cloud-report/#20733f6826f9

Morales, J. A. (2014). Características y tipos de bases de datos. Retrieved February 12, 2016, from

https://www.ibm.com/developerworks/ssa/data/library/tipos_bases_de_datos/

Neto, M. D. (2014). A brief history of cloud computing. Retrieved January 30, 2016, from http://www.thoughtsoncloud.com/2014/03/a-brief-history-of-cloud-computing/

Nodejs. (n.d.). Nodejs Documentation. Retrieved February 12, 2016, from https://nodejs.org/api/readline.html

Nubia Pinto - Asesores de seguros e inversiones. (n.d.). Duitama Calle 15 # 12 - 59 Piso 2.

Prol, D. O. (2014). Introducción a AngularJS. Retrieved February 12, 2016, from https://blog.chattyhive.com/2014/04/introduccion-angularjs/

Rose, R. (2004). Survey of system virtualization techniques. *Collections*, *15*, 1–15. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.58.9811&rep=rep1&type=pdf\nhttp://ir.library.oregonstate.edu/jspui/handle/1957/9907

Schwaber, K., & Sutherland, J. (2013). La Guía de Scrum. Scrum.org. Retrieved from https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&ved=0ahUKEwjt97Hhg8vKAhWCth4KHRGyD3EQFggoMAE&url=http://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-ES.pdf&usg=AFQjCNHRvg7sWErm-1iZTlV1sDE84q3gQA&sig2=Ji97XwR6X97_Czj4evn

Stefan Kolb. (2013). paas-profiles. Bamberg, Germany: GitHub. Retrieved from https://github.com/stefan-kolb/paas-profiles

Sullivan, D. (2015). NoSQL for Mere Mortals (1st ed., pp. 59–75). Michigan: Addison Wesley.

Urueña, A., Ferrari, A., Blanco, D., & Valdecasa, E. (2010). Cloud computing. *Principles and Paradigms*.

WhatIsCloud.com. (n.d.). Cloud. Retrieved February 7, 2016, from http://whatiscloud.com/basic_concepts_and_terminology/cloud

Wu, D., Hugenholtz, P., Mavromatis, K., Pukall, R., Dalin, E., Ivanova, N. N., … Eisen, J. a. (2009). CLOUD COMPUTING – An Overview An Overview. *White Paper*, *462*(7276), 1–5. http://doi.org/10.1038/nature08656

ANEXOS

ANEXO 1: MODELO DE DATOS Y RELACIONES ENTRE ENTIDADES

Las entidades involucradas en el prototipo son:

- Pago (recibo)
- Cliente
- Seguros (SOAT, todo riesgo vehículos, cumplimiento de contratos)
- Compañía
- Vehículo
- Usuario

Las relaciones existentes entre entidades son:

- Un cliente puede tener uno o varios seguros.
- Una compañía puede emitir uno o varios seguros.
- Un seguro puede tener uno o varios pagos.
- Un pago está asociado sólo a una póliza.
- En pólizas de seguros para vehículos, una póliza solo está asociada a un vehículo.
- Un usuario está asociado a sólo un cliente.

Dado que la oficina funciona como intermediario en la adquisición de las pólizas, es necesario manejar estados en cuanto a su ciclo de vida (solicitada, emitida, cancelada, etc).

Las pólizas pueden tener uno o múltiples pagos dependiendo de su tipo.

El valor del SOAT depende del tipo y subtipo de vehículo, toda esta clasificación se modifica anual por el gobierno, en la aplicación se maneja dichos valores mediante documentos almacenados en Cloudant; dichos valores se agrupan por años y se exponen mediante una vista en Cloudant.

Se maneja estados de las compañías, para evitar emitir pólizas nuevas con compañías que ya no se estén usando en la oficina

DATOS PARA CADA UNA DE LAS ENTIDADES

CLIENTE

Tabla 12: Detalle entidad Cliente

PARÁMETRO	TIPO DE DATO	TAMAÑO	REQUERIDO	ETIQUETA	DOMINIO
Nombre 1	String	50	SI	Nombre1	
Nombre 2	String	50	NO	Nombre2	
Apellido 1	String	50	SI	Apellido1	
Apellido 2	String	50	NO	Apellido2	
Tipo documento	Number	3	SI	TIpoDoc	1= Cédula 2=Pasaporte
Número documento	Number	18	SI	NumDoc	Entero sin puntos
Dirección	String	50	SI	Direccion	
Télefono	Number	10	SI	Telefono	Entero sin puntos
Celular	String	15	NO	Celular	Entero sin puntos
Departamento	String	30	SI	Departamento	
Ciudad	String	30	SI	Ciudad	
Correo electrónico	String	50	NO	Email	

VEHÍCULO

Tabla 13: Detalle entidad Vehículo

PARÁMETRO	TIPO DE DATO	TAMAÑO	REQUERIDO	ETIQUETA	DOMINIO
Marca	String	20	SI	Marca	
Línea	String	20	SI	Linea	
Modelo	Number	4	SI	Modelo	Entero sin puntos
Placa	String	8	SI	Placa	
Color	String	20	SI	Color	
Motor	String	20	SI	Motor	
Chasis	String	20	SI	Chasis	
Cilindraje	Number	4	SI	Cilindraje	
Servicio	String	20	NO	Servicio	
Categoría	String	20	SI	TipoVehiculo	
Subcategoría	String	20	SI	SubtipoVehiculo	

COMPAÑÍA

Tabla 14: Detalle entidad Compañía

DATO	TIPO DE DATO	TAMAÑO	REQUERIDO	ETIQUETA	DOMINIO
NIT	String	20	SI	NIT	
Razón social	String	50	SI	RazonSocial	
Dirección	String	20	SI	DireccionCom	
Teléfono	Number	15	SI	TelCompania	Entero sin puntos
Estado	Boolean	-	SI	Estado	

USUARIO

Tabla 15: Detalle entidad Usuario

DATO	TIPO DE DATO	TAMAÑO	REQUERIDO	ETIQUETA	DOMINIO
Tipo documento	String	2	SI	TipoDoc	CC, NT, CE, PA
Numero Documento	Number	20	SI	NumDoc	
Usuario Administrador	Boolean	-	SI	esAdmin	
Usuario	String	20	SI	User	
Contraseña	String	20	SI	Pass	

PÓLIZA / SEGURO

Tabla 16: Detalle entidad Póliza

DATO	TIPO DE DATO	TAMAÑO	REQUERIDO	ETIQUETA	DOMINIO
Tipo	Number	3	SI	TipoPoliza	1=SOAT 2=Cumplimiento 3=Todo riesgo vehículo
Descripción	String	20	SI	DescPoliza	
Compañía	Number	20	SI	Compañia	
Estado	String	1	SI	EstadoProceso	
Valor asegurado	Number	20	SI	ValorAsegurado	
Valor a Pagar	Number	20	SI	ValorAPagar	
Diferido	Boolean		NO	Diferido	

Fecha de Inicio	Date		NO	FechaInicio	DD/MM/YYYY
Fecha de vencimiento	Date		NO	FechaFin	DD/MM/YYYY
Tipo Documento Cliente	String	2	SI	TipoDocCliente	CC, NT, CE, PA
Numero Documento Cliente	Number	20	SI	NumDocCliente	
Vehículo *	String	6	SI	Vehículo	
NIT Empleador **	Number	20	SI	NITEmpleador	
Nombre empleador **	String	30	SI	NombreEmpleador	
Tipo de Contrato **	String	2	SI	TipoContrato	1=Término fijo 2=Término indefinido 3=Obra o labor 4=Prestación de servicios 5=Ocasional

* Solo son requeridos si el tipo de seguro involucra vehículo (SOAT, Todo Riesgo)
** Sólo son requeridos si el tipo de seguro involucra un contrato (Póliza de cumplimiento de contrato)

PAGO

Tabla 17: Detalle entidad Pago

DATO	TIPO DE DATO	TAMAÑO	REQUERIDO	ETIQUETA	DOMINIO
Serial Seguro	String	30	SI	SerialSeguro	
Compañia	Number	20	SI	Compañia	
Número de Pago	Number	1	SI	NumeroPago	

Monto	Number	20	SI	Monto	
Estado	String	1	SI	Estado	1=Pago pendiente 2=Pago realizado
Fecha límite de pago	Date		NO	FechaLimite	DD/MM/YYYY
Fecha de pago *	Date		NO	FechaPago	DD/MM/YYYY
Tipo Documento Cliente	String	2	SI	TipoDocCliente	CC, NT, CE, PA
Numero Documento Cliente	Number	20	SI	NumDocCliente	

* Sólo es requerido si el estado del pago es 2 (Pago realizado)

ANEXO 2: SEGUIMIENTO SPRINTS (DAILY, REVIEW, RETROSPECTIVE)

Tabla 18: Sprint 1

DAILY (Seguimiento)					
Fecha		**Integrante**	**¿Que se ha trabajado?**	**¿Qué se va a trabajar?**	**Impedimentos encontrados**
		Héctor Rodríguez	Se ha revisado documentación de los diferentes servicios disponibles en bluemix para almacenamiento de data NoSQL	Se trabajará en pruebas de concepto del servicio Cloudant para almacenamiento de documentos en formato JSON	NA
	16	Danilo Ariza	Se ha revisado documentación de los diferentes runtime disponibles en bluemix para ejecución de aplicaciones	Se trabajará en pruebas de concepto del runtime NodeJS para la construcción de servicios y exponer la aplicación web utilizando el lenguaje JavaScript	NA
		Héctor Rodríguez	Se ha revisado el API REST que posee cloudant para realizar el CRUD sobre la información almacenada	Se revisará el tema de índices, ya que en la documentación se menciona que son utilizados para la obtención de documentos, adicionalmente se investigará sobre las vistas en cloudant	Problemas en la obtención de documentos, mediante campos diferentes al _id
	18	Danilo Ariza	Se ha revisado el framework Express, con el cual se puede definir servicios en NodeJS	Se mirará como se puede separar por "capas" distintos archivos .js para simular un comportamiento orientado a objetos	Se encuentra que ciertas características de javascript que involucran arreglos no están presentes en el motor de javascript de Node JS
		Héctor Rodríguez	Se trabajó en índices y en el funcionamiento de vistas dentro de cloudant, se revisa la documentación de couchDB ya que en cuanto a vistas no es muy clara la de cloudant	Teniendo en cuenta el backlog del producto, se buscará definir los casos en los cuales sería de utilidad el uso de vistas	NA
	20	Danilo Ariza	Teniendo en cuenta las características que se encuentra que no están disponibles en el motor de javascript de node JS, se indaga y se encuentra una librería util para estos faltantes, la cual es underscore JS; se revisó su documentación y se deja en el repositorio para su posible uso	Se definirá la estructura de archivos y carpetas dentro del repositorio tanto para backend como para frontend (aplicación web)	Cada prueba en backend implica un redespliegue del servidor, el cual a veces tarda un poco, se buscará alternativas dentro de las herramientas de desarrollo de bluemix
Febrero	22	Héctor Rodríguez	Se inicia con la definición del adaptador para la comunicación con la base de datos, por ahora	Se trabajará, apoyado en la documentación	NA

			solo conexión y obtención de documentos	de la librería de cloudant para node, en la definición de los métodos para realizar todas las operaciones sobre documentos y sobre vistas	
		Danilo Ariza	De acuerdo a los problemas comentados en el último daily, se encuentra que para el runtime de node hay una opción llamada "live edit" la cual permite refrescar los archivos de backend sin necesidad de un redespliegue del servidor; los tiempos de pruebas de cambios se reducen considerablemente. adicionalmente se finaliza con la definición de las capas para el backend (adaptador, entidades, recursos, utilitarios, recursos frontend	Se integrará la versión inicial del adaptador de base de datos a una entidad de prueba para probar la comunicación de una entidad con un documento en la base de datos	NA
		Héctor Rodríguez	Se finaliza con la creación de los métodos en el adaptador de base de datos para la interacción con documentos y vistas	En conjunto trabajaremos en la definición de la estructura genérica para todos los documentos a almacenar en cloudant	NA
	24	Danilo Ariza	Se finaliza con la integración del adaptador, teniendo en cuenta los últimos métodos agregados, adicionalmente se hicieron pruebas desde un cliente REST para probar la capa de recursos.	En conjunto trabajaremos en la definición de la estructura genérica para todos los documentos a almacenar en cloudant	NA
		Héctor Rodríguez	En conjunto se finalizó con la definición genérica (plantilla) de los documentos JSON, teniendo en cuenta las posibles entidades que se almacenarán en base de datos	Lectura de documentación de angular para trabajar la aplicación web	NA
	26	Danilo Ariza	En conjunto se finalizó con la definición genérica (plantilla) de los documentos JSON, teniendo en cuenta las posibles entidades que se almacenarán en base de datos. Adicionalmente se documenta el modelo de datos en un archivo aparte	Teniendo en cuenta las capacidades del framework escogido para frontend (AngularJS) se definirá la estructura de capas que se maneja dentro de las aplicaciones	NA
	28	Héctor Rodríguez	Se realizó una lectura del framework angular y se apoya la definición de su uso dentro del	Se van a realizar pruebas generales del primer esquema del backend para poder	NA

			proyecto	cerrar la historia de usuario de arquitectura	
		Danilo Ariza	Se finaliza con la definición de las capas (vistas, controladores, servicios y directivas como componentes transversales)	Se hará de nuevo una revisión de lo requerido por el cliente para la aplicación con el fin de poder determinar qué componentes podrían ser útiles desde frontend	NA

Fecha	**Review**	**Retrospective**
febrero 28	Se entregó una estructura bien definida en cuanto a aplicación y a datos, la cual se utilizará como framework durante el desarrollo del proyecto permitiendo a todos los involucrados tener un entendimiento de cada uno de los módulos contra historias de usuario	La comunicación al interior del equipo fue de gran ayuda para resolver los problemas que se iban presentando, algunas cosas se pudieron hacer más rápido pero el hecho de no estar familiarizado del todo con la plataforma influyó en parte; cada integrante se compromete a indagar en la mayoría de las funcionalidades posibles tanto en la herramienta SCRUM como en las de desarrollo

Tabla 19: Sprint 2

DAILY (Seguimiento)					
Fecha		**Integrante**	**¿Qué se ha trabajado?**	**¿Qué se va a trabajar?**	**Impedimentos encontrados**
		Héctor Rodríguez	Se ha trabajado, basado en la definición del modelo de datos, en la construcción del JSON para la información de clientes	Basado en la definición del documento JSON, se iniciará con la definición de la entidad Cliente en la cual se van a declarar los métodos de interacción con la misma	NA
	1	Danilo Ariza	Se ha trabajado, basado en la definición del modelo de datos, en la construcción del JSON para la información de vehículos, teniendo en cuenta que estos tienen relación con clientes	Basado en la definición del documento JSON, se iniciará con la definición de la entidad Vehículo en la cual se van a declarar los métodos de interacción con la misma	NA
		Héctor Rodríguez	Se ha trabajado en la construcción del método creación de registro de cliente validando que como parámetro se le envíe los valores definidos en el modelo de datos	Las declaraciones de validaciones de formato y de valores que por pruebas iniciales se estaban realizando en la entidad, se moverán al módulo de utilitarios y se llamarán desde allí	Problemas para la definición de la validación de un dato numérico en javascript
	3	Danilo Ariza	Se ha trabajado en la construcción del método creación de registro de vehículo de cliente validando que como parámetro se le envíe los valores definidos en el modelo de datos	Las declaraciones de validaciones de formato y de valores que por pruebas iniciales se estaban realizando en la entidad, se moverán al módulo de utilitarios y se llamarán desde allí, adicionalmente se encuentra la necesidad de manejar listas tipo clave valor, para manejar valores paramétricos como tipo de documento, ciudades, departamentos, tipos y subtipos de vehículos, etc	NA
Marzo	4	Héctor Rodríguez	Se resuelve la validación de datos numéricos adicionalmente se adiciona la validación de ciertos valores contra listas paramétricas que antes se realizaban directamente en código	Se incluirá mensaje de respuesta a operaciones y se tendrá en cuenta los códigos de respuesta de HTTP para informar de una operación al	NA

				cliente final	
		Danilo Ariza	Se define las listas paramétricas a utilizar dentro de la aplicación, adicionalmente se integra la validación de ciertos valores contra listas paramétricas	Se incluirá mensajes de respuesta y códigos http para informar del resultado de la operación realizada adicionalmente se agrega validación de que el cliente exista registrado para poder registrar un vehículo	Cuando el archivo no existe se lanza una excepción y el servidor automáticamente se muere, se está mirando cómo resolver dicho asunto con manejo de excepciones
		Héctor Rodríguez	Se incluye mensajes de respuesta en la operación	Se procederá a integrar la entidad de cliente contra la capa de recursos para poder exponerla como servicio	NA
	7	Danilo Ariza	Se resuelve lo del manejo de excepciones en la lectura de archivos, adicionalmente se incluye los mensajes de respuesta para los resultados de las operaciones, se define que estos se llamarán desde el módulo de utilitarios para que si más adelante un mismo mensaje se va a utilizar, que este sea un recurso común. Se hizo validación de cliente contra tipo y número de documento	Se integrará entidad contra capa de recursos para exponerla como servicio	NA
		Héctor Rodríguez	Se integró entidad con capa de recursos, además se hacen pruebas con un JSON de ejemplo mediante un cliente REST	Se van a realizar pruebas teniendo en cuenta varias situaciones y a realizar ajustes generales para cerrar la tarea y arrancar con frontend	NA
	9	Danilo Ariza	Se integró entidad con capa de recursos, además se hacen pruebas desde un cliente REST para los distintos escenarios posibles en el registro de un vehículo	Se arrancará con la definición del servicio en angular	Cuando uno envía un JSON mal armado (mal estructurado), el servidor de una vez envía la traza del error como respuesta, no alcanza a ingresar al método de la entidad; se necesita manejar dicha situación dado que, para el cliente final de la

					aplicación, un mensaje así no sería entendible
		Héctor Rodríguez	Se realizó pruebas del servicio con cliente REST, adicionalmente se define el servicio en angular para el consumo del servicio de creación del cliente	Se definirá el controlador para el módulo en la aplicación web, el cual se comunicará con backend mediante el servicio de angular; adicionalmente se creará un formulario inicial para pruebas desde frontend	Realizando pruebas en múltiples vistas no sabría cómo pasar datos entre ellas mediante sus controladores, funcionalidad muy necesaria más adelante
	11	Danilo Ariza	Se define el servicio en angular para registro de vehículo de cliente, adicionalmente se adiciona un middleware en nodejs para el manejo de errores referentes a la estructura del JSON que se envía dentro del body de la petición REST; cuando este se encuentra mal construido, se intercepta el error y al cliente se le envía un mensaje más entendible	Se definirá el controlador para el módulo en la aplicación web, el cual se comunicará con backend mediante el servicio de angular; adicionalmente se creará un formulario inicial para pruebas desde frontend	NA
		Héctor Rodríguez	En conjunto se revisa lo de paso de datos usando sessionStorage, se define un servicio de angular para tal fin; además se finalizó con la construcción del formulario y las validaciones desde vista para cada campo	Se realizarán pruebas desde formulario contra backend para cerrar tareas, adicionalmente se iniciará a agregar estilos visuales con Bootstrap	NA
	13	Danilo Ariza	En conjunto se revisa el tema de paso de datos entre vistas mediante sus controladores, se define que mediante el uso de los recursos del navegador, más específicamente de "sessionStorage" se realizará este puente; además se finalizó con la construcción del formulario	Se realizarán pruebas desde formulario contra backend para cerrar tareas, adicionalmente se iniciará a agregar estilos visuales con Bootstrap	NA

			y las validaciones desde vista para cada campo		
		Fecha	**Review**	**Retrospective**	
		marzo 13	Se entrega una versión inicial y funcional de registro de clientes y sus vehículos, la cual el cliente puede usar ya que involucra frontend, servicios y base de datos ya definidos	La forma de trabajo definida en la iteración 1 ayudó mucho para la definición de los módulos en todas las capas. El hecho de no haber dimensionado de manera correcta las historias de usuario para este sprint afectó demasiado ya que para este sprint se había definido hacer las funcionalidades de SOAT y todo riesgo vehículos, ya que ambas involucran vehículos; de estas no se hizo en ninguna funcionalidad en sí de la póliza. Se realiza una revisión de las historias para realizar una redefinición de estas, se separará lo del sprint 2 en 4 historias (Clientes, vehículos, SOAT y todo riesgo vehículos) y se recalculara puntos de historia para las historias resultantes. Se debe tener mucho cuidado en no sobreestimar ni subestimar el trabajo ya que afecta el ritmo del equipo

Tabla 20: Sprint 3

Fecha		Integrante	¿Que se ha trabajado?	¿Qué se va a trabajar?	Impedimentos encontrados
Marzo	15	Héctor Rodríguez	Se ha trabajado en el diseño del documento JSON para la venta de póliza del cumplimiento, basado en el modelo de datos ya creado	Se trabajará en la construcción de la entidad de la póliza de cumplimiento, definición de los métodos de registro, actualización y eliminación.	N/A
		Danilo Ariza	Se ha trabajado en el diseño del documento JSON para la venta de SOAT, basado en el modelo de datos ya creado	Se trabajará en la construcción de la entidad de SOAT, definición de los métodos de registro, actualización y eliminación.	N/A
	17	Héctor Rodríguez	Se ha trabajado en el método de registro de la póliza de cumplimiento siguiendo la estructura definida en el documento JSON creado.	Se trabajará en la implementación de las validaciones de formato y de información para controlar posibles errores en el registro de la póliza de cumplimiento.	N/A
		Danilo Ariza	Se ha trabajado en el método de registro del SOAT siguiendo la estructura definida en el documento JSON creado.	Se trabajará en la utilización de una lista paramétrica que contenga los valores de SOAT según tipo de vehículo.	N/A
	19	Héctor Rodríguez	Se implementa las validaciones de los datos como Ciudad, Tipo de documento, etc., con las listas paramétricas previamente creadas y se utilizan los códigos de error también creados previamente en el módulo de utilitarios.	Se trabajará en la implementación de consultas de la información del cliente y vehículo para que esta quede relacionada en el registro de la póliza de cumplimiento.	N/A
		Danilo Ariza	Se crea la lista paramétrica con los valores del SOAT y se implementan las validaciones de formatos y de información en el registro de SOAT	Se trabajará en la implementación de consultas de la información del cliente y vehículo para que esta quede relacionada en el registro de SOAT.	Se encuentra que manejar los valores del SOAT en una lista paramétrica es un problema de seguridad ya que sería sencillo para un tercero ver y modificar estos valores.
	21	Héctor Rodríguez	Se realizaron las consultas de los documentos en la base de datos de cliente y vehículos en el método de registro de la póliza de cumplimiento.	Se trabajará en el manejo de las posibles excepciones de las consultas y en los códigos de error en el utilitario.	Se encuentra que es necesario contar con información de compañías aseguradoras registradas previamente para que se pueda proseguir con el registro de la póliza de cumplimiento.
		Danilo Ariza	Se soluciona el problema de los valores del SOAT creando un documento inicial en la base de datos con un id único. Se crea una vista en la base de datos que permita la consulta del valor del SOAT por categoría y subcategoría. Se implementa la consulta de la información del	Se trabajará en la creación de la consulta de los valores del SOAT basada en la vista creada, en los métodos de actualización y eliminación de SOAT, también la creación de la capa de recursos de registro de seguros.	N/A

			cliente y vehículo en el método de registro de SOAT		
	23	Héctor Rodríguez	Se soluciona el problema de la información de la compañía creando validaciones basadas en el modelo de datos creado previamente, se decide que se trabajará con registros simulados en el cliente REST para realizar pruebas ya que la creación de este módulo está prevista en un sprint más adelante.	Se trabajará en los métodos de actualización y eliminación en la entidad de póliza de cumplimiento y en el diseño de la capa de recursos para registro de seguros.	N/A
		Danilo Ariza	Se trabajó en la consulta de los valores del SOAT según categoría del vehículo, se crea la integración de la entidad con la capa de recursos	Se trabajará en las pruebas de registro de SOAT con un cliente REST enfocadas a que se consulte el valor del SOAT correctamente. Se diseñará el controlador y el servicio en Angular.	N/A
	25	Héctor Rodríguez	Se trabajó en la integración de la entidad con la capa de recursos del registro de póliza de cumplimiento	Se trabajará en las pruebas de registro de póliza de cumplimiento con un cliente REST y en el diseño del controlador que se comunicará con el servicio en angular. Se creará el formulario HTML de registro de póliza de cumplimiento	N/A
		Danilo Ariza	Se definió el controlador para el registro de SOAT y se diseñó el servicio en Angular	Se trabajará en el frontend del formulario de registro de SOAT y se realizarán pruebas del flujo de registro	N/A
	27	Héctor Rodríguez	Se definió el controlador para el registro de póliza de cumplimiento y se diseñó el servicio en Angular, se creó el formulario HTML ya con estilos visuales de registro de póliza de cumplimiento.	Se realizarán pruebas de registro de pólizas de cumplimiento desde el formulario y se hará validaciones de los documentos guardados en la base de datos.	Los eventos keycode y keypress no funcionan correctamente en los dispositivos móviles
		Danilo Ariza	Se realiza el formulario HTML con estilos visuales, se realizaron pruebas de los documentos obtenidos en la base de datos	Se realizarán pruebas de registro de SOAT desde el formulario y se validarán los documentos guardados en la base de datos.	N/A
		Fecha	**Review**	**Retrospective**	
		marzo 27	Se realizarán pruebas de registro de SOAT desde el formulario y se validará los documentos guardados en la base de datos.	Se evidencia que las soluciones creadas en los sprint anteriores como listas paramétricas, sessionStorage y manejo de errores, permitió realizar un desarrollo más ágil de las nuevas funcionalidades. Se realiza un revisión de las historias de este sprint teniendo en cuenta los errores encontrados anteriormente y se determina que no es necesario redefinir historias. El estar más familiarizados con la herramienta de control de SCRUM y la comunicación constante con el equipo de trabajo permitió que este sprint finalizará en el tiempo acordado. Los problemas que se presentaron permitieron conocer más funcionalidades de los frameworks utilizados.	

Tabla 21:Sprint 4

Fecha		Integrante	¿Qué se ha trabajado?	¿Qué se va a trabajar?	Impedimentos encontrados
		Héctor Rodríguez	Teniendo en cuenta modelo de datos y la necesidad detectada en el sprint pasado, se definen los JSON de prueba para las compañías aseguradoras	Teniendo en cuenta la definición de los JSON, se va a iniciar con la definición de la entidad compañía en el servidor	NA
	29	Danilo Ariza	Se definió el JSON de prueba para pólizas de seguro todo riesgo de vehículos	Teniendo en cuenta el JSON definido, se procederá con la declaración de la entidad Todo riesgo, además se agrega los índices necesarios en cloudant para la recuperación de información de este tipo de pólizas	NA
		Héctor Rodríguez	Se define el método registrar compañía en la entidad, incluyendo mensajes informativos y códigos http	Se integrará la entidad con la capa de recursos, además se creará servicio en angular para crear registrar compañías	NA
Marzo	31	Danilo Ariza	Se definió el método para registro de póliza todo riesgo teniendo en cuenta la forma de trabajo utilizada en el sprint pasado, además se agrega índices en cloudant y se prueban mediante "Queries" desde el dashboard de cloudant	Se integrará entidad con capa de recursos y se agregará mensajes informativos y códigos http	NA
		Héctor Rodríguez	Se integró entidad con capa de recursos, se hicieron pruebas iniciales con cliente REST, todo funciona bien	Se creará controlador y vista para registro de compañías	NA
	2	Danilo Ariza	Se realizó la integración de la entidad todo riesgo con la capa de recursos del servidor, se realizan pruebas y se encuentra que hacían falta algunas validaciones	Se incluirán las validaciones faltantes y se harán las pruebas pertinentes, adicionalmente se iniciará con la creación del servicio en angular	NA
Abril	4	Héctor Rodríguez	se finaliza con el controlador y el formulario de registro de	Se agregará estilos al formulario con bootstrap y se iniciará	

			la compañía	con la definición del método en backend para registro de valores de SOAT	
		Danilo Ariza	Se definió el método para registro de póliza todo riesgo teniendo en cuenta la forma de trabajo utilizada en el sprint pasado, además se agrega índices en cloudant y se prueban mediante "Queries" desde el dashboard de cloudant	Se iniciará con la creación del controlador en angular para póliza de seguro todo riesgo	Teniendo en cuenta que todo riesgo puede ser diferida, no se tiene claro si en todo el ciclo de vida de la póliza se pueda editar las cuotas
		Héctor Rodríguez	Se inició con la creación del método para registrar valores de SOAT; se concluyó la tarea de registro de compañías de forma exitosa, adicionalmente se incluyó la opción de activar e inactivar compañías desde el mismo módulo	Se finalizará con las validaciones faltantes en valores SOAT y se incluirá mensajes y códigos http; además se creará servicio de angular teniendo en cuenta la plantilla que se ha trabajado a lo largo de los sprints	NA
	6	Danilo Ariza	Se finalizó con el controlador y una versión inicial de la vista para su registro, se utilizó como base las plantillas de los seguros trabajados en el sprint anterior, además se aclaró con cliente el tema de las cuotas, se define que se pueden cambiar mientras las pólizas no se encuentre en estado "Emitido"	Se incluirá validaciones en campos del formulario, además se agregará estilos para que concuerde con apariencia de la aplicación	NA
		Héctor Rodríguez	Se finalizó con lo pertinente a backend para valores SOAT	Se procederá a crear controlador y vista para registro desde aplicación web	NA
	8	Danilo Ariza	Se finaliza con validaciones de campos y con inclusión de estilos en vista	Se realizarán pruebas generales desde aplicación con el módulo de todo riesgo para cerrar tareas	Se encuentra que en ocasiones por alguna razón se demora en traer las listas paramétricas cuando son varias
	10	Héctor Rodríguez	Se finalizó con el módulo de registro valores SOAT	Se realizarán pruebas del módulo para comprobar su correcto funcionamiento y cerrar historia de	En la creación del formulario se buscó utilizar la lista paramétrica de

				usuario	categorías de vehículos para construir el formulario dinámicamente con ng-repeat y asignando de una vez ng-model, no se logró por lo que se realiza manualmente dicho formulario
		Danilo Ariza	Se finalizará con las validaciones faltantes en valores SOAT y se incluirá mensajes y códigos http; además se creará servicio de angular teniendo en cuenta la plantilla que se ha trabajado a lo largo de los sprints	Se corregirá errores de controlador y se repetirá las pruebas en el módulo para cerrar historia de usuario	NA
		Fecha	**Review**	**Retrospective**	
		abril 10	Se entrega tanto el módulo administrativo como el de registro de pólizas de cumplimiento, se integran de manera correcta con la aplicación	Gracias a ciertos esquemas definidos en el sprint anterior, como la plantilla para servicios en angular y los métodos de registro de pólizas en backend, el trabajo fue un poco más fluido. Salvo los problemas ocurridos durante el sprint, no hay nada más para destacar.	

Tabla 22:Sprint 5

Fecha		Integrante	¿Que se ha trabajado?	¿Qué se va a trabajar?	Impedimentos encontrados
		Héctor Rodríguez	Se ha trabajado en cuales y como se va a realizar la adaptación de los métodos backend para el perfil cliente	Se trabajará en la creación de un perfil cliente con ciertas restricciones de funcionalidades	N/A
	12	Danilo Ariza	Se ha trabajado en el diseño de los posibles estados que pueden presentarse en el proceso de compra de seguros y se crea una lista paramétrica con estos estados.	Se trabajará en implementar las validaciones de los estados en los procesos que ya existen en el registro, actualización y eliminación de los 3 tipos de seguros.	N/A
		Héctor Rodríguez	Se crea un perfil cliente con restricciones en la modificación de información de solicitud de pólizas según los estados y validaciones que se crearon en la lista paramétrica y en los métodos del backend	Se trabajará en la creación de los formularios de registro y solicitud de pólizas la aplicación móvil por medio del framework ionic	N/A
	14	Danilo Ariza	Se implementan las validaciones de los estados en las entidades de cada tipo de seguro y se empieza con el diseño del módulo de actualizaciones de estados de una póliza	Se trabajará en el módulo de actualización de estado de una póliza y el módulo de actualización de información de una póliza.	N/A
		Héctor Rodríguez	Se termina la adaptación de los formularios por medio de ionic para la aplicación móvil, se trabajó en el diseño de una pantalla informativa con los tipos de pólizas ofrecidas al cliente.	Se trabaja en las pruebas de solicitud de pólizas por medio de la app y el flujo que debe seguir una solicitud por medio de la app.	N/A
	16	Danilo Ariza	Se trabajó en los métodos de actualización de estados de una póliza, restringiendo esta modificación según perfil, se habilita solo cuando es administrador dentro de la consulta del producto. Se realizan los métodos de consulta de una póliza y se condiciona la modificación de información solo para el perfil administrador.	Se trabajará en el diseño de la entidad pagos comenzando con el diseño del documento JSON que contendrá la estructura de dicha entidad.	N/A
		Héctor Rodríguez	Se realizan las pruebas de solicitud de pólizas por medio de la app, se encuentran varios defectos que son discutidos con el equipo de trabajo.	Se trabajará en la corrección de los defectos encontrados en las pruebas de solicitud de pólizas por medio de la app.	Al realizar las pruebas de solicitud por medio de la app se encuentran errores en la adaptación de la conexión del backend con el nuevo frontend, se decide que es necesario crear un nuevo recurso de conexión (ruta), que sea validado en los servicios y permita realizar control de funcionalidades disponibles para el cliente de la app.
Abril	18	Danilo Ariza	Se finaliza con el diseño del documento JSON para pagos y se inicia con la construcción de la	Se trabajará en terminar la construcción de la entidad de pagos y se validarán los	N/A

			entidad pagos con la definición de los métodos que contendrá.	estados de pólizas existentes ya que en la reunión se encontraron problemas en la adaptación de los servicios en la parte móvil y los estados planteados inicialmente pueden verse afectados.	
	20	Héctor Rodríguez	Se termina con la construcción del nuevo recurso de conexión el cual permite dividir las peticiones desde la parte web y la parte móvil, se comienza con la implementación de este nuevo recurso en los servicios ya existentes creando condiciones dentro de los mismos.	Se trabajará en la terminación de las condiciones de conexión que tienen que ir de la mano con el perfil cliente.	N/A
		Danilo Ariza	Se termina la construcción de la entidad pagos y se procede a realizar la integración de la entidad con la capa de recursos. Se revisan los estados creados anteriormente en la lista paramétrica y se crea un nuevo estado "Solicitado por el Cliente" que se usará exclusivamente para la aplicación móvil.	Se terminará la implementación de la entidad en la capa de recursos y en la creación del nuevo servicio de pagos.	N/A
	22	Héctor Rodríguez	Se terminan las condiciones de conexión desde la app y se basan las condiciones de consulta y modificación de pólizas al nuevo estado de pólizas creado en la lista paramétrica, se inician las pruebas de solicitud de pólizas desde la app	Se trabajará en las pruebas de solicitud de pólizas desde la app móvil para validar las acciones realizadas para corregir los errores encontrados en las pruebas iniciales.	N/A
		Danilo Ariza	Se trabajó en el servicio de pagos, se crearon los nuevos índices necesarios para las consultas en cloundant y se realizaron pruebas de consultas desde la base de datos, adicionalmente se comenzó la construcción del controlador en angular.	Se trabajará en los nuevos códigos de error necesarios para los pagos en el utilitario que se tiene para este fin y en la adaptación del controlador con el servicio, también en la construcción del nuevo formulario web.	Al realizar el servicio de pagos, se encontró que es necesario manejar estados de pagos y que estos puedan ser cambiados por un administrador, esto no se había contemplado anteriormente.
	24	Héctor Rodríguez	Se termina las pruebas y se evidencia que la solución planteada corrige el problema de permisos en las funcionalidades, se trabajó en terminar las validaciones de los campos del nuevo formulario en ionic y se le agregan estilos visuales al formulario.	Se trabajará en probar de nuevo el flujo ya con las validaciones en los campos de formulario, así mismo en la entrega de una primera versión funcional de un. apk que pueda ser probada en varios dispositivos Android.	N/A
		Danilo Ariza	Se corrige el inconveniente encontrado realizando un control de estados de pagos de forma similar como se hace con los estados de la póliza, se crea una lista paramétrica con los estados de los pagos y en el servicio se realiza el método de cambio de estado de pagos, se finaliza con el formulario web y la adaptación del	Se trabajará en las pruebas del flujo de pagos de una póliza por medio del formulario web y se entregará una nueva versión de la aplicación web con el servicio de pagos funcional para el perfil administrador.	N/A

			controlador con el servicio, siguiendo el esquema planteado en los sprint anteriores.		
		Fecha	**Review**	**Retrospective**	
		abril 24	Se entrega una versión funcional de la aplicación web con el servicio de pagos y con la posibilidad de actualizar información de una póliza ya solicitada. Se entrega la primera versión funcional de la aplicación móvil, que muestra información de las pólizas disponibles y a su vez le permite a un cliente hacer la solicitud de una de estas pólizas.	Se aplica el esquema de trabajo de los anteriores sprint para la construcción de los nuevos servicios lo que permite reducir tiempos de desarrollo teniendo en cuenta los atrasos provocados en el pasado. Se presentaron inconvenientes en el entendimiento de los nuevos frameworks utilizados, pero fueron discutidos y resueltos en el transcurso del desarrollo por el equipo de trabajo y no provocaron mayores complicaciones. Finalmente se define un estilo visual para la app el cual llevaran los nuevos servicios que se implementen	

Tabla 23:Sprint 6

Fecha		Integrante	¿Que se ha trabajado?	¿Qué se va a trabajar?	Impedimentos encontrados
Abril	26	Héctor Rodríguez	Se ha trabajado en la creación de los métodos de modificación de clientes en las entidades relacionadas para la aplicación web.	Se va a trabajar en la implementación de los nuevos métodos en los servicios y controladores relacionados	N/A
		Danilo Ariza	Se ha trabajado en la creación del formulario de consulta de productos de un cliente (mis productos) para la aplicación móvil en ionic.	Se va a trabajar en aplicar el estilo visual acordado con las validaciones necesarias en los campos del formulario web	N/A
	28	Héctor Rodríguez	Se ha trabajado en los controladores y servicios afectados con los nuevos métodos de modificación de información de clientes	Se trabajará en la creación del nuevo formulario de información del cliente que permita la modificación de información	N/A
		Danilo Ariza	Se terminó el formulario de la consulta de productos del cliente en la aplicación móvil con el estilo visual acordado.	Se trabajará en la creación de los métodos necesarios para la consulta de productos del cliente basados en el recurso de conexión creado anteriormente y así mismo en el perfil cliente.	N/A
	30	Héctor Rodríguez	Se ha trabajado en el formulario de consulta de cliente con el estilo visual acordado en la aplicación web.	Se trabajará en la adaptación de los métodos del backend con el nuevo formulario de consulta y modificación de la información del cliente.	N/A
		Danilo Ariza	Se ha trabajado en la adaptación de los métodos de consulta creando las validaciones necesarias basadas en el perfil cliente creado para la aplicación móvil	Se trabajará en la creación del controlador para este nuevo servicio solo disponible para la aplicación móvil.	N/A
Mayo	2	Héctor Rodríguez	Se trabajó en la adaptación del frontend con el backend creado anteriormente para la modificación de información del cliente.	Se trabajará en terminar la adaptación y se comenzaran a realizar las pruebas	N/A
		Danilo Ariza	Se trabajó en la construcción del controlador en angular para la consulta de productos del cliente restringido al perfil de cliente creado anteriormente solo para la aplicación móvil.	Se trabajará en crear el nuevo servicio de consulta basados en el controlador ya creado	N/A
	4	Héctor Rodríguez	Se realizan las pruebas de la consulta de la información de clientes ya creados. Se encuentran formatos erróneos en los campos del formulario creado anteriormente.	Se trabajará en las pruebas de modificaciones de información de un cliente, y se agregaran los formatos faltantes.	N/A
		Danilo Ariza	Se ha trabajado la construcción del servicio y se realizan pruebas iniciales del servicio basadas en el funcionamiento de esta funcionalidad solo para la aplicación móvil.	Se trabajará en la adaptación del servicio con el controlador de la consulta de productos del cliente desde la aplicación móvil.	N/A
	6	Héctor Rodríguez	Se trabajó en agregar los formatos faltantes en algunos campos del formulario de consulta de cliente y se comienza con las pruebas de modificación de información de clientes ya existentes.	Se trabajará en las pruebas de modificación de información basándose en que los documentos ya creados no pierdan la integridad.	N/A
		Danilo Ariza	Se trabajó en la adaptación del servicio con el controlador para la consulta de productos del cliente desde la aplicación móvil.	Se trabajará en las pruebas desde la app de consulta de productos ya adquiridos por un cliente y se creará	N/A

				directivas de emisión de notificaciones acordes para la aplicación móvil y la aplicación web.	
	8	Héctor Rodríguez	Se realizan pruebas de modificación de información del cliente solo desde la aplicación web, se verifica que la información quede correcta en la base de datos.	Se trabajará en la formación de la nueva versión de la aplicación web con las nuevas funcionalidades implementadas.	N/A
		Danilo Ariza	Se trabajó en las pruebas de consultas de productos del cliente y se crearon nuevas directivas de emisión de notificaciones más acordes a los mensajes desplegables que se tenían antes.	Se trabajará con implementar las nuevas directivas de la notificación en los demás servicios de la aplicación web y en los servicios ya creados en la aplicación móvil, adicionalmente se trabajará en la entrega de la nueva apk con las nuevas funcionalidades.	N/A
		Fecha	**Review**	**Retrospective**	
		mayo 8	Se entregó una versión funcional de la aplicación web que permite modificar información de un cliente ya creado así mismo se entregó una versión funcional de la aplicación móvil con la posibilidad de que un cliente pueda consultar los productos solicitados con anterioridad. Las dos nuevas versiones cuentan con directivas de emisión de notificaciones más acordes a la parte móvil.	Se agilizo en la creación de las nuevas funcionalidades de acuerdo a el esquema planteado en los demás sprint, las pruebas fueron acordes para reducir los inconvenientes que pudieran pausar el desarrollo completo. La adaptación de los servicios ya implementados fue rápida debido a la construcción inicial del software.	

Tabla 24: Sprint 7

Fecha		Integrante	¿Que se ha trabajado?	¿Qué se va a trabajar?	Impedimentos encontrados
Mayo	10	Héctor Rodríguez	Se ha trabajado en la definición del documento JSON que se almacenará en cloudant, teniendo en cuenta que se debe manejar perfil cliente y perfil administrador por las dos aplicaciones	Se finalizará con la definición del documento JSON teniendo en cuenta el incluir además el tipo y número de documento del usuario, además se iniciará con la definición del método login	NA
		Danilo Ariza	Se definieron las reglas y restricciones en cuanto a la edición de la información de vehículos	Se definirá en la entidad vehículo el método que se encargará de validar la información a editar	NA
	12	Héctor Rodríguez	Finaliza la definición del documento usuario y se crea una primera versión del método login solo para validar que concuerde usuario y contraseña	Se complementará el método login teniendo en cuenta el devolver la información de tipo y numero de documento del usuario, con el cual durante la sesión se realizarán consultas de todo tipo, adicionalmente se iniciará con la definición del servicio en angular	NA
		Danilo Ariza	Se finalizó con el método que valida los datos a editar adicionalmente se inició con la adaptación del método que se encarga de la creación de vehículos ya que en este se definió todas las reglas de integridad según el modelo de negocio	Se concluirá la adaptación del método de creación de vehículo para poder integrar con la capa de recursos e iniciar con las pruebas pertinentes	NA
	14	Héctor Rodríguez	Se finalizó con login en backend, se hicieron pruebas de funcionamiento y se definió el servicio en angular, el cual servirá tanto para la aplicación web como para la aplicación móvil	Se implementará los controladores y vistas en ambas aplicaciones	NA
		Danilo Ariza	Se finalizó la adaptación del método, se integró con capa de recursos y se realizaron pruebas del servicio	Se definirá vista y controlador para módulo información vehículo	NA
	16	Héctor Rodríguez	Se finaliza con implementación de vistas en aplicaciones, se realizan pruebas	Se iniciará con la creación del método de registro de usuario	NA
		Danilo Ariza	Se finalizó con la implementación del módulo información y modificación de vehículo	Se iniciará con la implementación del formulario de registro de usuario teniendo en cuenta cuando es sólo para usuarios y cuando es abierto y la integración con el servicio en backend	NA
	18	Héctor Rodríguez	Se implementó método de registro para ambos casos, se integró con capa de recursos	Se apoyará la integración de backend y frontend para registro de usuario	
		Danilo Ariza	Se finalizó con la construcción de los componentes frontend para registro de usuario	Se integrará con backend y se realizaran pruebas de funcionamiento	

		Fecha	**Review**	**Retrospective**	
		mayo 22	Se entrega versión funcional en la cual ya se incluyen las historias de usuario definidas para el prototipo	En este sprint el trabajo fue mucho más fluido ya que las funcionalidades más complejas que estaban involucradas eran adaptaciones de lo trabajado en Sprint pasados	