Universidad de Oriente
Núcleo de Nueva Esparta
Escuela de Ingeniería y Ciencias Aplicadas
Departamento de Informática

INVENTAME DATA ACCESS LAYER

Trabajo de Grado Modalidad Tesis de Grado, presentado como requisito parcial para optar al Título de Licenciado en Informática.

Autora:
Br. Yulialci Nohemy Adrián Hernández

Guatamare, Marzo de 2017

Universidad de Oriente
Núcleo de Nueva Esparta
Escuela de Ingeniería y Ciencias Aplicadas
Departamento de Informática

INVENTAME DATA ACCESS LAYER

Autora: Br. Yulialci Nohemy Adrián Hernández. C.I.: 22.996.846

En concordancia con el artículo 13 del reglamento de trabajo de pregrado de la Universidad de Oriente se levanta la presente acta una vez presentado y defendido públicamente el trabajo en referencia. Los miembros del jurado han acordado APROBADO MENCIÓN HONORÍFICA
Se recomienda su Publicación

Para dar fe de lo expresado anteriormente firman:

Msc. Braumalis Malavé
C.I.: 12.673.143

Dra. Suhail Zabala
C.I.: 12.067.253

Ing. Eyamir Ugueto
C.I.: 8.730.054

Guatamare, marzo 2017

ÍNDICE GENERAL

pp.

LISTA DE CUADROS

Cuadro **pp.**

LISTA DE GRÁFICOS

Gráfico **pp.**

DEDICATORIA

Para mis padres por su apoyo, consejos, comprensión, amor, ayuda en los momentos difíciles, y por ayudarme con los recursos necesarios para estudiar. Me han dado todo lo que soy como persona, mis valores, principios, empeño y perseverancia.

Todo este trabajo ha sido posible gracias a ellos.

AGRADECIMIENTOS

Principalmente a Jehová por darme las fuerzas y el discernimiento para alcanzar esta meta. A mis padres por su apoyo incondicional en sentido físico, emocional, moral, espiritual y económico.

A la Prof. Eyamir Ugueto por su dedicación y orientación a lo largo de la realización y culminación de éste trabajo de grado.

De igual manera, al Lcdo. Rodrigo Navarro por su paciencia y el tiempo dedicado para orientación y guía en el uso de las herramientas de programación y el comportamiento de los distintos componentes de Inventame Data Access Layer.

Universidad de Oriente
Núcleo de Nueva Esparta
Escuela de Ingeniería y Ciencias Aplicadas
Departamento de Informática

INVENTAME DATA ACCESS LAYER

Autora: Br. Yulialci Nohemy Adrián Hernández
Asesor Académico: Ing. Eyamir Ugueto
Fecha: Marzo 2017

RESUMEN

El propósito de esta investigación fue brindar soporte a las herramientas desarrolladas por Inventame Group LLC, tal es el caso de Inventame Studio, sistema que está siendo desarrollado para diseñar y crear aplicaciones móviles, las cuales requieren de acceso a servidores en la nube para el almacenamiento masivo de datos. En vista del crecimiento en la gestión de los datos se brinda soporte a estas herramientas utilizando la programación en capas, pues se necesitaba de una capa de acceso a datos que soportara esta expansión en los mismos. Es por eso que, se hizo necesario utilizar una base de datos escalable horizontalmente, denominadas NoSql, particularmente Cassandra bajo a una estructura llave-valor. Para el control de los datos almacenados en Cassandra se utilizó base de datos relacional (MySQL) para guardar la estructura de los namespaces, los cuales contienen esquemas y estos a su vez columnas. Siendo los valores de las columnas los almacenados en Cassandra. Para la gestión de los mismos, el núcleo de Inventame Data Access Layer consiste en una API REST desarrollada en Node JS (lenguaje de programación utilizado por la empresa), documentada al estilo Swagger; la cual permite al usuario gestionar sus datos por medio de endpoints. Además se diseñó una interfaz web en React JS bajo la metodología SCRUM para que los usuarios que no conocen de programación (clientes) gestionen sus datos. La investigación es proyectiva basada en Hurtado (2000), La metodología de desarrollo utilizada fue SCRUM desarrollada por Schwaber y Sutherland en el año 2011.

Descriptores: Cassandra, MySQL, API REST, SCRUM, Node JS, React JS, Web

INTRODUCCIÓN

Actualmente, el éxito de una empresa no depende sólo de cómo maneje sus recursos materiales, sino de que exista un adecuado flujo de información entre la empresa y su entorno. Por eso es necesario que la misma se encuentre organizada para su manipulación. En vista de que ésta aumenta a lo largo del tiempo, se necesitan herramientas informáticas que soporten cantidades masivas de datos y faciliten la organización de los mismos.

Debido a ello, Inventame Group LLC se ha enfocado en desarrollar herramientas que buscan facilitar el manejo de grandes volúmenes de información. A su vez, para lograr ese objetivo, la empresa se vale de la programación en capas. Sin embargo, en vista del exponencial crecimiento de información se ha observado la necesidad de utilizar nuevas tecnologías en las que el crecimiento no sea una limitante, como es el caso de las base de datos escalables horizontalmente.

Como resultado, surgió la necesidad de la presente investigación, que contribuye a superar lo antes planteado a través de una capa de acceso a datos denominada Inventame Data Access Layer que valiéndose de estas nuevas tecnologías en bases de datos facilita la manipulación de la información a través de una interfaz web de usuario de fácil uso.

El trabajo se encuentra estructurado de la siguiente manera:

Capítulo I. Marco Introductorio, comprendido por el planteamiento del problema, formulación del objetivo general y objetivos específicos, además de la justificación de la investigación.

Capítulo II. Marco Teórico, donde se presentan los antecedentes o investigaciones relacionadas con el trabajo realizado, así como toda la

información o fundamentos teóricos involucrados y que sustentan a la investigación.

Capítulo III. Marco Metodológico, en el cual se desarrollan todos los aspectos referentes a la metodología de la investigación. Tales como: tipo de investigación, diseño de investigación, instrumentos de recolección de datos, y la metodología de desarrollo utilizada.

Capítulo IV. Resultados, en éste se presentan los resultados obtenidos luego de aplicar la metodología para planificación de desarrollo de software implementada en Inventame Group LLC, propuesta por Schwaber y Sutherland (2011), denominada Scrum.

*Capítulo V***.** Conclusiones y Recomendaciones, dedicado a mostrar las conclusiones y recomendaciones obtenidas luego de alcanzados los objetivos previstos para la investigación que pudieran ser pertinentes para el futuro.

CAPÍTULO I

MARCO INTRODUCTORIO

Planteamiento del Problema

La información se ha convertido en un aspecto clave dentro de las empresas, es un elemento no sólo necesario sino también estratégicamente definitorio. Se puede decir que no es un activo más, sino un recurso estratégico. Sin embargo, el volumen de la información dentro de las mismas aumenta a nivel exponencial. Así lo evidencia la ley de rendimientos acelerados, en la que se explica que el crecimiento exponencial tecnológico, conlleva cierto tipo de barrera por lo que se inventará otra tecnología para permitir cruzar esa barrera (Kurzweil, 2012, p. 25). En concordancia, se hace necesario mejorar las herramientas existentes para procesar todo ese cúmulo de datos y más importante aún para organizarlos, debido a que al necesitarlos debe ser sencillo ubicar lo que se desea. De hecho, aparecen procesos y tecnologías nuevas que buscan suplir las necesidades del manejo de información (Sierra, 2006, p.3).

En otras palabras, nacen nuevos conceptos aplicados al manejo de grandes volúmenes de información. De hecho, según Vargas y Maltés (s.f., p.2) se: “necesitará seguir una serie de pasos complejos los cuales deben ser definidos para cada proyecto; haciendo uso de la programación en capas”.

En términos de programación, los modelos o la programación en capas consisten en separar la lógica de negocios de la lógica de diseño, es decir, separar la capa de datos de la capa de usuario. Una de sus

características es que cada capa está totalmente abstraída del resto y para poder comunicarse entre ellas deben utilizar una Interfaz de Programación de Aplicaciones (API). Cabe resaltar que, existen diversos modelos de en capas, pero todos requieren de presentación, de lógica de negocio y de datos (Cardador, 2014, p.12).

Para esta investigación se analiza la importancia de la capa de datos, pues depende de ella que se logre el éxito en el manejo de cúmulo de datos para las empresas. Ésta es la encargada de realizar transacciones con bases de datos y con otros sistemas para obtener o ingresar información al sistema. El manejo de los datos debe realizarse de forma tal que haya consistencia en los mismos, de tal forma los datos que se ingresan y extraen deben ser consistentes y precisos. Es en esta capa donde se definen las consultas a realizar para la generación de reportes más específicos (Vargas y Maltés, s.f., p.3).

Sin embargo, la capa de datos requiere de un sistema gestor de base de datos, el cual según Sierra (2006, p.4): “es un programa que permite introducir información almacenarla, ordenarla y manipularla en general”. Comúnmente las base de datos son relacionales, pero cuando es necesario manejar volúmenes tan grandes de información en diversos formatos aparecen problemas de gestión y de escalabilidad. Tal es el caso de las aplicaciones web y las móviles; pues éstas requieren de acceso a servidores en la nube para el almacenamiento de datos y mientras más datos se manipulen los tiempos de respuesta a la hora de obtenerlos son mayores. Es allí cuando surgen las bases de datos no relacionales que de acuerdo a Díaz (2013): “no se ajustan al modelo de bases de datos relacionales y sus características, pues no tienen esquemas y escalan horizontalmente; así hacen uso amplio de la memoria principal del computador y resuelven el problema de los altos volúmenes de información”. Por ejemplo, hay empresas que desarrollan software para manejar un alto volumen de

información a nivel empresarial e incluso herramientas para desarrolladores, con el objetivo de que la gestión de datos se realice de manera sencilla.

Una de las compañías que ofrece dar solución a este tipo de usuarios (empresas y desarrolladores independientes) es Inventame Group LLC, la cual se inició en Margarita, Estado Nueva Esparta pero que actualmente funciona en Chile. En entrevista no estructurada realizada (Marzo 2, 2016) al Licenciado Rodrigo Navarro, director ejecutivo de la empresa, se obtuvo mayor información acerca de la misma y sus objetivos. Esta compañía busca facilitar el manejo de grandes volúmenes de información de manera rápida, sencilla y a un bajo costo. Debido a ello, sus productos están basados principalmente en un solo lenguaje, JavaScript; pues éste ofrece la posibilidad de desarrollar la lógica una sola vez, y facilita el poder compartirla en distintas plataformas con sus respectivas interfaces de diseño. Además, la empresa se basa en la programación en capas, ya que permite que en el desarrollo de nuevos productos se reutilice la lógica utilizada en productos existentes.

El principal producto de la empresa es Inventame Studio, el cual consiste en un sistema que está siendo desarrollado para diseñar y crear aplicaciones móviles. En vista del exponencial crecimiento de este tipo de aplicaciones y el volumen de datos que se maneja por Inventame Studio se observó la necesidad de utilizar base de datos escalables horizontalmente, pues de esta forma no hay limitaciones de crecimiento. Sin embargo, como era necesario tener control sobre los datos y facilitar la manipulación de los mismos, se requería registrar la estructura de los esquemas creados, haciendo uso de una base de datos relacional.

Al observar estas necesidades, se planteó el desarrollo de un nuevo producto, el cual consiste en una capa de acceso a datos que permitiera crear y manipular esquemas a través de una interfaz web de usuario de fácil uso; facilitando la manipulación de cantidades masivas de datos sin requerir

un conocimiento especializado de programación. Esta sirve de complemento a otros servicios que necesitan de gran manejo de datos (Big Data).

Este producto tiene control sobre los datos y facilita la manipulación de los mismos, registrando la estructura de los esquemas en la base de datos relacional, particularmente se planteó el uso de MySql; lo que permite que los datos almacenados en la base de datos no relacional se puedan organizar, cotejar y utilizar en otras aplicaciones. Dicha estructura de datos se crea en tiempo de ejecución a partir de lo que indique el usuario. Para el almacenamiento de los datos, se planteó el uso de Apache Cassandra que es una Base de Datos no relacional (NO SQL), distribuida y basada en un modelo de almacenamiento de Clave-Valor, escrita en Java.

En síntesis, la finalidad era que los datos se ingresen una sola vez, de una forma determinada y luego pudieran manipularse para extraer la información ordenada y seleccionada por múltiples criterios, valiéndose de la estructura guardada en la base de datos relacional y sus correspondientes datos guardados en la base de datos no relacional. Así, facilitar la generación de información para los usuarios; ya que podrían organizar y analizar la misma de acuerdo a los criterios deseados. De hecho, para la creación de dicha información, se genera una API que permite al usuario a través de URL utilizando punto de entrada (Entry Point) realizar búsquedas por un criterio determinado, hacer uso de filtros y manipular los datos como desee.

Además, se permite el manejo flexible de diversos tipos de datos. De hecho, el usuario puede importar datos de manera organizada en el formato que desee, entre éstos se comprenden los siguientes: CSV, EXCEL, XML y JSON. Una vez importados los datos la aplicación se encarga de crear el espacio de claves y el esquema inmerso en ese archivo. En vista de que, no existía esta capa de acceso a datos que permitiera manipular grandes volúmenes de datos, no se podía ofrecer este servicio a las empresas que lo requieren, ya que esta capa junto con la base de datos en Cassandra vienen

siendo un punto clave en los productos que esta empresa ofrece, como es el caso de Inventame Studio. Sin embargo, esta capa de acceso a datos no sólo brinda servicios a Inventame Studio sino que podrá ser utilizado en las distintas aplicaciones o productos que la empresa comercialice.

Por lo tanto, de acuerdo a la situación planteada por la empresa Inventame Group LLC, se desarrolló la capa de acceso a datos Inventame (Data Access Layer) para que las organizaciones sin conocimientos de programación puedan gestionar grandes volúmenes de datos y utilizarlos o manipularlos como deseen.

Objetivos de la Investigación

Objetivo General

Desarrollar Inventame Data Access Layer para el manejo de datos de los servicios brindados por la empresa Inventame Group LLC.

Objetivos Específicos

1. Determinar los requerimientos del software que permita el armado del documento backlog.
2. Establecer la lista de tareas y los objetivos principales para organizar en Sprints.
3. Crear estructura de espacio de claves en el manejador de base de datos Cassandrav3.2.1 y estructura de esquemas en el manejador de base de datos MySql 5.7.12
4. Diseñar la interfaz web utilizando la metodología SCRUM desarrollada por Schwaber y Sutherland en el año 2011.
5. Codificar la aplicación web haciendo uso de JavaScript, HTML5, CSS3 y Node JS v4.4.5LTS.

Justificación

La cantidad de datos que se genera actualmente es abrumadora. De hecho, se recolectan grandes cantidades de información de diversas áreas, tales como: empresariales, sociales y gubernamentales. Debido a ello, es necesario almacenar y administrar los datos de forma que no se pierda información y se mantenga un buen rendimiento. Para lograr esto, se requiere de la capacidad de escalar horizontalmente usando la menor cantidad de recursos. Las bases de datos que cumplen con este requerimiento se denominan NoSql, éstas brindan una solución factible cuando se tienen grandes cantidades de información; pues optimizan de mejor manera la búsqueda de datos sin hacer uso del lenguaje SQL para ello.

En vista de que la empresa Inventame Group LLC desarrolla software para manejar estos volúmenes de información a nivel empresarial, incluidas el desarrollo de aplicaciones móviles como es el caso de Inventame Studio se hizo necesario el desarrollo de esta capa de acceso a datos que facilita la manipulación de cantidades masivas de datos a través de interfaz web y que además sirve de complemento a otros servicios para el usuario; combinando para ello el uso de base de datos Sql y NoSql.

Además, con esta investigación se busca apoyar el aprendizaje de las nuevas tendencias en la gestión de la información, fomentando de esta manera el campo de investigación en la comunidad universitaria, contribuyendo a realizar análisis más detallados de los distintos casos que se puedan presentar y que no puedan ser resueltos mediante bases de datos relacionales, fortaleciendo así la temática para posteriores vinculaciones en proyectos de investigación o trabajos afines.

CAPÍTULO II.

MARCO TEÓRICO

Antecedentes

Al momento de realizar cualquier tipo de investigación es de gran importancia hacer una revisión de trabajos vinculados en el área de estudio, pues éstos servirán de guía al momento de establecer bases teóricas, metodológicas y técnicas. Además, este proceso es fundamental para adentrarse en el tema de estudio. Debido a ello, la revisión referencial permitió encontrar diferentes trabajos sobre la programación en capas y base de datos no relacionales (NoSQL) realizados a nivel internacional, que sirvieron de soporte a este trabajo de investigación:

Daián, López y Florencia (2015) desarrollaron un trabajo de grado titulado "Utilización de NoSQL para resolución de problemas al trabajar con cantidades masivas de datos – Buenos Aires, Argentina". El mismo estudia el comportamiento de las bases de datos NoSQL, para conocer el impacto que tienen en cantidades masivas de datos. Para ello, utilizan datos correspondientes a compras realizadas por personas, datos médicos de pacientes, sus historiales y relaciones con médicos, y datos de redes sociales. Se realizó un estudio de campo con la metodología GEMIS (Metodología de la Ingeniería del Software desarrollada en Buenos Aires, Argentina) para ver en qué casos se obtiene mejor resultado utilizando cada una de las tecnologías NoSQL (Clave-Valor-Redis, Familia de Columnas-Cassandra, Documentales-MongoDB y Gráficas-Neo4J). Este estudio es de gran relevancia para la investigación puesto que muestra cómo la separación

por capas de las partes de las aplicaciones permite introducir NoSQL en una aplicación preexistente, pudiendo además coexistir arquitecturas SQL y NoSQL, aprovechando las ventajas de cada una de ellas en lo que se conoce como persistencia políglota; siendo esto lo que se planteó en el caso de estudio. Sin embargo, se utiliza una metodología diferente conocida como GEMIS, mientras que en la investigación se utilizó SCRUM.

Barragan y Forero (2013) realizaron un trabajo de grado titulado "Implementación de una base de datos NoSQL para la Generación de la Matriz O/D – Bogotá, Colombia". Éste consiste en la generación de la matriz O/D, encargada en el sector transporte de estimar el flujo de tránsito de un punto de origen a un punto de destino, utilizando base de datos NoSQL. Para ello, generaron un modelo propio de base de datos NoSQL transformando un modelo relacional a uno no relacional por medio del proceso de desnormalización. La investigación es de campo y se vale de la programación extrema (XP) como metodología de desarrollo. Su relevancia radica, en que muestra como ajustar esta tecnología NoSQL para optimizar los resultados de las consultas desde la etapa de diseño. Sin embargo, el proceso consiste en transformar un modelo relacional a uno no relacional; mientras que en la presente investigación estos dos modelos son utilizados, el relacional para almacenar la estructura de los esquemas y el no relacional para el almacenamiento de los datos. Cabe destacar que, aunque difiere en la metodología de desarrollo, ésta se encuentra en el grupo de metodologías ágiles al igual que SCRUM, metodología utilizada en la presente investigación.

Campoverde (2012) realizó un trabajo de grado titulado "Desarrollo de Aplicación Web mediante HTML5 y la base de datos NoSQL MongoDB – Cuenca, Ecuador". Consiste en el desarrollo de la aplicación web PreServi, una red social orientada a compartir y ofrecer servicios que puedan prestar los usuarios registrados en ésta; utiliza HTML5, agregando funcionalidades

de JavaScript y añadiendo el gestor de base de datos NoSQL MongoDB, para solventar limitaciones del modelo relacional al trabajar con cantidades masivas de datos. La investigación es de tipo proyectiva y la metodología de desarrollo es OOHDM. En el trabajo se resalta la importancia de considerar las bases de datos NoSQL al manejar gran cantidad de datos con grandes volúmenes de escritura junto con una arquitectura basada en capas. La relevancia de esta investigación radica en que muestra cómo utilizar base de datos no relacional en una estructura Big Data, en este caso una red social; haciendo uso de herramientas como JavaScript y HTML5 que también fueron utilizadas en el desarrollo de Inventame Data Access Layer. Sin embargo, difiere de esta investigación en la metodología de desarrollo.

Pérez (2010) desarrolló un trabajo de grado titulado "Propuesta de análisis y diseño basada en UML y UWE para la migración de arquitectura de software centralizada hacia Internet – San Carlos, Guatemala". En este trabajo se plantea la migración de datos de un sistema de contratos de personal a otra arquitectura de software centralizada, esto debido al crecimiento del sistema. Para ello, se vale de la arquitectura de tres capas: capa de presentación, capa de negocio y capa de datos. Se realizó un estudio de campo y se utiliza UML y UWE para el diseño de la interfaz de la aplicación web, sin embargo plantea el uso de una base de datos relacional. En las conclusiones del trabajo se evidencia que el desarrollo en capas de esta propuesta facilita el desarrollo de la aplicación final, ya que permite que los servicios puedan ser reutilizados y compartidos entre múltiples aplicaciones. Esta investigación muestra cómo realizar migración de datos a través de la programación en capas e incluso explica cómo un servicio puede ser reutilizado en otras aplicaciones, siendo éste el caso de Inventame Data Access Layer. Como diferencia con la presente investigación está la metodología de desarrollo UWE y el que no utiliza base de datos no relacional.

Márquez y De La Cruz (2010) desarrollaron un trabajo de grado titulado “Interfaz de monitoreo para máquinas de hemodiálisis Fresenius - México”. Consiste en una interfaz web para que el personal de un hospital pueda hacer el seguimiento clínico continuo de un paciente con insuficiencia renal a través de una aplicación que trasmite los parámetros fisiológicos a través de la red, lo que favorece la monitorización en tiempo real. Para su desarrollo utiliza un modelado de tres capas (presentación, negocio y datos), en el cual la capa de presentación está representada por la interfaz y el uso de la tecnología AJAX en el momento de recepción continua de los datos (parámetros fisiológicos), por otra parte, la capa de negocios para esta investigación se representa con PHP y en la capa de datos se utiliza MySql como gestor de base de datos. El estudio es de campo y utiliza la metodología de desarrollo UWE junto con UML. La relevancia de esta investigación radica en que muestra cómo la programación en capas facilita la elaboración de estadísticas, generación de reportes e incluso la aplicación de minería de datos. Sin embargo, difiere de la investigación al utilizar sólo la base de datos relacional y la metodología de desarrollo UWE.

Bases Teóricas

En este apartado se esbozarán conceptos asociados al tema principal del trabajo, enfocado al desarrollo de Inventame Data Access Layer, los cuales en conjunto proveen el sustento teórico referencial del estudio.

Para iniciar es necesario comprender lo que es la computación en la nube (Cloud Computing), ya que la API correspondiente a Inventame Data Access Layer se conecta a servidores en la nube. De acuerdo con Amazon Web Services (AWS) (s.f.) el término Cloud Computing se refiere a: “la entrega bajo demanda de recursos informáticos y aplicaciones a través de Internet con un sistema de precios basado en el consumo realizado”. En

efecto, la computación en la nube ofrece un método sencillo de obtener acceso a servidores, almacenamiento, bases de datos y una amplia gama de servicios de aplicaciones a través de Internet.

Por su parte, Cierco (2011, p.11) explica que según el laboratorio de tecnologías de información: “cloud computing es un modelo que permite el acceso bajo demanda y a través de la red a un conjunto de recursos compartidos y configurables (como redes, servidores, capacidad de almacenamiento, aplicaciones y servicios)”. En otras palabras, se elimina la necesidad de grandes inversiones y costos fijos; por lo que en la presente investigación se trabajó con cloud computing, debido a que fue necesario contar con dos (2) servidores AWS uno para cada tipo de base de datos (Relacional y No Relacional).

En concordancia, las bases de datos según Connolly y Begg (2005, p.14) consisten en: “una colección compartida de datos lógicamente relacionados, junto con una descripción de estos datos, que están diseñados para satisfacer las necesidades de información de una organización”. Es decir, que las bases de datos consisten en una serie de datos organizados y relacionados entre sí, los cuales son recolectados y explotados por los sistemas de información de una empresa o negocio en particular. De acuerdo con (ob. cit., p. 24) entre las principales características de los sistemas de base de datos se pueden mencionar:

- Independencia lógica y física de los datos.
- Redundancia mínima.
- Acceso concurrente por parte de múltiples usuarios.
- Integridad de los datos.
- Consultas complejas optimizadas.
- Seguridad de acceso y auditoría.
- Respaldo y recuperación.
- Acceso a través de lenguajes de programación estándar.

Para esta investigación fue necesario trabajar con base de datos relacional (RDBMS) y no relacional (NoSql) para el almacenamiento de: la estructura de los esquemas y los datos de los mismos. Para comprender las diferencias entre éstas se explican sus características a continuación.

Respecto a la base de datos relacional (RDBMS) es una colección o depósito de datos integrados con redundancia controlada y con una estructura que refleje las interrelaciones y restricciones existentes en el mundo real. Los datos, deben mantenerse independientes de usuarios y aplicaciones, y su definición y descripción únicas para cada tipo de dato, han de estar almacenadas junto con los mismos (Piñeiro, 2013, p.3).

En síntesis, las bases de datos relacionales consisten en una colección de elementos organizados en un conjunto de tablas formalmente descritas, desde la que se puede acceder a los datos o volver a montarlos de muchas maneras diferentes sin tener que reorganizar las tablas de la base. En otras palabras, es un conjunto de tablas que contienen datos provistos en categorías predefinidas. Cada tabla contiene una o más categorías de datos en columnas. Cada fila contiene una instancia única de datos para las categorías definidas por las columnas (TechTarget, 2015). Para la investigación se utiliza RDBMS para el almacenamiento de la estructura de los esquemas.

Por su parte, la Base de Datos No Relacional (NoSql) se posiciona como la siguiente generación de tecnologías relacionadas con las bases de datos. Este subconjunto de bases de datos es diferente a varios modelos de bases de datos tradicionales con características muy importantes como por ejemplo, no usa lenguaje de consultas SQL y su forma de escalar es horizontalmente (Fernández, 2015, p. 34).

Éstas han sido diseñadas para manipular grandes volúmenes de datos de manera muy rápida, y no sigue el modelo entidad-relación típico de las bases de datos tradicionales; no requieren una estructura de datos en forma

de tablas y relaciones entre ellas y no imponen un esquema prefijado de tablas. Las bases de datos NoSql son más flexibles, ya que suelen permitir información en otros formatos como clave-valor, mapeado de columnas, documentos o grafos. Son idóneas para aplicaciones que requieran de lectura/escritura de grandes volúmenes de datos y donde se necesita brindar un servicio a miles o millones de usuarios (Joyanes, 2016, p.16)

Las principales características de NoSQL de acuerdo con (ob. cit., p. 18) son las siguientes:

- *Escalabilidad horizontal*: Es la característica que permite aumentar el rendimiento del sistema añadiendo, simplemente más nodos, sin necesidad de realizar ninguna otra operación excepto indicar cuáles son los nodos disponibles.
- *Habilidad de distribución:* Hace referencia a la escalabilidad horizontal que lo caracteriza, pero hace énfasis en su soporte; para ello se tiene en cuenta la habilidad de replicar y distribuir los datos sobre los servidores o nodos.
- *Uso eficiente de recursos:* Aprovecha las nuevas tecnologías, como los discos en estado sólido, el uso eficiente de recursos como la memoria RAM y los sistemas distribuidos en general.
- *Ausencia de esquema:* Los datos no tienen una definición de atributos físicos, es decir, cada registro o documento puede contener una información con diferente formato en cada ocasión, que permite almacenar solo aquellos atributos que interesen, facilitando el polimorfismo de datos bajo una misma colección de información.
- *Alta velocidad:* Realizan las operaciones directamente en memoria y sólo vuelcan los datos en disco cada ciertos períodos de tiempo. De esta forma, las operaciones de escritura son muy rápidas.

Gracias a su flexibilidad y su calidad adecuada de respuesta para grandes cantidades de información; los datos correspondientes a los esquemas se almacenan en base de datos NoSql. Sin embargo, para la interacción entre la base de datos y el usuario es necesario contar con los Sistemas de Gestión de Base de Datos (SGBD).

Los Sistemas de Gestión de Base de Datos (en inglés DataBase Management System) son un tipo de software muy específico, que permite a los usuarios definir, crear, mantener y controlar el acceso a la base de datos. Es el software que interactúa con los programas de aplicación del usuario y con las base de datos. De manera que los mismos, sirven de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. Además, se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta (Conolly y Begg, 2005, p.15). Debido a las necesidades planteadas en la organización se utilizó MySQL (RDBMS) y Apache Cassandra (NoSql) para Inventame Data Access Layer como Sistemas Gestores de Base de Datos.

En primer lugar, MySQL es un sistema de administración de bases de datos relacional (RDBMS) de código abierto, basado en lenguaje de consulta estructurado (SQL). Se trata de un programa capaz de almacenar una enorme cantidad de datos de gran variedad y de distribuirlos para cubrir las necesidades de cualquier tipo de organización, desde pequeños establecimientos comerciales a grandes empresas y organismos administrativos. Se ejecuta en prácticamente todas las plataformas, incluyendo Linux, UNIX y Windows. MySQL compite con sistemas RDBMS propietarios conocidos, como Oracle, SQL Server y DB2. (Gilfillan, 2011, p. 40).

Además, Apache Cassandra es un motor de bases de datos NoSQL, Open Source, basada en el modelo clave-valor. También soporta el concepto de columna y supercolumna, esencialmente para anidar los pares clave-valor que facilitan el modelado de estructuras de datos más complejas. De este modo, permite también la lectura y actualización de un registro sin recuperar el registro completo. Está escrita en Java con lo que funciona sobre cualquier sistema operativo, y utiliza Thrift para señalizar y comunicar los datos con programas externos. Esto permite usar Cassandra desde prácticamente

cualquier lenguaje de programación. Puede manejar varios terabytes de datos si lo necesita y puede, fácilmente, manejar millones de ficheros, incluso en un clúster pequeño (Big Data). Joyanes, 2016, p.120).

De acuerdo con, Martínez (2013, p. 4) las características principales de este manejador son las siguientes:

- *Esquema dinámico***.** El esquema que define la estructura de los datos puede cambiar en tiempo de ejecución.
- *No hay un único punto de fallo***.** Los datos se replican automáticamente a varios nodos. Perder un nodo no causa la baja del clúster.
- *Alta disponibilidad.* Los datos están disponibles la mayor parte del tiempo gracias a la redundancia que introduce la replicación de datos.
- *Particionamiento de los datos.* La topología de Cassandra es la de un anillo a través del cual se distribuyen los datos para minimizar cuellos de botella en el acceso a los mismos.
- *Escalabilidad horizontal.* Hasta un alto número de máquinas la capacidad de cómputo aumenta linealmente con el número de máquinas.
- *Capacidad* para manejar cientos de gigabytes de datos.

Por otra parte, los elementos que conforman una base de datos en Cassandra según Fernández (2015, p. 34) son:

- *Columna***.** Es la unidad más básica en el modelo de datos de Cassandra. Una columna es un triplete de un key (un nombre) un value (un valor) y un timestamp. Los valores son todos suministrados por el cliente. El tipo de dato del key y el value son matrices de bytes de Java, el tipo de dato del timestamp es un longprimitive.
- *SuperColumna***.** Es una columna cuyos valores son una o más columnas, que en este contexto se llamarán subcolumnas. Las subcolumnas están ordenadas, y el número de columnas que se puede definir es ilimitada.
- *Familia de Columnas.* Es más o menos análogo a una tabla en un modelo relacional. Se trata de un contenedor para una

colección ordenada de columnas. Cada familia de columna se almacena en un archivo separado.

- *Espacio de claves*. Es el contenedor para las Familias de Columnas. Es más o menos análogo a una base de datos en un modelo relacional, usado en Cassandra para separar aplicaciones.
- *Clúster.* Conjunto de máquinas que dan soporte a Cassandra y son vistas por los clientes como una única máquina.

Por otra parte, la interacción de los usuarios finales con Inventame Data Access Layer se realiza a través de un sitio web, el cual consiste en un conjunto de páginas web relacionadas entre sí. Entendiendo como página web, tanto el fichero que contiene el código HTML como todos los recursos que se emplean en la página (imágenes, sonidos, código JS). En un sitio web el conjunto de páginas web relacionadas pertenecen a un dominio web específico, que es frecuentemente conocido como World Wide Web (WWW), al cual se accede utilizando un Navegador Web, que permite utilizar el protocolo HTTP (por sus siglas en inglés, HyperText Transfer Protocol) para poder acceder a documentos de tipo HTML (Mora, 2002, p. 62).

La conjunción de los distintos sitios web permite conformar una extensa red de información que es conocida genéricamente como World Wide Web, pero cuentan con partes constitutivas respectivas a su estructura general, teniendo una parte física, que comprende al servidor donde está alojado, mientras que, por otro lado, se tiene un componente lógico, encontrándose esto último en lo que respecta a su organización, jerarquía y conexión que ofrecen entre los hiperenlaces.

Sin embargo, el sitio web requiere para su funcionamiento de un servidor web, éste es definido por Mora (2002, p. 49), como “un programa que está esperando permanentemente las solicitudes de conexión mediante el protocolo HTTP por parte de los clientes web”. Además, según (ob. cit.) la parte servidor de las aplicaciones web está formada por:

- Páginas estáticas (HTML) que siempre muestran el mismo contenido.
- Recursos adicionales (multimedia, documentos adicionales, etc.) que se pueden emplear dentro de las páginas o estar disponibles para ser descargados y ejecutados o visualizados por el cliente.
- Programas o Scripts que son ejecutados por el servidor web cuando el cliente solicita algunas páginas. La salida de estos scripts suelen ser una página HTML que se envía al navegador del cliente, así como también pudieran ser datos obtenidos de la base de datos en forma de documento XML o JSON.

Una vez comprendido este punto es importante destacar que, para Inventame Data Access Layer no sólo es posible interactuar a través de la interfaz web sino también por medio de la API que se le proporciona al usuario. Una Interfaz de Programación de Aplicaciones (API) es un conjunto de reglas (constantes, funciones y protocolos) que permiten programas aplicaciones. Una buena API facilita la tarea de desarrollar aplicaciones, ya que facilita todas las piezas y el programador sólo tiene que unirlas para lograr el fin que desea. De esta forma, sirve de interfaz entre programas diferentes de la misma manera en que la interfaz de usuario facilita la interacción humano-software (Mora, 2002, p. 11).

Las APIs pueden servir para comunicarse con el sistema operativo (WinAPI), con bases de datos (DBMS) o con protocolos de comunicaciones (Jabber/XMPP). En los últimos años, por supuesto, se han sumado múltiples redes sociales y otras plataformas online (ob. cit.).

Las API son valiosas, porque permiten hacer uso de funciones ya existentes en otro software, reutilizando así código que se sabe que está probado y que funciona correctamente. En el caso de herramientas propietarias, son un modo de hacer saber a los programadores de otras aplicaciones cómo incorporar una funcionalidad concreta sin por ello tener

que proporcionar información acerca de cómo se realiza internamente el proceso.

En vista de los beneficios que aporta el uso de API en cuanto a reutilización de lógica para el desarrollo de nuevos productos se hace uso de ella en Inventame Data Access Layer. Ésta se vale de entry point por la URL para que el usuario pueda realizar las búsquedas deseadas.

En concordancia, EndPoint es la manera en la que se publica la metadata del servicio web. Cada servicio tiene una dirección que define donde reside el servicio, un contrato que define lo que el servicio hace y un Binding que define como comunicarse con el servicio. La relación entre estos tres elementos son llamados EndPoint. Básicamente, es la dirección URL donde el servicio puede ser localizado y consumido (Cardador, 2015, p. 45).

Por su parte, el término Localizador de Recursos Uniformes (URL) hace referencia a una sucesión de caracteres mediante la que se asigna una dirección única a cada uno de los recursos que se encuentran en la WWW. Esta dirección será única y solamente válida para un recurso concreto y será la herramienta que utilice el navegador web para identificar, localizar y mostrar al usuario la información requerida. El URL se clasifica por su esquema, que generalmente indica el protocolo de red que se usa para recuperar a través de la red la información del recurso identificado (Corrales, 2015, p. 142).

Finalmente, sea la interacción por la interfaz web o por API la autenticación de Inventame Data Access Layes es basada en tokens. Para comprenderlo, de acuerdo con Andreu, Pellejero y Lesta (2006, p. 75) la autenticación basada en tokens es stateless (sin estado). Esto quiere decir que no guarda ningún tipo de información del usuario en el servidor ni en la sesión. Este tipo de autenticación envía un token en cada petición a través de las cabeceras HTTP en lugar de mantener la información de autenticación en sesiones o cookies. De esta forma, no se guarda ningún estado entre las

diferentes peticiones de nuestra aplicación y nuestros datos pueden ser consumidos desde diferentes tipos de clientes.

Para el uso de tokens se necesita de un sistema de codificación, tal es caso de JSON WEB TOKEN (JWT) utilizado en Inventame Data Access Layer, éste cifra en un objeto JSON los datos requeridos (en este caso los datos del usuario) para la autenticación y es útil para diseñar la comunicación entre dos sistemas de manera segura, para ello consta de tres partes: encabezado, carga útil, y firma (Dickey, 2014, p. 89)

JWT necesita para codificar y decodificar el token una clave secreta o semilla. Es importante que esta clave permanezca lo más oculta posible. Para Inventame Data Access Layer la semilla o clave secreta es una certificado rsa, éste se genera un algoritmo asimétrico (algoritmo rsa) cifrador de bloques, que utiliza una clave pública, la cual se distribuye (en forma autenticada preferentemente), y otra privada, la cual es guardada en secreto por su propietario (ob. cit.).

Por otra parte, para el almacenamiento de contraseñas es necesario valerse de una función hash, la cual viene siendo un método para generar claves o llaves que representen de manera unívoca a un documento o conjunto de datos, principalmente a su contenido. De manera que, si durante un transporte a través de la red algún tercero malintencionado modifica un archivo, se puede comprobar si el archivo final es exactamente igual al original comparando sus claves hash (Aguilera, 2010, p. 169). En otras palabras, el hash es un código que se obtiene tras aplicar un algoritmo especial a una cadena de texto. En Inventame Data Access Layer se utiliza el algoritmo SHA-512 para el encriptamiento de las contraseñas, añadiéndole al hash un salt, un número de dígitos aleatorios que se le agrega al hash ya sea al principio o al final, el cual es distinto para cada contraseña; lo que por supuesto aumenta la seguridad de Inventame Data Access Layer.

Bases Legales

A continuación se muestran documentos legales de Chile que explican la importancia de la firma electrónica y la protección de datos:

La ley 19.799 titulada el Documento Electrónico en el Derecho Civil Chileno explica cuál debe ser el tratamiento correcto de los documentos electrónicos, firma electrónica y servicios de certificación de dicha firma. Ésta constituye una base sobre la cual se debe construir toda una infraestructura tanto legal, consuetudinaria y técnica que permita el reemplazo definitivo del documento tradicional. (Fernández, 2004) En vista de que, Inventame Data Access Layer se vale de autenticación basada en token utilizando como semilla un certificado RSA es posible cumplir con el debido tratamiento de los datos, cumpliendo así con lo estipulado en esta ley. A continuación se muestra lo que afirma esta ley respecto a la firma electrónica:

Artículo 3. Los actos y contratos otorgados o celebrados por personas naturales o jurídicas, suscritos por medio de firma electrónica, serán válidos de la misma manera y producirán los mismos efectos que los celebrados por escrito y en soporte de papel. (ob. cit)

De manera que, es un asunto delicado y de importancia cumplir con la firma electrónica a nivel legal. Además, la ley 19628 sobre Protección de Datos de carácter personal especifica los principios de acceso y de finalidad que es necesario tomar en cuenta. En cuanto al principio de acceso éste consiste en otorgar derecho de acceso a los titulares de datos, respecto de la información que le concierne, para cerciorarse, en particular, de su exactitud y de la licitud del tratamiento. Por su parte, el principio de finalidad establece que el tratamiento de datos personales debe ser informado al momento de la recogida de estos datos, finalidad que debe perdurar durante todas las etapas posteriores del tratamiento, salvo que exista consentimiento del titular de los datos. (Jervis, 2006, p.66)

CAPÍTULO III

MARCO METODOLÓGICO

En esta sección se plantea el modo con el cual se procedió a alcanzar los objetivos, mostrando el conjunto de pasos o etapas generales que sirvieron de guía en este proceso. Para ello, se indica la metodología de la investigación y la metodología de desarrollo, explicadas a continuación.

Metodología de la Investigación

Según Arias (2006, p. 22), sostiene que: “la investigación científica es un proceso metódico y sistemático dirigido a la solución de problemas o preguntas científicas, mediante la producción de nuevos conocimientos, los cuales constituyen la solución o respuesta a tales interrogantes”. Por lo tanto, para generar conclusiones y obtener resultados, fue necesario atacar dichas interrogantes tomando en cuenta una serie de procedimientos y métodos, los cuales se presentan a continuación:

Tipo de investigación

De acuerdo a, Hurtado (2000, p. 130) la investigación de tipo proyectiva, consiste en: “Todas aquellas investigaciones que conducen a inventos, programas, diseños o a creaciones dirigida a cubrir una determinada necesidad y basada en conocimientos anteriores”. Tomando en cuenta el perfil de la investigación, se determinó que es de tipo proyectiva, puesto que se desarrolló la capa de acceso a datos a través de una interfaz web para para el manejo de datos de los servicios brindados por la empresa Inventame Group LLC.

Diseño de investigación

En concordancia con Arias (2006, p. 31) el diseño de campo consiste en: “la recolección de datos directamente de los sujetos investigados, sin manipular o controlar variable alguna (datos primarios)”. A partir de esta definición, se pudo constatar que el diseño de la investigación es de campo, puesto que se realizaron visitas frecuentes para obtener los datos directamente del lugar de los hechos, con el fin de adquirir la información real y precisa necesaria para llevar a cabo el desarrollo de la investigación.

Sin embargo, no sólo se obtuvieron datos primarios sino también secundarios; pues fue necesario realizar una revisión bibliográfica de diversas fuentes en el área de investigación. En este orden de ideas, se toma en cuenta el diseño de investigación documental que (ob. cit., p.27) lo define como: “un proceso basado en la búsqueda, recuperación, análisis, critica e interpretación de datos secundarios, es decir, los obtenidos y registrados por otros investigadores en fuentes documentales: impresas, audiovisuales o electrónicas”. En vista de ello, se consideró que el diseño de la investigación también es de tipo documental, debido a que se consultaron fuentes de información impresas, electrónicas, trabajos de investigación,

trabajos de grado, libros, documentos técnicos y leyes, con el fin de adquirir las bases necesarias para el desarrollo de la investigación.

Instrumentos de Recolección de Datos

La recolección de datos según Hurtado (2000, p.153) “comprende procedimientos y actividades que le permiten al investigador obtener información necesaria durante la investigación”. Para obtener la información requerida y darle respuesta a los objetivos propuestos se utilizaron la observación directa, la entrevista no estructurada y la revisión documental.

La observación directa según (ob. cit., p. 459) consiste en “visualizar o captar mediante la vista, en forma sistemática, cualquier hecho, fenómeno o situación que se produzca en la naturaleza o en la sociedad, en función de unos objetivos de investigación preestablecidos”. En esta investigación se utilizó la observación directa como instrumento de recolección de datos debido a que se participó directamente en los procesos pertinentes que se llevan a cabo en la empresa Inventame Group, en la cual se realizó la investigación.

Por otra parte, la entrevista no estructurada según (ob. cit., p. 469) se define como “técnica en la que no se dispone de preguntas elaboradas previamente, pero seguida por unos objetivos preestablecidos que permitan definir el tema de la entrevista”. Se toma en cuenta éste instrumento porque se realizaron entrevistas sin formatos específicos con características de conversación, que permitieron que la entrevista fuera más fluida y menos rígida. De esta manera, las entrevistas se dieron de manera más clara y simple y se pudo adquirir la información necesaria para la investigación.

Finalmente, la revisión documental según (ob. cit., p.427): “es aquella que se basa en la obtención y análisis de datos provenientes de materiales impresos u otros tipos de documentos”. Basado en la definición anterior, fue

tomada en cuenta la revisión documental como instrumento de recolección de datos, puesto que esta investigación se apoya en documentos escritos, trabajos de investigación y de grado, entre otros, con el fin de recolectar, seleccionar, analizar y presentar resultados coherentes para la investigación.

Metodología de Desarrollo

La metodología de desarrollo utilizada fue SCRUM. Schwaber y Sutherland (2016) la definen como: “Un proceso de desarrollo de software iterativo y creciente utilizado comúnmente en entornos basados en el desarrollo ágil de software”.

Schwaber y Sutherland (2016), explican que esta metodología consta de un Dueño de Producto (Product Owner), el Equipo de Desarrollo (Development Team), y un Scrum Master (Facilitador). Los Equipos Scrum son auto-organizados y multifuncionales. Los equipos auto-organizados eligen la mejor forma de llevar a cabo su trabajo, en lugar de ser dirigidos por otros externos al equipo. Los equipos multifuncionales tienen todas las competencias necesarias para llevar a cabo el trabajo sin depender de otros que no son parte del equipo. El modelo de equipo en Scrum está diseñado para optimizar la flexibilidad, la creatividad y la productividad.

El dueño de producto (Product Owner) es el responsable de maximizar el valor del producto y del trabajo del Equipo de Desarrollo. Es el responsable de gestionar la Pila de Producto (ProductBacklog). Además, el Scrum Master es el responsable de asegurar que Scrum es entendido y llevado a cabo, es decir, debe asegurarse de que se trabaja ajustándose a la teoría, prácticas y reglas.

Se utilizan eventos en la forma de bloques de tiempo (timeboxes), de modo que todos tienen una duración máxima. Esto asegura que se emplee una cantidad apropiada de tiempo en la planificación, de forma que no se admita desperdicio en este proceso de planificación. Para llevar un control de las actividades a realizar se cuenta con Scrum Diario que es una reunión restringida a un bloque de tiempo de quince (15) minutos, para que el Equipo de Desarrollo sincronice sus actividades y cree un plan para las siguientes veinticuatro (24) horas. Esto se lleva a cabo inspeccionando el trabajo avanzado desde el último Scrum Diario y haciendo una predicción acerca del trabajo que podría ser completado antes del siguiente.

Por otra parte, el desarrollo se realiza de forma iterativa e incremental. Cada iteración, denominada Sprint, tiene una duración preestablecida de entre dos (2) y cuatro (4) semanas, obteniendo como resultado una versión del software con nuevas prestaciones listas para ser usadas. En cada nuevo Sprint, se va ajustando la funcionalidad ya construida y se añaden nuevas prestaciones priorizando siempre aquellas que aporten mayor valor de negocio. (ob. cit.)

Los tres (3) artefactos que comprende SCRUM de forma general de acuerdo a Schwaber y Sutherland (2016) se muestran a continuación:

- *Planificación del Backlog.* Se define un documento en el que se reflejarán los requisitos del sistema por prioridades. Además, se define la planificación del Sprint 0, en el que se decidirá cuáles serán los objetivos y el trabajo a realizar para esa iteración. También, se obtiene un Sprint Backlog, que es la lista de tareas y el objetivo más importante del Sprint.
- *Seguimiento del Sprint*. En esta fase se hacen reuniones diarias en las que las tres (3) preguntas principales para evaluar el avance de las tareas serán: ¿Qué trabajo se realizó desde la reunión anterior? ¿Qué trabajo se hará hasta una nueva reunión? ¿Qué inconvenientes han surgido y qué hay que solucionar para poder continuar?

- *Revisión del Sprint*. Cuando se finaliza el Sprint se realizará una revisión del incremento que se ha generado. Se presentarán los resultados finales y una demo o versión, esto ayudará a mejorar el feedback con el cliente.

A continuación se muestran los principales eventos involucrados en esta metodología los cuales son definidos por Schwaber y Sutherland (2016) como:

- *ProductBacklog:* Conjunto de requisitos denominados historias descritos en un lenguaje no técnico y priorizados por valor de negocio, o lo que es lo mismo, por retorno de inversión considerando su beneficio y coste. Los requisitos y prioridades se revisan y ajustan durante el curso del proyecto a intervalos regulares.
- *Sprint Planning:* Reunión durante la cual el Product Owner presenta las historias del backlog por orden de prioridad. El equipo determina la cantidad de historias que puede comprometerse a completar en ese sprint, para en una segunda parte de la reunión, decidir y organizar cómo lo va a conseguir.
- *Sprint:* Iteración de duración prefijada durante la cual el equipo trabaja para convertir las historias del ProductBacklog a las que se ha comprometido, en una nueva versión del software totalmente operativo.
- *Sprint Backlog:* Lista de las tareas necesarias para llevar a cabo las historias del sprint.
- *Daily sprint meeting:* Reunión diaria de cómo máximo 15 min. En la que el equipo se sincroniza para trabajar de forma coordinada. Cada miembro comenta que hizo el día anterior, que hará hoy y si hay impedimentos.
- *Demo y retrospectiva:* Reunión que se realiza al final del sprint y en la que el equipo presenta las historias conseguidas mediante una demostración del producto. Posteriormente, en la retrospectiva, el equipo analiza qué se hizo bien, qué procesos serían mejorables y discute acerca de cómo perfeccionarlos.

Las tres (3) fases en las que se divide SCRUM de forma general de acuerdo a Schwaber y Sutherland (2011, p.68) se muestran a continuación:

- Pre-juego

- Planeamiento: Tiene como propósito establecer la visión, definir las expectativas y asegurar la financiación del proyecto, tiene como principales actividades:
  - Escribir la visión.
  - Escribir el presupuesto.
  - Escribir el registro de acumulación o retraso (backlog) del producto inicial y los ítems estimados así como la arquitectura de alto nivel, el diseño exploratorio y los prototipos.

  Los registros de acumulación pueden incluir presentaciones del software, funciones, corrección de errores, mejoras requeridas y actualizaciones de tecnología. Hay un registro para el total del producto y otro específico para cada corrida de 30 días (sprint), todo a un alto nivel de abstracción.
- Montaje (Staging): Tiene como propósito identificar más requerimientos y priorizar las tareas para la primera iteración, las actividades son:
  - Planificación
  - Diseño exploratorio
  - Prototipos

- Juego o Desarrollo: Tiene como propósito implementar un sistema listo para entrega en una serie de iteraciones de 30 días llamadas “corridas” (sprints), las actividades son:
  - Un encuentro de planeamiento de corridas en cada iteración.
  - La definición del registro de acumulación de corridas
  - Los estimados y los encuentros diarios de Scrum. En los encuentro diarios todos tienen que ser puntuales y se debe responder a las siguientes preguntas
    - ¿Qué hice desde la reunión anterior?
    - ¿Qué voy a hacer hasta la siguiente reunión?
    - ¿Qué impide que avance en las tareas?
- Post juego (liberación): El propósito es el despliegue operacional de la iteración, las actividades a desarrollar son:
  - Documentación
  - Entrenamiento
  - Mercadeo y venta

CAPÍTULO IV.

RESULTADOS

A continuación se presentan los resultados de la investigación de acuerdo a las fases descritas en el capítulo anterior correspondientes a la metodología de desarrollo. A su vez se identifican los elementos involucrados en la misma:

Actores

Es necesario identificar los actores que intervienen en el proceso de desarrollo, los cuáles de forma general son: El dueño del producto, el maestro de desarrollo o facilitador y el equipo de desarrollo. Para la investigación se tienen los siguientes actores:

- *Dueño del Producto*. Lic. Rodrigo Navarro, quien es el director ejecutivo de la empresa Inventame Group LLC.
- *Maestro de Desarrollo*. Lic. Rodrigo Navarro, quien ha coordinado el proceso de desarrollo, ha asesorado en materia informática, dando las

instrucciones correspondientes, y se ha encargado de liberar los obstáculos para conseguir cumplir los objetivos planteados.

- *Equipo de Desarrollo*. El equipo para el desarrollo del producto está integrado en este caso por 2 desarrolladores del equipo de Inventame Group LLC y adicionalmente la Br. Yulialci Adrián, estudiante de la carrera Licenciatura en Informática de la Universidad de Oriente Núcleo Nueva Esparta (UDONE), quien como autora de esta investigación se incorpora a la empresa Inventame Group LLC.

Fases del Desarrollo

El desarrollo del sistema web se extiende como un ciclo desde la fase de determinación de requerimientos hasta la fase de codificación del producto en la organización. La diferencia que marca la metodología Scrum es que minimiza la cantidad de diagramación abstracta de un sistema y se enfoca en el desarrollo programado, la constante interacción con el dueño del producto y los clientes y la supervisión y guía constante y programada del maestro de desarrollo.

De modo que el desarrollo de esta investigación está guiado por los objetivos de la investigación y la práctica de tres elementos importantes: ProductBacklog, Sprint Backlog y DailyScrum Meeting.

Fase 1. Pre-Juego

Durante esta etapa se realizó la recolección y análisis de la información que se juzgaba necesaria para el desarrollo del producto. Las técnicas aplicadas a los procesos desarrollados en la empresa fueron las entrevistas no estructuradas y la observación directa.

La entrevista no estructurada fue de utilidad durante todo el proceso de codificación del software, ya que fue posible recurrir de forma no planificada y en distintas oportunidades a los actores vinculados con el desarrollo. Las personas responsables del suministro de la información fueron: Lcdo. Rodrigo Navarro e Ing. Eyamir Ugueto. Además, en lo que respecta a la observación, se tuvo la oportunidad de notar las metodologías utilizadas en la codificación permitiendo profundizar en lo que se hace y cómo se hace.

Con base a la información obtenida luego de la aplicación de las técnicas de recolección de datos se elaboró el *ProductBacklog* documento resultante de la Fase 1 que resume los requisitos o necesidades de negocio desde el punto de vista del cliente, éste demarca lo siguiente:

1. Definir modelo relacional.
2. Definir modelo no relacional.
3. Definir clase para conexión con Mysql.
4. Definir clase para conexión con Cassandra.
5. Crear base de datos Mysql.
6. Definir base de datos Cassandra.
7. Generar datos de prueba para Mysql.
8. Generar datos de prueba para Cassandra.
9. Realizar pruebas de conexión con Mysql.
10. Realizar pruebas de conexión con Cassandra.
11. Definir servicio base para el core.
12. Definir clases principales del core API.
13. Definir elementos y métodos de las clases principales.
14. Crear clase User.
15. Prueba de clase User.
16. Crear clase Namespace.
17. Prueba de clase Namespace.

18. Crear clase schema.
19. Prueba de clase schema.
20. Crear clase columna.
21. Prueba de clase columna.
22. Crear clase datos.
23. Prueba de clase datos.
24. Definir parser.
25. Añadir definición de clase para JSON.
26. Realizar pruebas JSON.
27. Añadir definición de clase para XML.
28. Realizar pruebas XML.
29. Añadir definición de clase para CSV.
30. Realizar pruebas CSV.
31. Añadir definición de clase para EXCEL.
32. Realizar pruebas EXCEL.
33. Generar documentación de REST API.
34. Definir Frontend.
35. Definir Backend.

Después de recopilar la información y determinar las necesidades del cliente mediante el ProductBacklog se debe producir el Sprint Backlog, el cual es un documento detallado donde se describe cómo el equipo va a implementar los requisitos durante el siguiente sprint. Las tareas se dividen en horas, pero ninguna tarea con una duración superior a 16 horas. Si una tarea es mayor de 16 horas, deberá ser dividida en otras menores, se sitúa en el área de especificación de requisitos del software necesarios para dar respuestas a las funcionalidades esperadas por el cliente y que fueron determinadas en la lista anterior.

En esta etapa la intervención del Lcdo. Rodrigo Navarro como Scrum Master fue crucial, ya que, demarcó las directrices que en materia de software deberían ser cumplidas para satisfacer los requerimientos del cliente y los estándares de desarrollo del departamento.

De esta manera, el Sprint Backlog resume especificaciones de software que deben ser cumplidas, estas son (ver Cuadro 1).

Cuadro 1.

Lista de Actividades (Sprint Backlog)

N°	Requerimiento	Herramientas de Desarrollo	Horas
1	**Definir modelos y generar datos de prueba**		**12**
1.1	Establecer modelo relacional	Star UML	1
1.2	Establecer modelo no relacional	Power Point	3
1.3	Generar datos de prueba Mysql	Google Drive – Excel	4
1.4	Generar datos de prueba Cassandra	Google Drive – Excel	4
2	**Definir clases para conexión a bases de datos relacional y no relacional**		**12**

2.1	Definir clase para conexión a base de datos en Mysql	Node JS, mysql	1
2.2	Realizar pruebas de conexión con Mysql	Node JS, mysql	1
2.3	Definir clase para conexión a base de datos en Cassandra	Node JS, cassandra-driver	5
2.4	Realizar pruebas de conexión con Cassandra	Node JS, cassandra-driver	5
3	**Establecer comportamiento de idal-core**		**16**
3.1	Definir servicio base para idal-core	Node JS, Express	8
3.2	Definir clases principales para idal-core	Node JS	8

(Cont.) Cuadro 1

4	**Definir elementos y métodos de clases principales**		**60**
4.1	Crear clase Usuario	Node JS, node-uuid, nodemailer, redis	16
4.2	Realizar pruebas para clase Usuario	Node JS, Postman, frisby	8
4.3	Crear clase Namespace	Node JS, node-uuid	8
4.4	Realizar pruebas para clase Namespace	Node JS, Postman, frisby	4
4.5	Crear clase esquema	Node JS, node-uuid	8
4.6	Realizar pruebas para clase esquema	Node JS, Postman, frisby	4
4.7	Crear clase columna	Node JS, node-uuid	8

4.8	Realizar pruebas para clase columna	Node JS, Postman, frisby	4
5	**Definir elementos y métodos para manejo de datos**		**20**
5.1	Crear clase datos	Node JS, node-uuid	10
5.2	Realizar pruebas para clase datos	Node JS, Postman, frisby	10
6	**Definir parser**		**75**
6.1	Añadir definición de clase para JSON	Node JS, lodash	10
6.2	Realizar pruebas con JSON	Node JS, Postman	10
6.3	Añadir definición de clase para XML	Node JS, xml2js	15
6.4	Realizar pruebas con XML	Node JS, Postman	10
6.5	Añadir definición de clase para CSV	Node JS, csvtojson	15
6.6	Realizar pruebas con CSV	Node JS, Postman	10

(Cont.) Cuadro 1

6.7	Añadir definición de clase para EXCEL	Node JS, node-excel-to-json	20
6.8	Realizar pruebas con EXCEL	Node JS, Postman	10
7	**Generar documentación de REST API**	Swagger	**8**
8	**Diseñar interfaz idal-web-server**		**90**
8.1	Definir frontend	React JS (react-router, react-redux, webpack, whatwg-fetch)	45
8.3	Definir backend	React JS (react-router, react-redux, webpack, whatwg-fetch)	45

Por su parte, se procedió a estimar los costes de desarrollo que comprenden comercialización, entrenamiento y lanzamiento (ver Cuadro 2)

Cuadro 2.

Costes estimados de Desarrollo

Comercialización	Entrenamiento	Lanzamiento
100$ por Mes por 10 Gb de transferencia	120.000 Bs	350.000 Bs

Fase 2. Juego

En este proceso fue vital el uso de los DailyScrum Meetings, las cuales se caracterizan por ser reuniones diarias y continuas de muy corta duración entre los miembros del equipo de desarrollo. Éstas fueron realizadas a través de chats y videollamadas ya que el Lcdo. Rodrigo Navarro (Scrum Master) se encuentra en el extranjero. El objetivo de las mismas fue revisar de forma permanente el desarrollo del software para no perder de perspectiva las necesidades del cliente y los requerimientos. Estas reuniones también permitieron reajustar el plan inicial, reacomodar las necesidades y requerimientos, evolucionando así mediante un diseño incremental y evolutivo sin degradar la arquitectura pensada inicialmente. El cumplimiento de las prácticas habituales del desarrollo ágil debe ser el norte de actuación durante este proceso. Estas prácticas se resumen en: trabajar en iteraciones, desarrollo incremental, desarrollo evolutivo, auto-organización y colaboración.

Con base a lo expuesto, las iteraciones para el desarrollo estuvieron compuestas por un conjunto de reuniones realizadas para llevar un seguimiento del trabajo del equipo. De manera general se realizaron ocho (8) iteraciones, que abarcaron un período de seis (6) meses.

En la *Primera Iteración o Primer Sprint*, se buscó principalmente recopilar toda la información posible para crear la estructura de espacio de claves en Cassandra y estructura de esquemas en MySql así elaborar un modelo de bases de datos íntegro, para ello se realizaron varias reuniones y una vez obtenida una idea inicial sólida se contrató el servicio de dos (2) servidores para empezar la instalación de los programas necesarios e implementar la plataforma.

Durante estas reuniones quedaron definidas las siguientes pautas:

- El usuario puede tener n namespaces.
- El usuario tiene una apikey o llave de acceso, la cual puede bloquear en caso de verse comprometida y generar una nueva.
- Un namespace tiene n esquemas.
- Un esquema tiene n columnas.

Por otra parte, para el almacenamiento de datos en Cassandra se estableció que los datos se almacenarían en una estructura llave-valor, donde la llave es el identificador de la columna almacenada en Mysql. Esta estructura se mapea para obtener una familia de columnas, la cual tiene como llave el identificador de la tabla en Mysql y el identificador del registro. Para llegar a esto, se realizó un modelo conceptual (diagrama de clases) basado en UML correspondiente a la base de datos en MySql y un modelo o estructura de espacio de claves para la base de datos en Cassandra (Ver Gráficos 1 y 2). Cabe destacar que, el modelo conceptual realizado no forma parte de SCRUM, por lo que no es necesario realizarlo. Éste se diseñó a petición del asesor. Apegándonos a lo ágil de la metodología, no hay mejor forma de describir lo que se hizo sino a través del propio código, en cuyo caso, se presenta en el Gráfico 3 el diagrama obtenido en el manejador de base de datos MySql, utilizando phpMyAdmin el cual sí se corresponde más con la filosofía de trabajo que abandera la metodología utilizada.

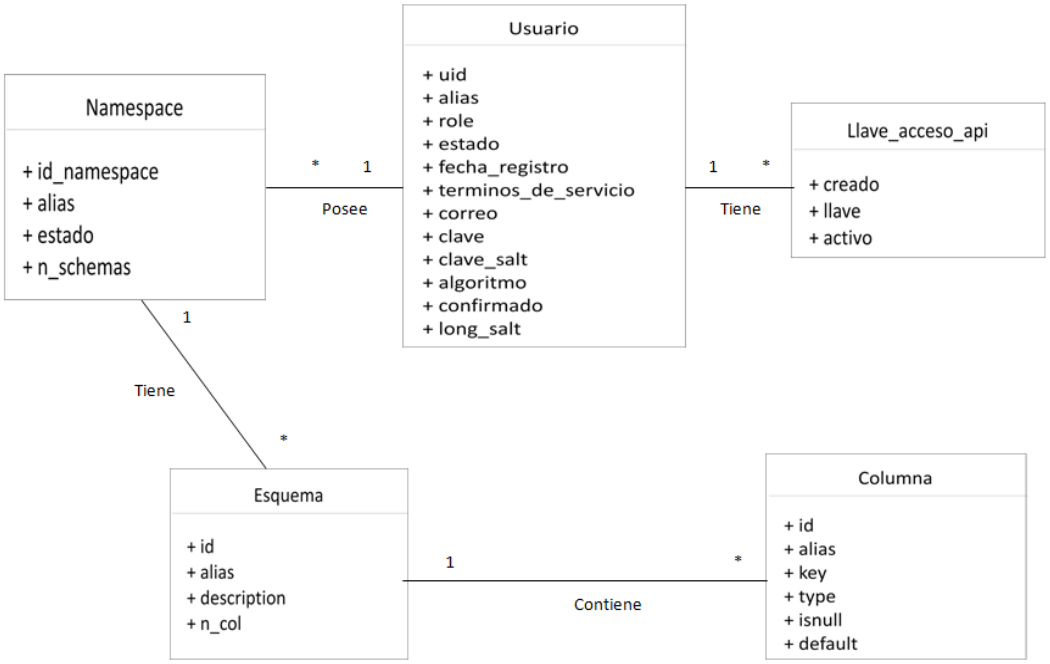


***Gráfico 1.*Diagrama de clases base de datos Mysql.**

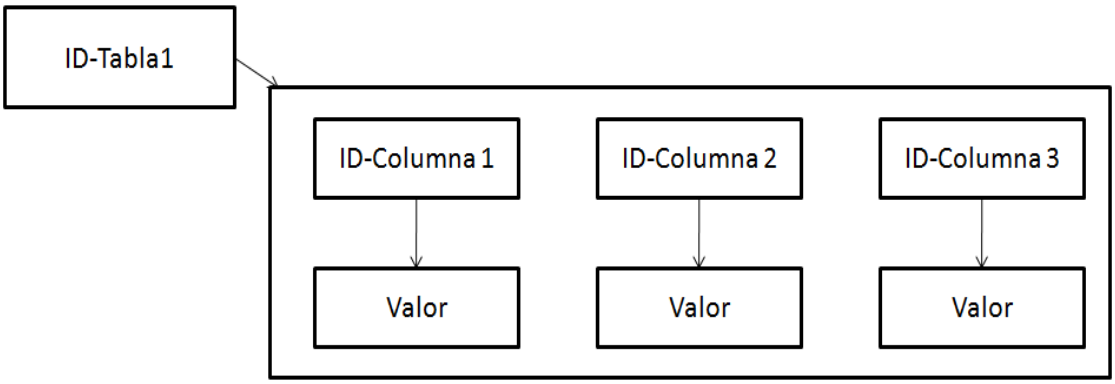


***Gráfico 2.*Estructura de almacenamiento de base de datos en Cassandra.**

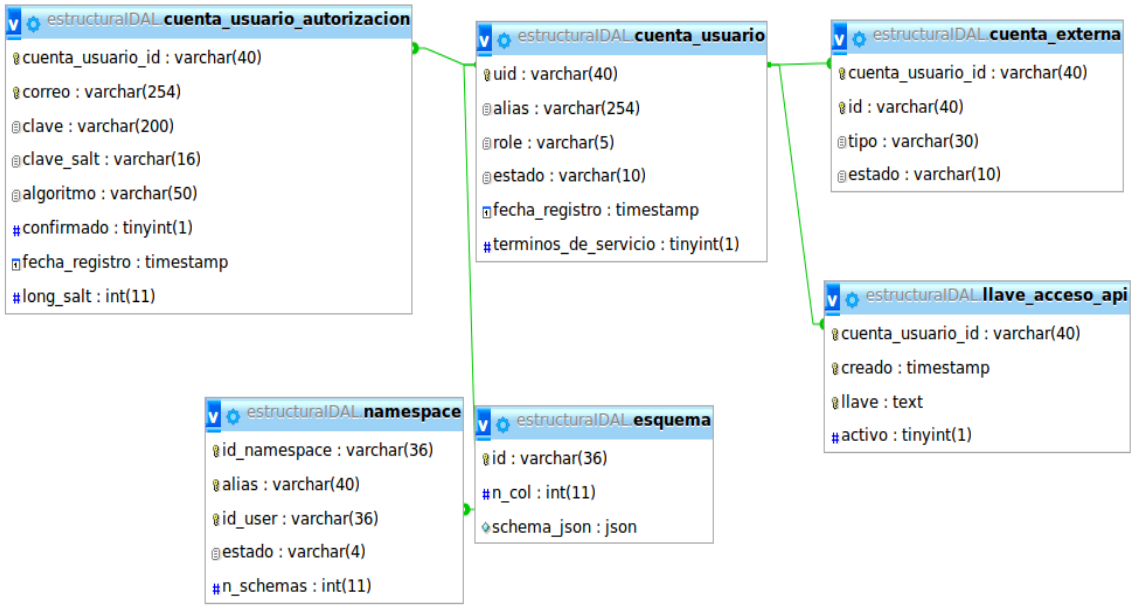


Gráfico 3. **Diagrama de base de datos Mysql generado por el manejador.**

Una vez implementados estos modelos, se procedió a generar los datos de prueba (Ver Gráficos 4 a 6).Para los mismos el control fue llevado en Excel, utilizando Google Drive, lo que permitió establecer con el Scrum Master estos datos. En Mysql se generaron los siguientes datos:

- Tres (3) usuarios sin namespaces.
- Diez (10) usuarios con un namespace.
- Diez (10) usuarios con n namespaces.
- Tres (3) usuarios sin registrar.
- En total treinta (30) namespaces.
- Tres (3) namespaces con un esquema.
- En total tres (3) esquemas.
- Un (1) esquema con tres (3) columnas.

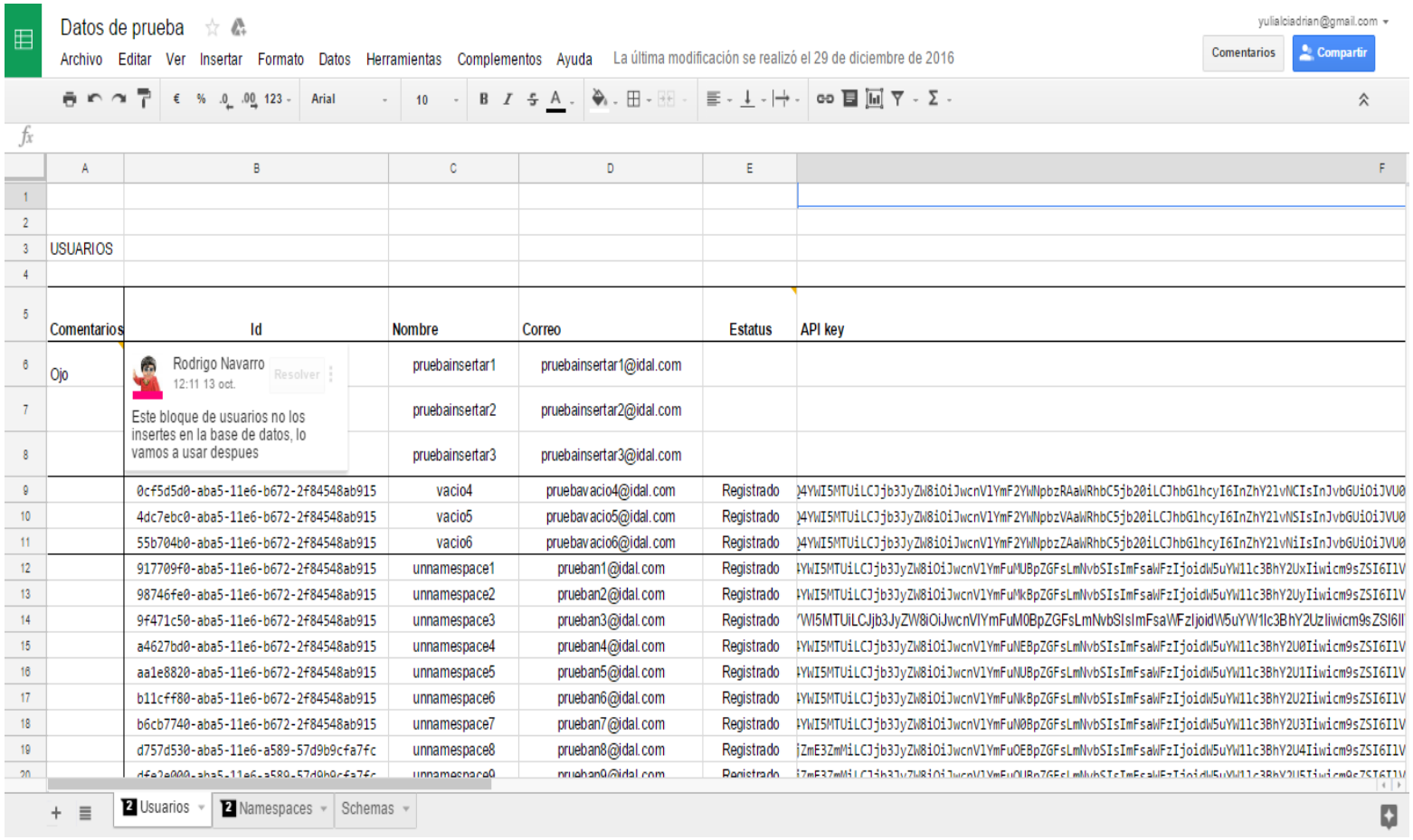


Gráfico 4. **Datos de Prueba correspondiente a los Usuarios**

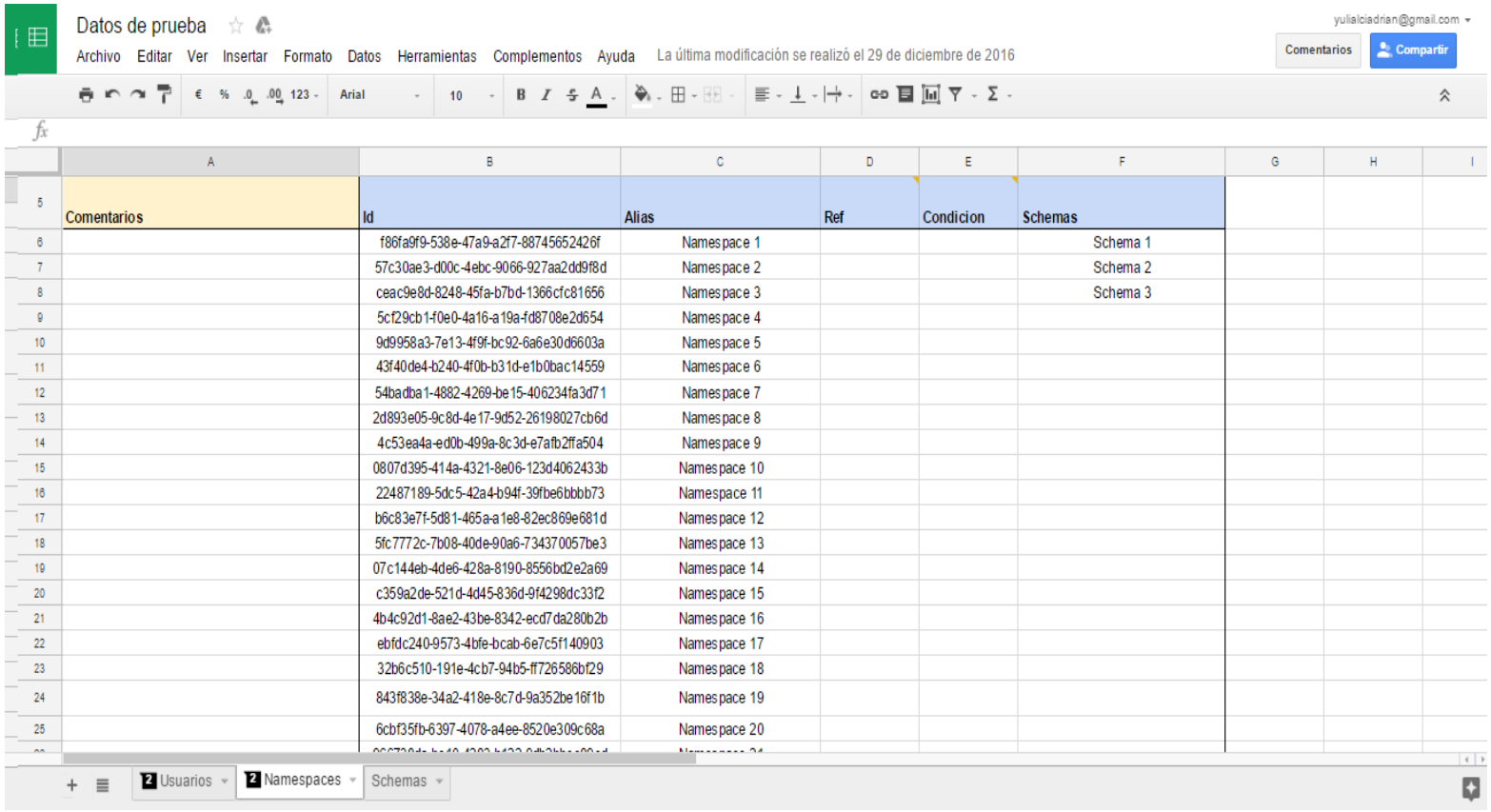


Gráfico 5. **Datos de Prueba correspondiente a los Namespaces**

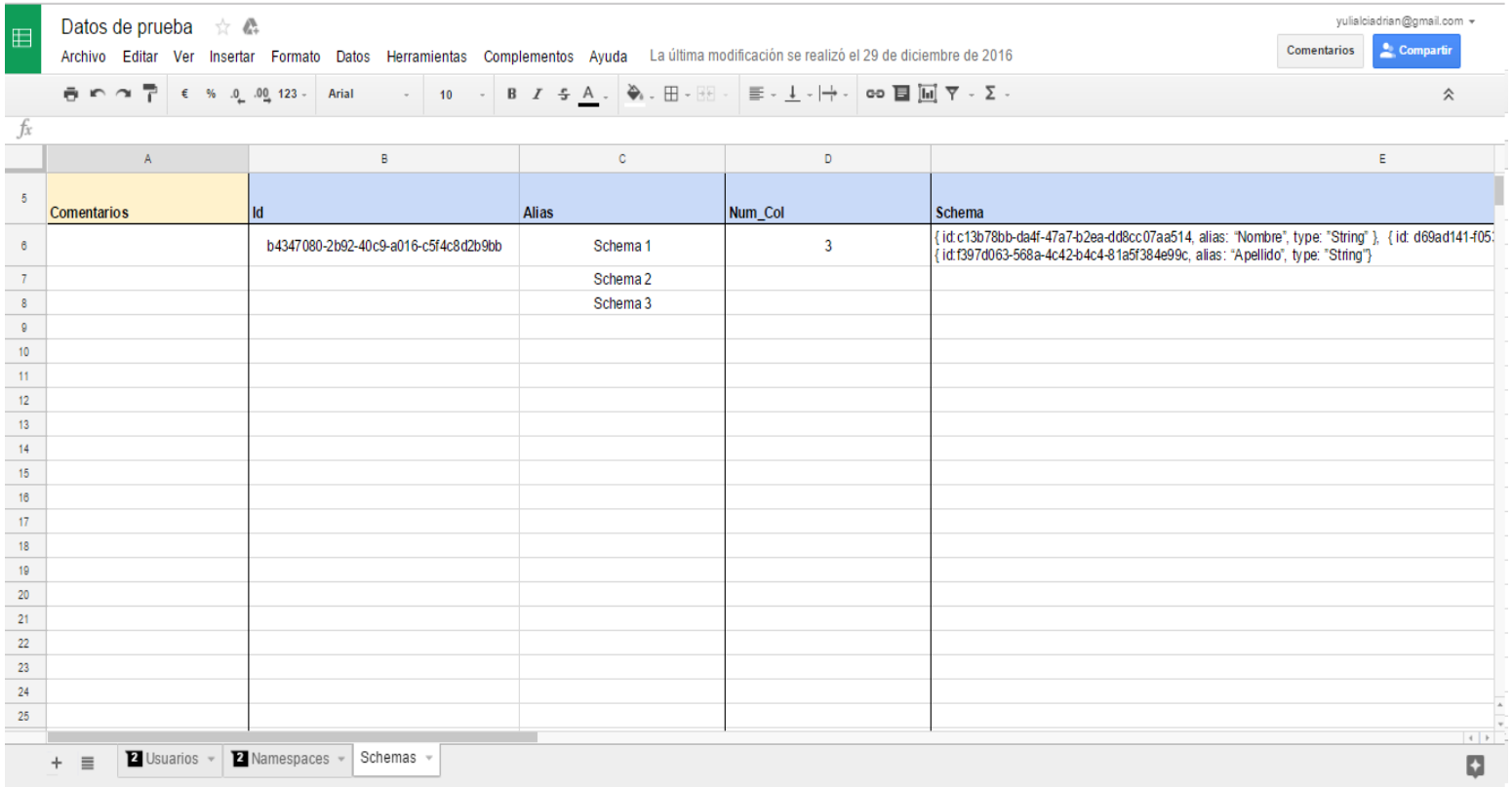


Gráfico 6. **Datos de Prueba correspondiente a los Esquemas.**

Para Cassandra se crearon datos correspondientes al esquema de tres (3) columnas (Ver Gráfico 7). En éste se puede observar la escalabilidad horizontal de la base de datos, pues las columnas (correspondientes a un registro en base de datos relacional) se almacenan en una estructura map, donde cada columna contiene en sí todos los campos asociados.

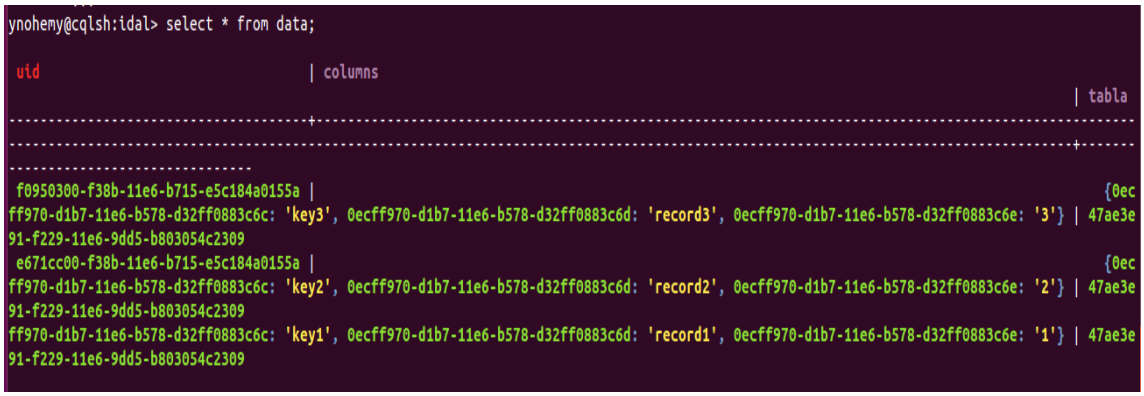


***Gráfico 7*. Datos de Prueba en Cassandra.**

Para la Segunda Iteración el objetivo principal fue llevar a cabo una investigación para determinar cómo realizar la conexión a las bases de datos. En vista de que existen diversas librerías era necesario decidir las apropiadas para Inventame Data Access Layer. Como resultado de dicha investigación las librerías utilizadas para la conexión desde Nodejs serían Mysql y cassandra-driver. Además de la conexión, era necesario verificar cómo realizar las diversas consultas para el manejo de los datos. Esto fue principalmente importante para la base de datos en Cassandra, ya que la manipulación de los datos es diferente a lo acostumbrado. Luego, se realizaron pruebas de conexión y se realizaron pruebas con las diversas consultas. Particularmente en Cassandra se analizó como realizar consultas por llave, valor, columna e identificador de familia de columnas.

Una vez finalizado este proceso, a través de reuniones con el Scrum Master se pasó a definir la organización o estructura de Inventame Data Access Layer, como resultado se vio la necesidad de separar los componentes necesarios para esta capa de acceso a datos:

- *IDAL-CORE:* Correspondiente al núcleo de la plataforma. Éste consiste en un API única que da soporte tanto a las acciones del servidor web como a

los desarrolladores para utilizarlo en otros productos que la empresa comercialice.

- *IDAL-WEB-SERVER:* Éste consiste en la interfaz web que permitirá a los clientes manipular los datos.
- *IDAL-LIBRARIES:* Se corresponde con las librerías que se desarrollarán a futuro para la conexión con IDAL-CORE. Éstas exceden del ámbito del trabajo de investigación.

El control de estos repositorios se realiza a través de Git, lo que permite que el Scrum Master verifique la organización del código, realice correcciones y las pruebas necesarias a distancia (ver Gráfico 8).

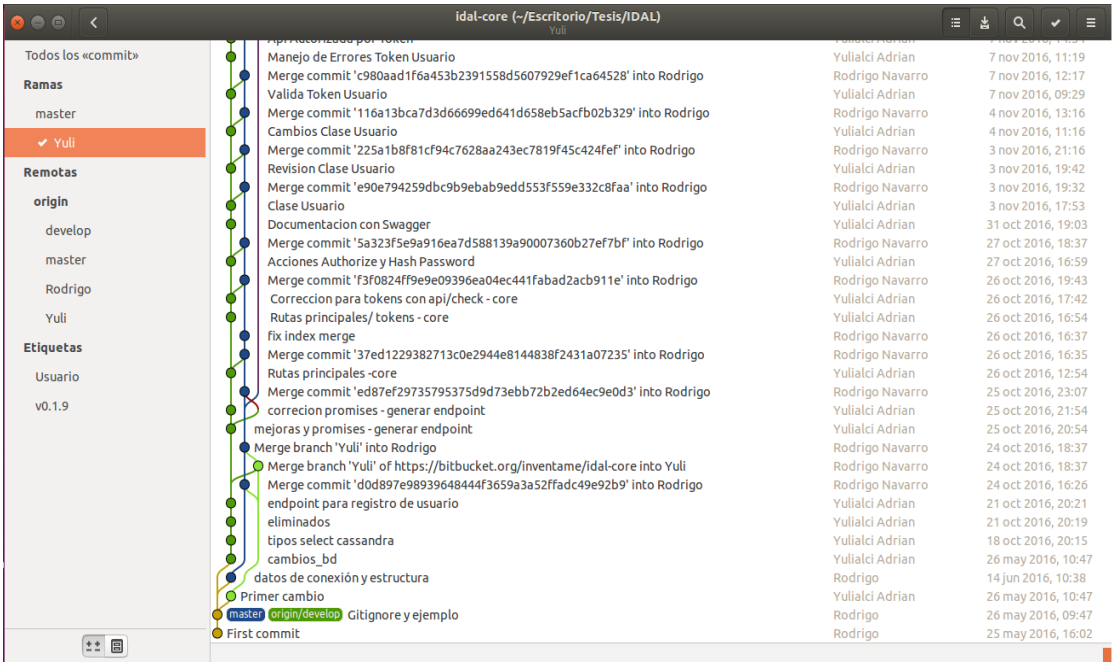


Gráfico 8. **Repositorio IDAL-CORE en Gitg.**

En la *Tercera Iteración* se empieza a trabajar con el primer repositorio (IDAL-CORE), iniciando por definir su comportamiento. A través de chats se estableció la estructura del mismo, obteniendo que éste se encuentre comprendido por:

- Classes. Clases necesarias para el funcionamiento de la plataforma.

- Routes. Rutas (entrypoint) correspondientes a la API para la manipulación de los datos.
- Templates. Para plantillas en caso de ser necesario.
- Utiles. En éste directorio se encuentran las conexiones a las bases de datos y se tienen otros archivos js útiles para la API.

  Una vez definida la estructura del core, se procedió a establecer el servicio base para el mismo. Como resultado se obtuvo lo siguiente.
- Cada usuario tiene un rol que indica su nivel de acceso, puede ser USER, ADMIN o SUPER.
- Para el control de roles en la rutas, se utilizó el manejo de ruta wildcard y middleware. De manera que un SUPER puede acceder a todas las rutas. El ADMIN puede ingresar a sus rutas y las de USER. Mientras que, el USER sólo a sus rutas.
- Para controlar el flujo o la autorización de EndPoint se utilizan tokens, estos pueden ser de dos tipos:
    - apikey: Token permanente que no caduca.
    - tmpkey: Token temporal, utilizado por el web-server
- Los tokens contienen los siguientes campos.
    - uid: Id de usuario.
    - correo: para mostrar si es necesario sin conexión a la base de datos.
    - alias: igual que correo.
    - role: Rol del usuario
    - type: Tipo de token (api - tmp)
    - apiVersion: Versión de api para la que fue generado el token.
    - generated: Fecha en la que se generó el token.
    - expires: Tiempo de caducidad en milisegundos en caso de tokens temporales. Para apikey es igual a -1.

    - invalidBefore: Rango máximo para que en caso de token expires generar un nuevo token, permite alargar la sesión para el usuario. Para apikey es igual a -1.
- Para la generación de los tokens se estableció el uso de la librería jsonwebtoken, y para la semilla se utilizó un certificado rsa, lo que permite brindar mayor seguridad a la plataforma.
- Se utilizan versiones para definir cambios en la API sin afectar las anteriores. Para el trabajo de investigación es la versión uno (1)
- La ruta raíz es /api. Por lo tanto, tomando en cuenta la versión la ruta base es api/v1.
- Esta ruta tiene en principio 4 (cuatro) EndPoint
    - api/v_i/*: Para usuarios registrados indicando la versión de la API.
    - api/register: Para el registro de usuarios.
    - api/authorize: Para generar un tmpkey.
    - api/check: En caso de caducar tmpkey, se genera uno nuevo.
- Para enviar datos a las rutas se realizan en formato JSON.
- Para el manejo de respuestas de las diversas rutas se definió un JSON con los siguientes campos:
    - status: Indica si la transacción fue exitosa o si hubo un error, con los valores ok y error respectivamente
    - code: Correspondiente a código http que indica el estado de la transacción.
    - data o message: Donde se envían los datos resultantes o el mensaje de error correspondiente.

En el siguiente gráfico se muestra un resumen establecido con el Scrum Master del comportamiento por parte del idal-core e idal-web-server

respecto al acceso por parte de los usuarios. Así como los criterios establecidos para las rutas.

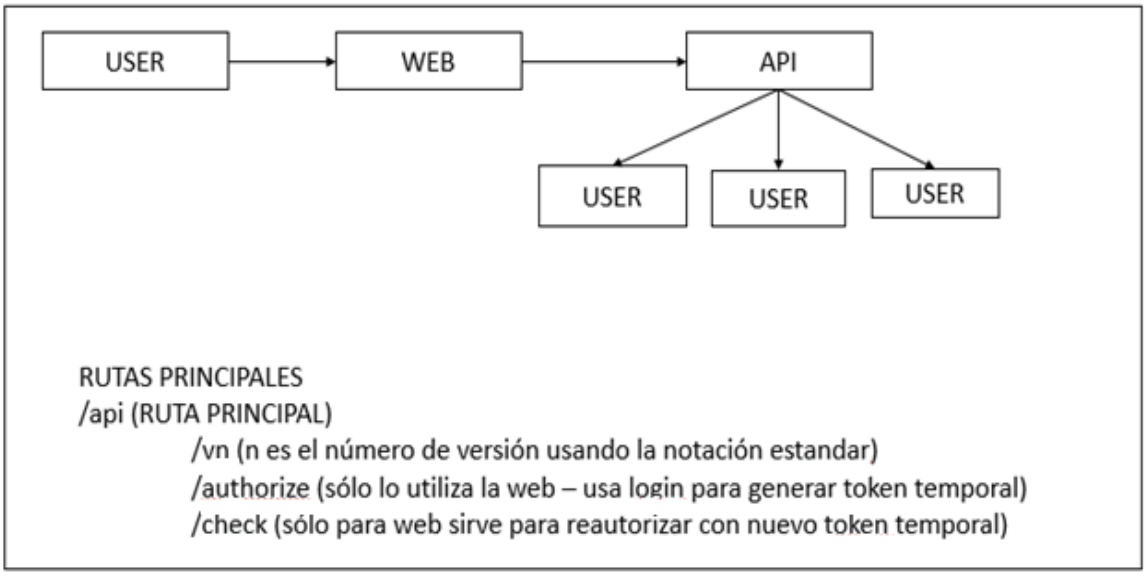


Gráfico 9. **Pautas para rutas de idal core y acceso de usuario.**

Una vez establecido el comportamiento del core se procedió a definir cuáles serían las clases principales del core. Teniendo como resultado las siguientes:

- Usuario.
- Namespace.
- Esquema.
- Columna.
- Datos.

Para la *Cuarta iteración* se procedió a especificar el comportamiento de las diferentes clases. Empezando por la clase Usuario y luego pasando a las clases correspondiente a la estructura de los datos almacenados, es decir referente a los namespaces, esquemas y columnas, los cuales se gestionan

en MySql. Para ello se tomaron en cuenta las rutas implicadas en cada caso. En este punto, se realizaron diversas reuniones vía Skype con el Scrum Master para definir el comportamiento de las clases de forma tentativa, pasando por correcciones y pruebas que permitieran llegar a las clases definitivas.

Empezando por la clase Usuario encargada de brindar las diversas funciones correspondientes a los diferentes niveles de acceso para los usuarios, gestionar las llaves de acceso y mantener el control sobre los tokens. Los elementos o atributos de clase comprenden:

- uid: Identificador del usuario
- alias: Nombre del usuario
- role: Nivel de acceso (USER, ADMIN, USER). Al registrarse se le asigna como rol USER
- estado: Estado del usuario (ACT, BLOQ, ELIM)
- fecha_registro: Fecha en la que se realizó el registro
- términos_de_servicio: Indica si aceptó los términos de servicio
- correo: Correo del usuario que le permite loguearse
- clave: Contraseña hash del usuario
- clave_salt: Salt asignado a la contraseña
- algoritmo: Algortimo utilizado para el encriptamiento de la contraseña
- confirmado: Indica si el usuario confirmo su cuenta. Al momento del registro se le asigna false
- long_salt : Longitud de salt asignado a la contraseña

Además, en este punto se definieron los atributos de la llave de acceso (apikey) que es asignada en el registro de usuario, estos comprenden:

- creado: Fecha de creación del apikey.

- llave: Token codificado (apikey).
- activo: Indica si el apikey está activo.

En caso de que el apikey se vea comprometido el usuario o el administrador puede bloquearla y generar un nuevo apikey. Para esta clase la ruta base establecida para los usuarios registrados fue: api/v1/user/*. Además, para el CRUD correspondiente a los usuarios y el resto de las clases se procedió a utilizar los protocolos POST, PUT, GET, DELETE para identificar la acción a realizar. Las rutas establecidas bajo api/v1 son las siguientes:

- /register. Registrar un nuevo usuario asignándole apikey (Token Permanente). En este punto se vio la necesidad de realizar la confirmación del usuario a través de un correo que le permita confirmar su cuenta (ver Gráfico 10). Por eso se creó una base de datos en Redis que permita mantener el control en este caso. Al registrarse un usuario se almacena en Redis con una estructura llave-valor el correo y un token generado con correo, fecha y alias, utilizando como semilla el certificado rsa. Cuando el usuario presiona el botón confirmar cuenta se verifica que el token esté en Redis, se actualiza en MySql que el usuario confirmó la cuenta y se elimina de Redis al mismo por si intenta usar de nuevo el formulario enviado. Además, para el almacenamiento de contraseñas se utilizó sha512 como algoritmo de encriptamiento para generación de hash y además se le asigna a cada contraseña un salt que brinda una capa de seguridad extra.
- /authorize. Genera un token temporal (tmpkey) a un usuario registrado que acceda a Inventame Data Access Layer por la web.
- /check/{token}. Esta ruta se encarga en caso de token expires de verificar el campo invalidBefore. De manera que, si está en este rango genera un

nuevo tmpkey para alargar la sesión. De lo contrario, se desloguea al usuario.

Para el establecimiento de las rutas authorize y check se realizaron diagramas de interacción que permitieran observar los distintos casos a considerar y así establecer el comportamiento de las mismas. Nuevamente, es importante acotar que estos no forman parte de SCRUM, sin embargo, se realizaron de manera conjunta con el Scrum Master, persiguiendo la posibilidad de visualizar mejor la secuencia correspondiente al logueo en la aplicación (ver Gráficos 11 a 13) y lograr un entendimiento mutuo y certero del funcionamiento. Asimismo, en los Gráficos 14 y 15 se muestra el código fuente correspondiente a las rutas authorize y check, que implementan los diagramas de interacción antes referenciados, ya que tal como lo fomenta la filosofía de desarrollo ágil, no hay mejor documentación del sistema que el código en sí mismo.

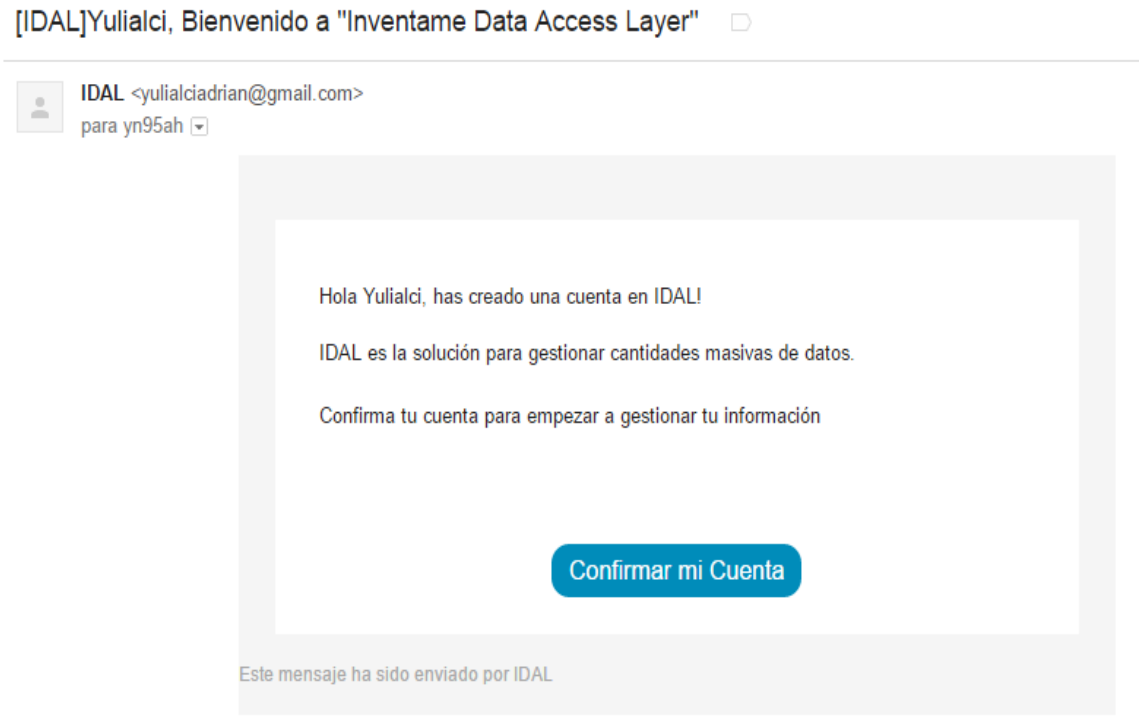


***Gráfico 10.*Correo enviado al usuario para confirmación de cuenta.**

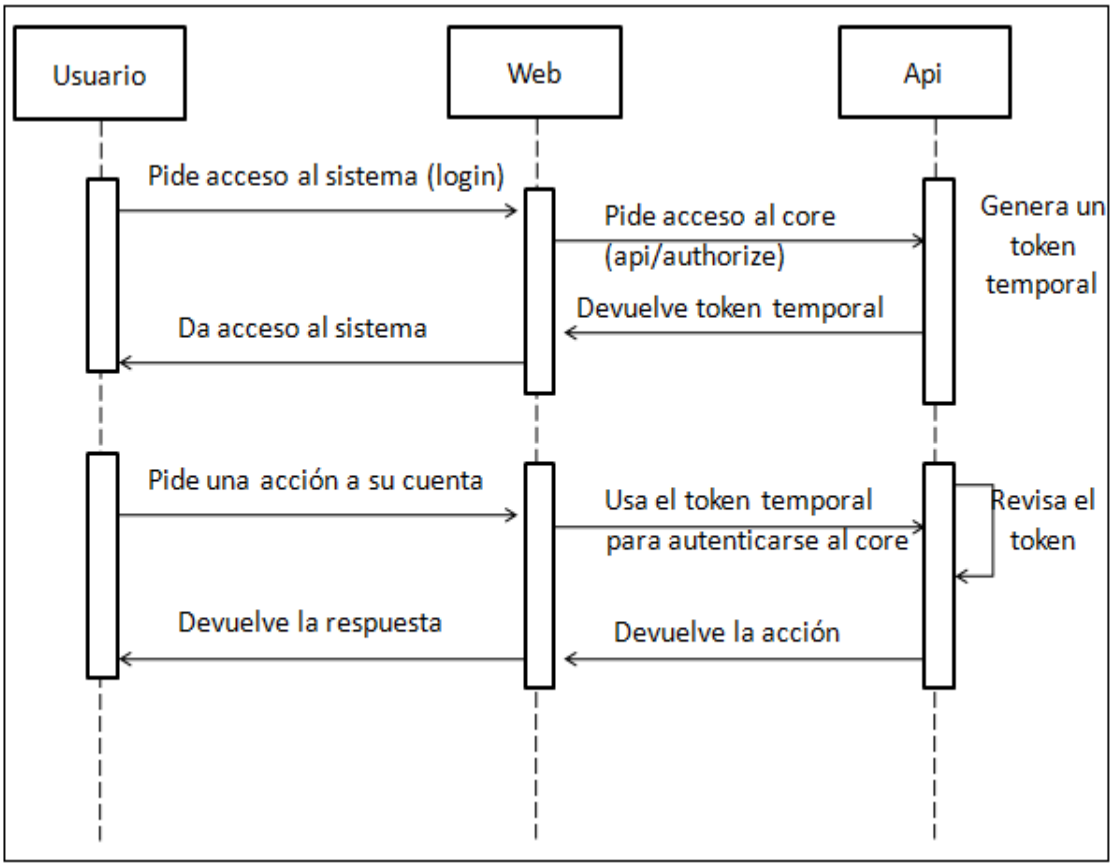


Gráfico 11. **Diagrama de Interacción – Login Web.**

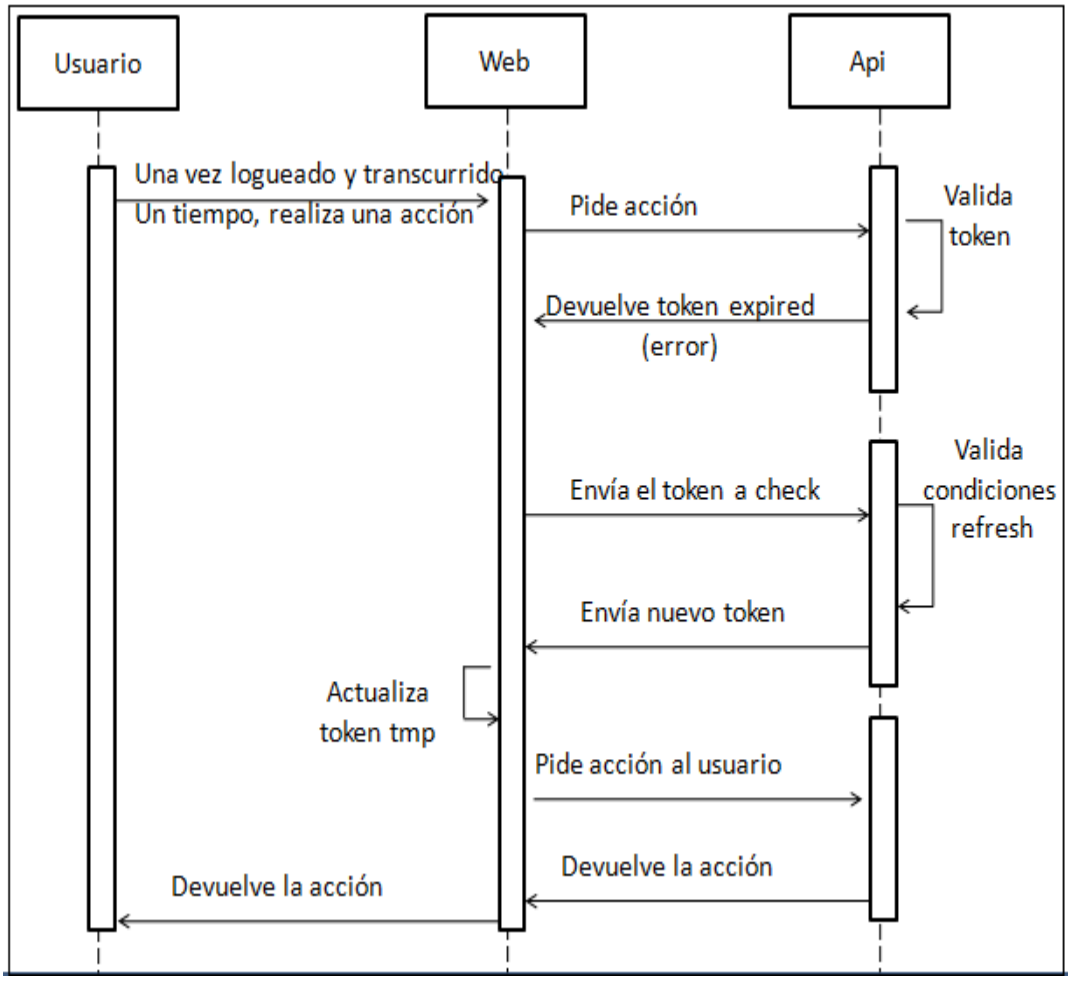


Gráfico 12. **Diagrama de Interacción – Acción con token caducado.**

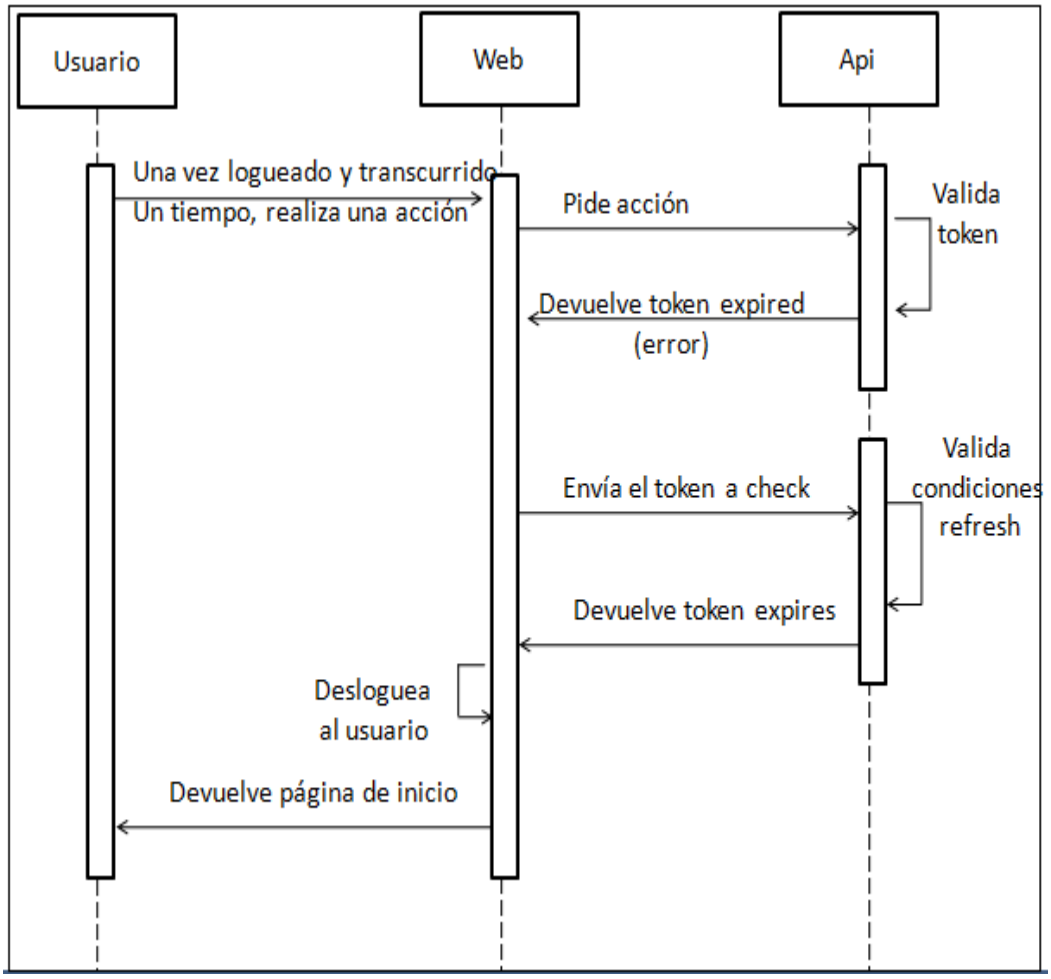


Gráfico 13. **Diagrama de Interacción – Acción con token caducado sin refresh.**

```
import * as Fetch from '../../Utiles/Fetch.jsx'

export const Login = ( _username, _password )=>{
    return ( _dispatch, _state )=>{
        if( _username == '' || !validateEmail( _username ) )
            return _dispatch({type:'ERROR_USERNAME_EMPTY', usernameError:'Indica un correo válido', usernameS
        _dispatch({type:'VALID_FIELD', usernameError:'', usernameStatus:'success'})
        if( _password == '' )
            return _dispatch({type:'ERROR_PASSWORD_EMPTY', passwordError:'Indica la constraseña', passwordSta
        _dispatch({type:'VALID_FIELD', passwordError:'', passwordStatus: ''})


        var data = {
            correo: _username,
            clave: _password,
            version: "1.0.0"
        }
        _dispatch({type:'FETCH_START', loading: true})
        Fetch.post( `${Fetch.api}/authorize`, data )
            .then(_response =>{
                _dispatch({type:'FETCH_STOP', loading: false})
                if( _response.data.tmp_key ){
                    localStorage.auth = _response.data.tmp_key
                    localStorage.email = _username
                    _dispatch({type: 'REVIEW_AUTH',cookie:true})
                }
            })
            .catch(_error =>{
                _dispatch({type:'FETCH_STOP', loading: false})
                _dispatch({type:'ERROR_CONEXION', error: _error.message})
            })
    }
```

Gráfico 14. **Código correspondiente al login de usuario, utilizando authorize.**

```
__http ( _method, _url, _data) {
    return new Promise ( (_done,_fail) => {
        let req = {}
        if (_method == 'GET'){
            req = Fetch.get( `${Fetch.api}${_url}` )
        }else
            req = Fetch.post( `${Fetch.api}${_url}`, _data )
        req
        .then (_response=>{
            _done(_response)
        })
        .catch (_response => {
            if (_response.code != 498){
                this.__logout()
                _fail()
            }else{
                Fetch.removeDefaultAuth()
                Fetch.get( `${Fetch.api}/check/${localStorage.auth}`)
                .then (_response =>{
                    localStorage.auth = _response.data.tmp_key
                    Fetch.addDefaultAuth( _response.data.tmp_key)
                    this.__http(_method, _url, _data)
                    .then(_response => {
                        _done(_response)
                    })
                    .catch(_response => {
                        this.__logout()
                        _fail()
                    })
                })
                .catch (_response =>{
                    this.__logout()
                    _fail()
                })
            }
        })
    })

}
```

Gráfico 15. **Código correspondiente a la verificación de token, utilizando check.**

Por su parte, bajo la ruta api/v1/user se procedió a trabajar con los protocolos y se crearon las rutas para el administrador (ver Cuadro 3).

Cuadro 3.

Rutas Correspondientes api/v1/user/*

Protocolo	Rol	Ruta	Descripción
GET	USER	api/v1/user/	Permite obtener los datos del usuario para mostrarlos
PUT	USER	api/v1/user/	Para actualizar alias o clave del usuario
GET	USER	api/v1/user/key	El usuario consulta la apikey activa.
GET	USER	api/v1/user/lock_key	El usuario bloquea apikey activa y genera una nueva.
GET	ADMIN	api/v1/user/list	Para ver listado de usuarios.
DELETE	ADMIN	api/v1/user/{id}	Para eliminar un usuario determinado.
GET	ADMIN	api/v1/user/block/{id}	Para bloquear un usuario determinado.
GET	ADMIN	api/v1/user/active/{id}	Para activar un usuario bloqueado.

(Cont.) Cuadro 3

GET	ADMIN	api/v1/user/key/{id}	Para consultar apikeys de un usuario
GET	ADMIN	api/v1/user/{id}	Datos de un usuario determinado
GET	ADMIN	api/v1/user/lock_key/{id}	Bloquea apikey a un usuario determinado
POST	ADMIN	api/v1/user/update_role/{id}	Actualizar rol de un usuario determinado

Una vez definido el comportamiento de la clase usuario se procedió a la codificación y finalmente se realizaron las pruebas correspondientes. Para las mismas se utilizó Postman que es una extensión gratuita para el navegador Google Chrome que permite probar servicios web fácilmente. Además, a través de la librería frisby, se realizó un test unitario que permitirá reproducir al pasar a producción, lo cual excede del ámbito de este trabajo de investigación.

Posteriormente se procedió a definir el comportamiento de la clase namespace. Los atributos de esta clase son los siguientes:

- id_namespace: Identificador del namespace.
- alias: Nombre del namespace indicado por el usuario.
- estado: Estado del namespace (ACT - ELIM).
- n_schemas: Número de esquemas del namespace.

Para esta clase se utiliza como ruta base api/v1/namespace. Bajo el mismo concepto utilizado para los usuarios se establecieron las siguientes rutas:

Cuadro 4.

Rutas Correspondientes api/v1/namespace/*

Protocolo	Rol	Ruta	Descripción
POST	USER	api/v1/namespace/	Crear un namespace vacío
PUT	USER	api/v1/namespace/{id}	Actualizar alias de namespace
GET	USER	api/v1/namespace/list	Consultar listado de namespaces
GET	USER	api/v1/namespace/{id}	Consultar un namespace determinado
DELETE	USER	api/v1/namespace/{id}	Eliminar un namespace
GET	ADMIN	api/v1/namespace/list/{id_user}	Listado de namespaces de un usuario

Una vez discutida la clase namespace, se codificó, se realizaron las pruebas pertinentes, pasando luego a la definición de la clase esquema. Los atributos son los siguientes:

- id: Identificador del esquema.

- n_col: Número de columnas que tiene el esquema.
- alias: Nombre del esquema asignado por el usuario.
- description: Descripción del esquema.

Además, para esta clase se establecieron las rutas bajo la ruta base api/v1/schema (ver Cuadro 5)

Cuadro 5.

Rutas Correspondientes api/v1/schema/*

Protocolo	Rol	Ruta	Descripción
POST	USER	api/v1/schema/	Registrar un esquema vacío
PUT	USER	api/v1/schema/{id}	Para actualizar alias y description del esquema
DELETE	USER	api/v1/schema/{id}	Eliminar un esquema determinado
GET	USER	api/v1/schema/list/{id_namespace}	Esquemas de un namespace determinado

Finalmente se pasó a la definición de la clase columna, la cual es de suma importancia pues ésta clase es la que se relaciona de forma directa con los datos en Cassandra. Los atributos comprenden:

- id: Identificador de la columna.
- alias: nombre de la columna asignado por el usuario.
- key: Indica si el campo es la clave primaria.
- type: Tipo de dato (String – Number - Real).
- isnull: Indica si el campo admite valores nulos
- default: Valor por defecto que se le asigna a la columna, en caso de no admitir nulos.

Por su parte, la ruta base para esta clase es api/v1/column (ver Cuadro 6).

Cuadro 6.

Rutas Correspondientes api/v1/column/*

Protocolo	Rol	Ruta	Descripción
POST	USER	api/v1/column/ {id_esquema}	Añadir columna a un esquema determinado
PUT	USER	api/v1/column/{id}	Actualizar opciones de la columna. En caso de contener datos en Cassandra, sólo puede actualizar alias
DELETE	USER	api/v1/column/{id}	Eliminar una columna
GET	USER	api/v1/column/ {id_esquema}	Listado de columnas de un esquema
GET	USER	api/v1/column/{id}	Datos de una columna determinada.

En este punto se estableció que si al añadir una nueva columna a un esquema ya existen datos en Cassandra, se le asignará a los registros existentes el valor null si lo admite o de lo contrario el valor por defecto.

Además, no se permite la actualización de las diferentes opciones para la columna si ya se han almacenado datos en Cassandra para la misma. Por último, al hacer un delete es importante verificar que ésta no sea clave primaria.

La *Quinta Iteración* fue dedicada a definir la clase de datos, la cual se corresponde con el CRUD de los datos pero sin el manejo en lote, es decir, en este punto se estableció el comportamiento de los datos para el web-server a nivel de cada registro. La ruta base en general para el manejo de los datos es api/v1/warehouse, para esta iteración se establecieron las siguientes rutas iniciales:

Cuadro 7.

Rutas Correspondientes api/v1/warehouse/*

Protocolo	Rol	Ruta	Descripción
POST	USER	api/v1/warehouse/ {id_esquema}	Inserta un nuevo registro para un esquema determinado en Cassandra
PUT	USER	api/v1/warehouse/ {id_esquema}	Actualiza valores de un registro indicando su clave primaria
DELETE	USER	api/v1/warehouse/ {id_esquema}	Eliminar un registro indicando su clave primaria
GET	USER	api/v1/warehouse/list/ {id_esquema}	Consultar registros de un esquema
POST	USER	api/v1/warehouse/get /{id_esquema}	Consultar un registro indicando clave primaria

Como se puede observar las rutas definidas para esta iteración comprenden el manejo individual o simple de los registros no abarca el manejo en lote. A su vez, se observa que para realizar cambios en un

registro o consultarlo es necesario indicar la clave primaria, ésta permite ubicar en Cassandra el registro solicitado. Por lo que, la clave primaria en este punto es de sólo lectura, no se permite modificarla.

El manejo en lotes se dejó para una *Sexta Iteración*, lo que permitió verificar que el funcionamiento del CRUD para registros de forma individual se estaba realizando correctamente. Una vez logrado esto, entonces se añadieron nuevas rutas a clase de datos para el manejo en lote (batch). Fue necesario realizar reuniones a distancia continuamente con mayor frecuencia, ya que estas rutas son de suma importancia para correcto funcionamiento de la plataforma. Para el manejo de la data en lotes la ruta base es api/v1/warehouse/batch (ver Cuadro 8).

Cuadro 8.

Rutas Correspondientes api/v1/warehouse/batch/*

Protocolo	**Rol**	**Ruta**	**Descripción**
POST	USER	api/v1/warehouse/{id_esquema}/batch	Insertar registros en lote
PUT	USER	api/v1/warehouse/{id_esquema}/batch	Actualizar registros en lote
DELETE	USER	api/v1/warehouse/{id_esquema}/batch	Eliminar registros en lote
POST	USER	api/v1/warehouse/{id_esquema}/get/batch	Consultar registros en lote

En el caso de la inserción en lotes, ésta se realiza a través de archivos en formato JSON, XML, CSV y EXCEL, lo cuales contienen los registros a

insertar, esto es manejado principalmente por el web-server. A continuación se muestran ejemplos de cada uno de ellos:

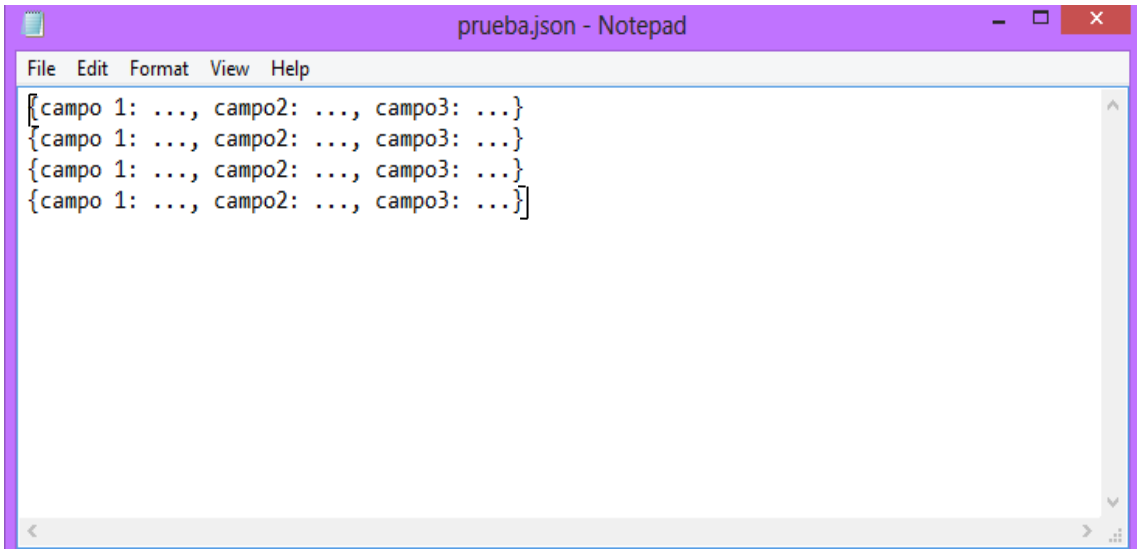


Gráfico 16. **Ejemplo Archivo JSON.**

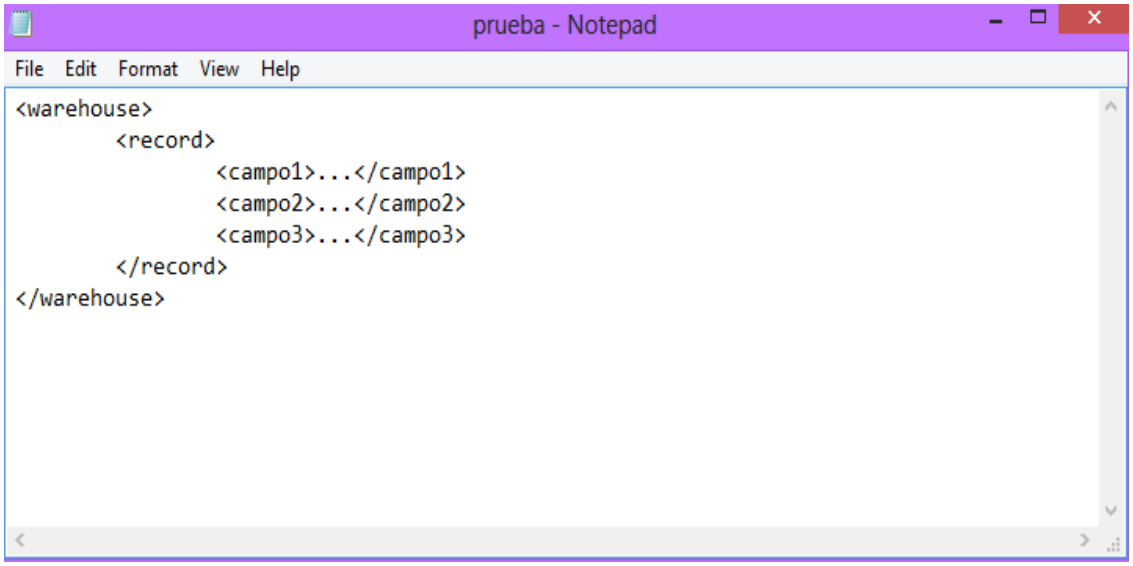
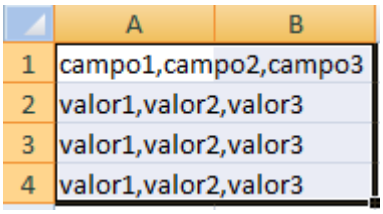


Gráfico 17. **Ejemplo Archivo XML.**



	A	B
1	campo1,campo2,campo3	
2	valor1,valor2,valor3	
3	valor1,valor2,valor3	
4	valor1,valor2,valor3	

Gráfico 18. **Ejemplo Archivo CSV.**

	A	B	C
1	campo1	campo2	campo3
2	valor1	valor2	valor3
3	valor1	valor2	valor3

Gráfico 19. **Ejemplo Archivo EXCEL.**

Para las consultas en lotes se establecieron las opciones que el usuario debe indicar:

- page: Correspondiente a la paginación. Por defecto es cero (0).
- pageSize: Tamaño de una página. Por defecto cien (100).
- search: Arreglo con campos a consultar. Por defecto *. Equivalente a SELECT *.
- filter: Arreglo con campos que permitirán filtrar los datos. Por defecto null. Equivalente a WHERE x…. AND y….
- ffilter: Arreglo de JSON con los valores correspondientes al filtrado. En este se indica el operador y el valor a ser comparado. Por defecto null. Operadores permitidos: =, !=, >, <, >=, <=
- order: Criterio de ordenamiento. Puede ser ascendente (ASC) o descendente (DESC). Por defecto se ordena ascendente.
- sort: Columna por la que se desea ordenar. Por defecto se ordena por la clave primaria.

Para comprender su funcionamiento a continuación se plantean algunos ejemplos:

- SELECT * FROM data WHERE x=1 AND y >= 2 ORDER BY x

  {
  search: *
  filter: [x,y]
  ffilter: [{op:=,value:1},{ op:>=, value: 2}],
  sort: x
  }
- SELECT a, b FROM data WHERE x = 2.5

  {
  search: [a,b]
  filter: [x]
  ffilter: [{op:=,value:2.5}
  }

Para actualizar y eliminar registros en lote, se sigue el mismo funcionamiento. El parámetro search para la actualización se denomina update y para indicar los valores de cada SET se utiliza un nuevo campo fupdate, que es un arreglo con estos valores. Además, para eliminar registros en lote el campo search se denomina delete.

Una vez completado el core y haber realizado las pruebas pertinentes, se procedió a generar la documentación de REST API, llegando así a la *Séptima Iteración*, aunque este paso puede ser algo tedioso es de suma importancia para realizar mejoras o cambios en la API. Para ello se utilizó el estilo Swagger, el cual es una herramienta muy útil para describir, producir, consumir y visualizar APIsRESTful. De manera que, la API se documente a medida que surgen cambios. En este sprint se documentaron las diversas clases (ver Gráficos 20 a 24).

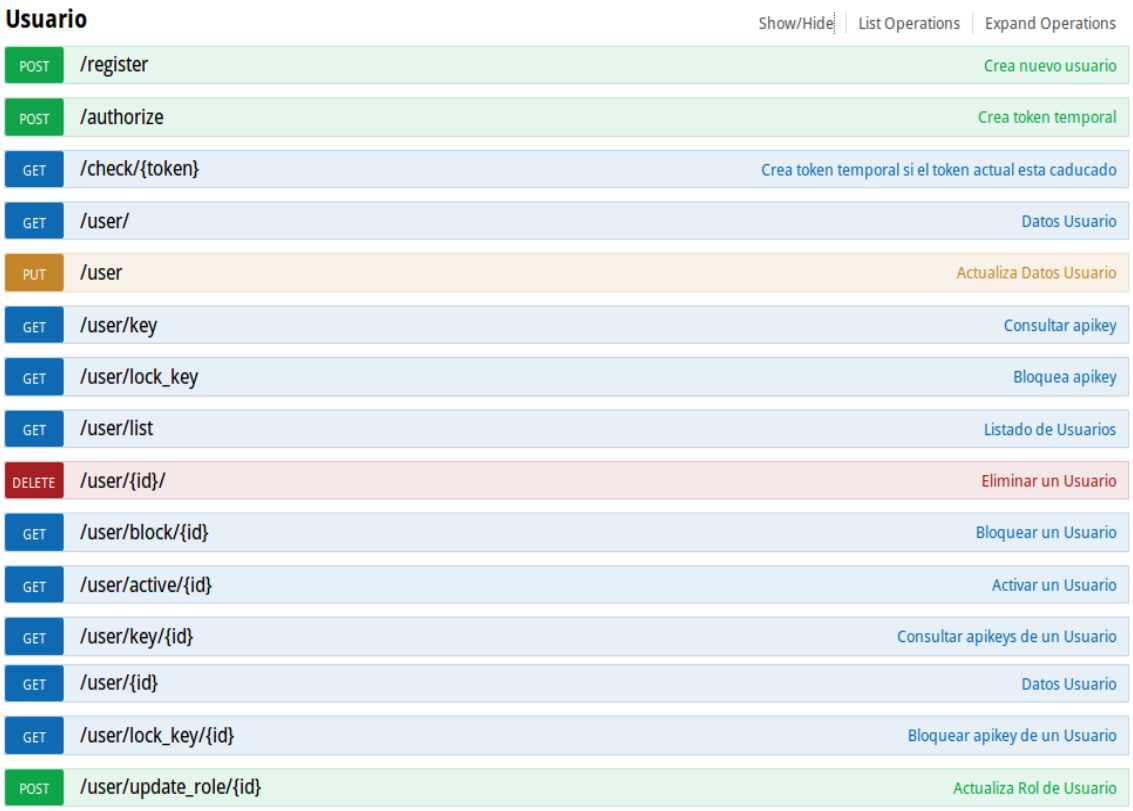


***Gráfico 20.*DocumentacionSwagger Clase Usuario.**

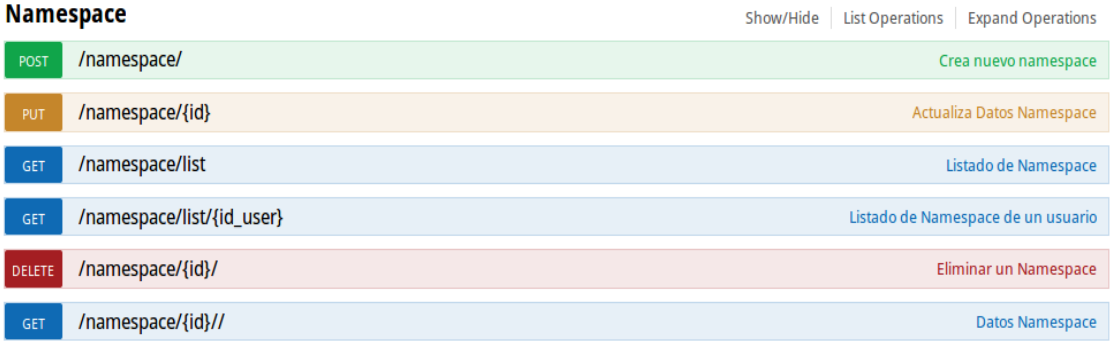


***Gráfico 21.*DocumentacionSwagger Clase Namespace.**

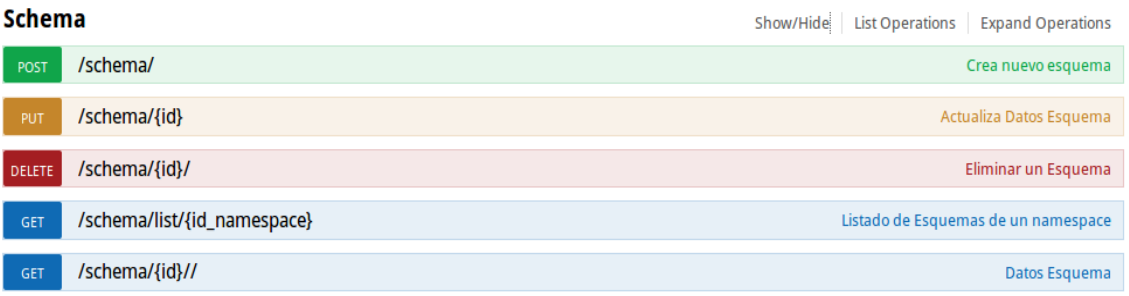


***Gráfico 22.*DocumentacionSwagger Clase Schema.**

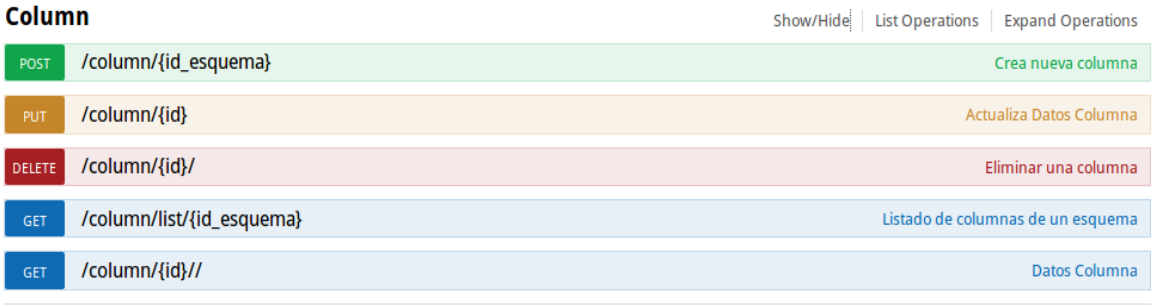


***Gráfico 23.*DocumentacionSwagger Clase Columna.**

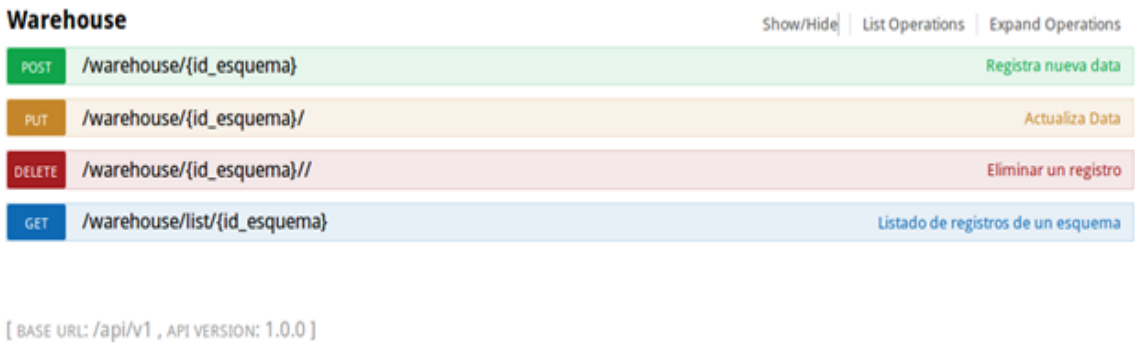


***Gráfico 24.*Documentacion Inicial Swagger Clase Data.**

Finalmente, en la *Octava Iteración* se procedió a realizar el diseño de la interfaz correspondiente al frontend del web-server éste fue revisado por el Scrum Master, se realizaron las correcciones y mejoras pertinentes llegando a la interfaz final. El diseño se realizó en React JS utilizando Ant Design para el diseño de los componentes.

Inicialmente se definieron los componentes básicos a utilizar, tales como:

- Login: Pantalla Login.
- NewUser: Registro de Usuario.
- ForgetPassword: Olvidó Contraseña.
- Profile: Perfil.
- TableNamespace: Tabla Namespaces.
- TableSchema: Tabla Esquemas.
- TableColumns: Tabla Columnas.
- TableData: Tabla Data.
- Import: Importar Data.
- Query: Consultas en lote.

Para el manejo de los componentes se procedieron a establecer las escenas respectivas, lo cual consiste en agrupar dichos componentes. De manera que, por ejemplo las tablas cuenten con las distintas acciones tales como editar, eliminar, insertar, entre otros. Para la conexión con idal-core se utilizó la librería fetch y el cambio de state en React JS se realiza a través de React-Redux. Además, para las rutas se utilizó React-Router.

Una vez establecidas las escenas, se procedió a establecer el prototipo base del web-server el cual se muestra a continuación en el Gráfico 25

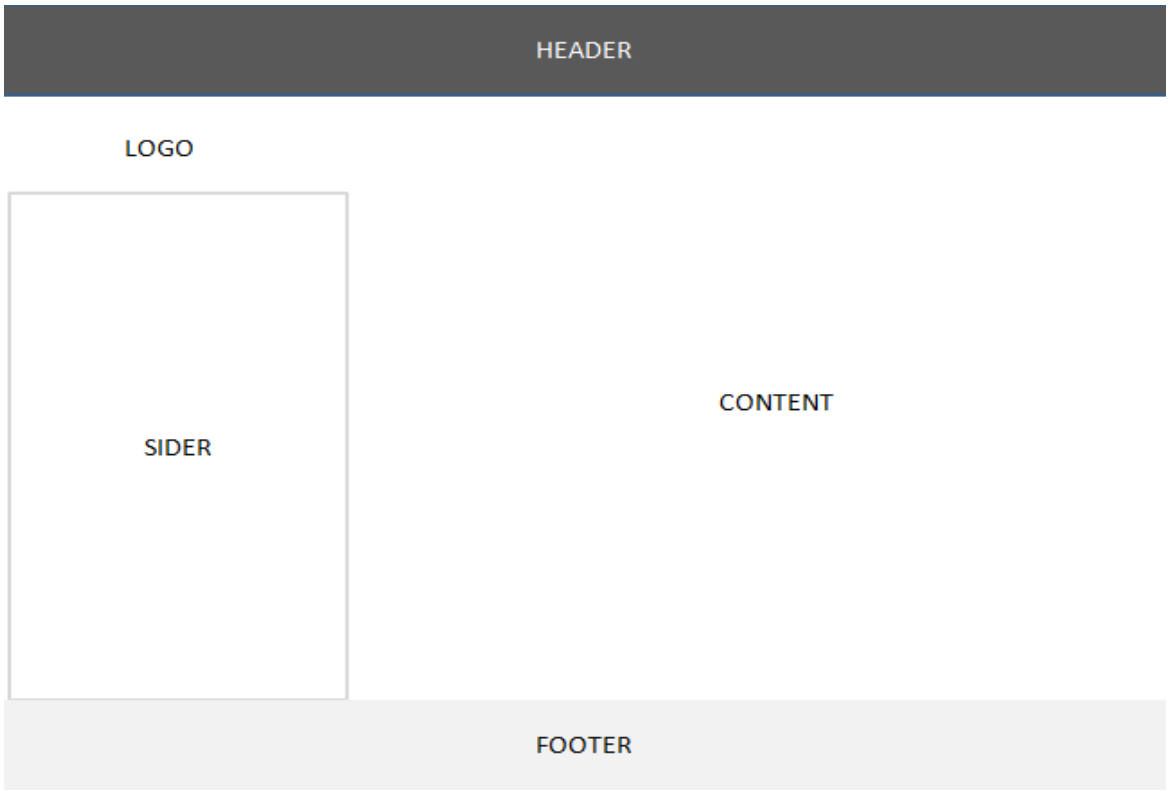


Gráfico 25. **Prototipo Estructura idal-web-server.**

Una vez establecido el prototipo inicial se procedió al diseño de las pantallas valiéndose de Ant Design, empezando por el login (ver Gráfico 26) en el cual el usuario ingresa su username (correo) y password.

Gráfico 26. **Formulario Login.**

El enlace regístrate ahora, lleva al componente de registro (ver Gráfico 27) que consiste en un modal donde el usuario debe indicar sus datos y aceptar los términos de servicio. Una vez realizado el registro se le envía un correo de confirmación (ver Gráfico 10).

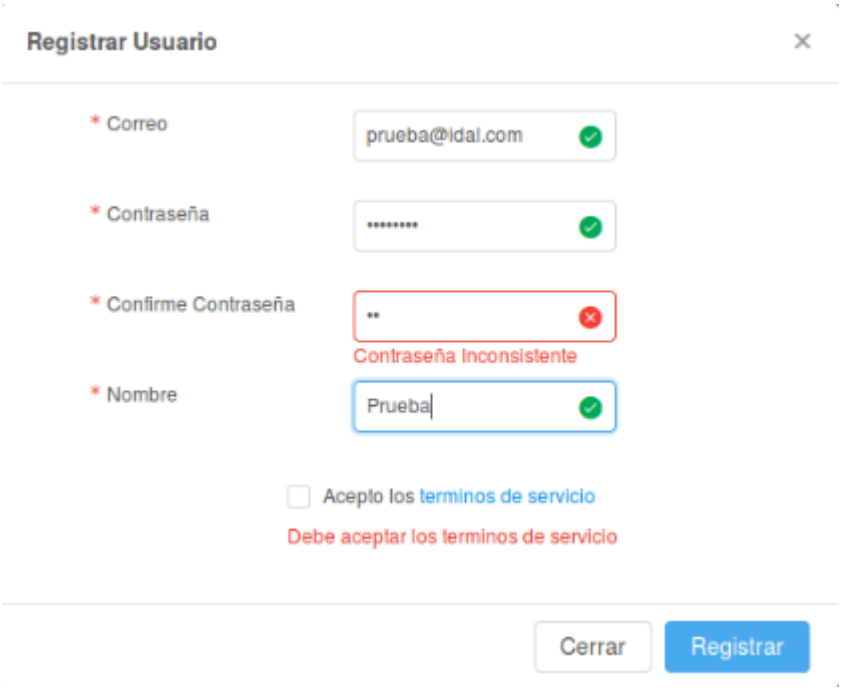


Gráfico 27. **Formulario Modal Registro de Usuario.**

En caso de que el usuario olvide su contraseña el enlace Olvidaste tu Clave? lo lleva al modal de solicitar cambio de contraseña (ver Gráfico 28), donde debe indicar su username (correo), al cual se le enviará un correo en el cual se le indica que ha solicitado el cambio (ver Gráfico 29). Cabe destacar, que para esto se crea un token en Redis tal como es el caso de la confirmación de registro, con la diferencia de que es un token temporal con una duración de veinticuatro (24) horas, por lo que el usuario cuenta con este lapso de tiempo para realizar el cambio.

Gráfico 28. **Formulario Modal Solicitar Cambio de Contraseña.**

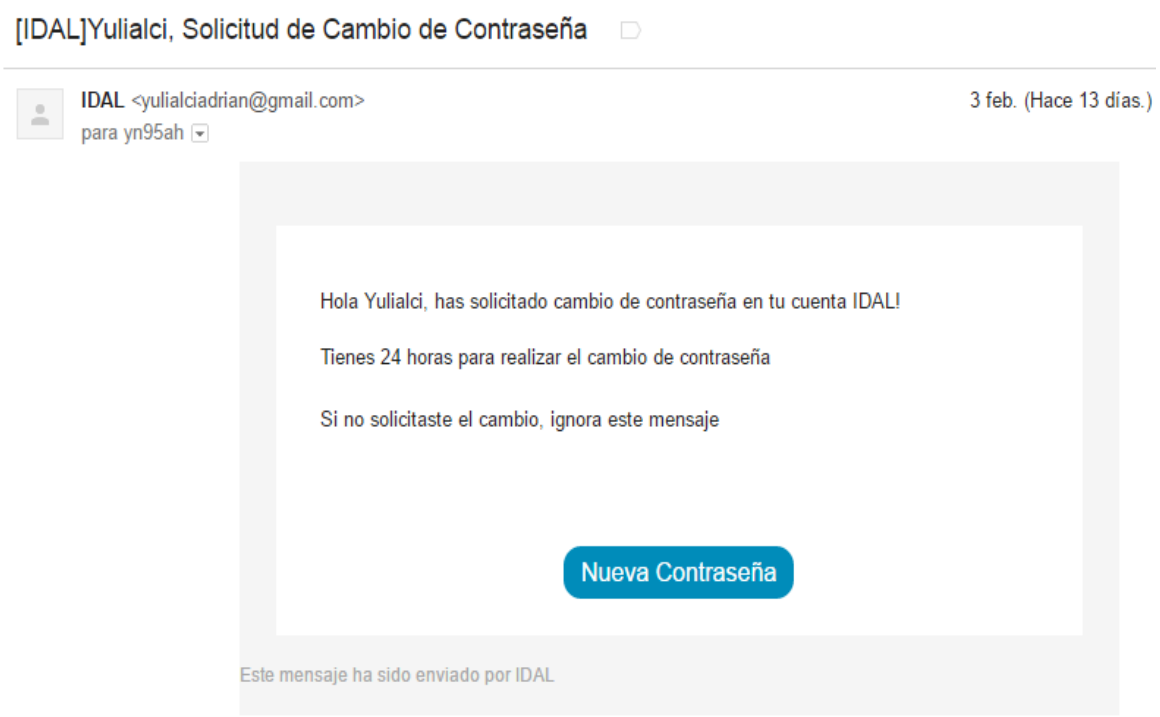


Gráfico 29. **Correo enviado al usuario para cambio de contraseña.**

Para el usuario registrado, el index una vez logueado cuenta con los componentes del prototipo correspondiente a la estructura (ver Gráfico 30), el cual en el header tiene un menú con dos (2) ítems:

- Namespaces: Para ver listado de namespaces.
- Nombre de Usuario: Popover con enlaces a perfil y a salir o cerrar sesión.

En el sider cuenta con un loading tree con el listado de namespaces el cual se expande y muestra el listado de esquemas. Éste permite ver listado de esquemas de un namespaces y data de un esquema.

En el content se muestra el listado de namespaces en una tabla. En este punto es importante explicar el significado de los iconos a encontrar en la tabla.

- Ordenar Ascendente o Descendente (ver Gráfico 31).
- Buscar (ver Gráfico 32).
- Editar (ver Gráfico 33).
- Guardar Cambios.
- Cancelar.
- Eliminar (ver Gráfico 34).
- Consultar.

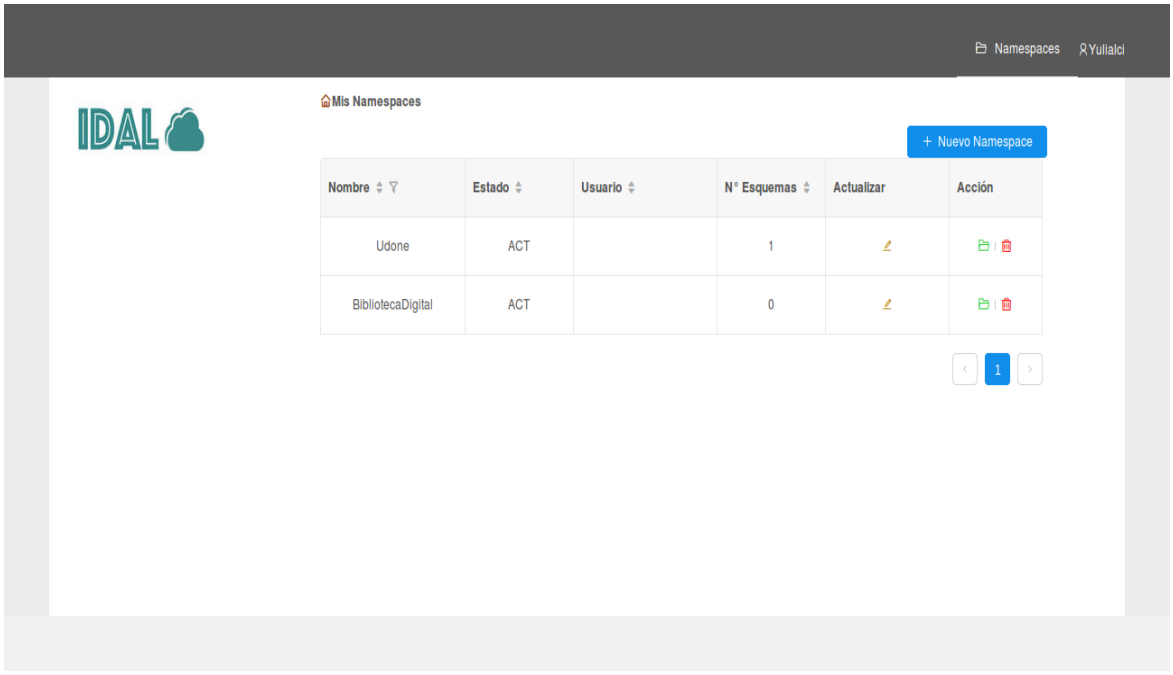


Gráfico 30. **Index Usuario Logueado.**

Gráfico 31. **Icono Ordenar en Tabla Namespace.**

Gráfico 32. **Icono Buscar en Tabla Namespace.**

Gráfico 33. **Editar en Tabla Namespace.**

Como se puede observar en el Gráfico 33 el único campo editable de un namespace es el nombre.

Gráfico 34. **Eliminar Namespace.**

Para registrar un namespace, debe utilizar el botón Nuevo Namespace, indicar el nombre (ver Gráfico 35) y éste se agregará a la tabla.

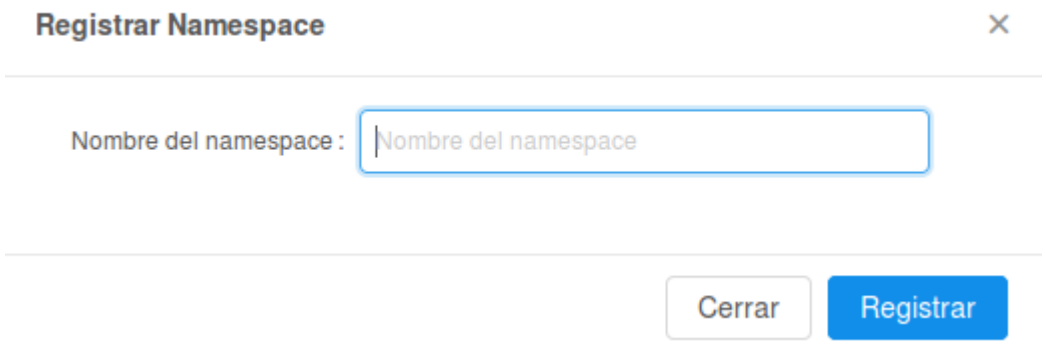


Gráfico 35. **Formulario Modal Registro Namespace.**

En caso de seleccionar la acción consultar, se le mostrará la tabla correspondiente a los esquemas del namespace seleccionado (ver Gráfico 36), ésta tabla cuenta con las mismas opciones de la tabla para los namespaces. Los campos editables son nombre y descripción (ver Gráfico 37).

Además, para agregar un nuevo esquema debe utilizar el botón correspondiente e indicar los datos del esquema (ver Gráfico 38). En este punto, es necesario indicar la estructura de las columnas correspondientes a dicho esquema, especificando el número de estas en el campo agregar.

Por último, la opción consultar lleva a la tabla de la data correspondiente al esquema seleccionado (ver Gráfico 39).

Gráfico 36. **Tabla Esquemas.**

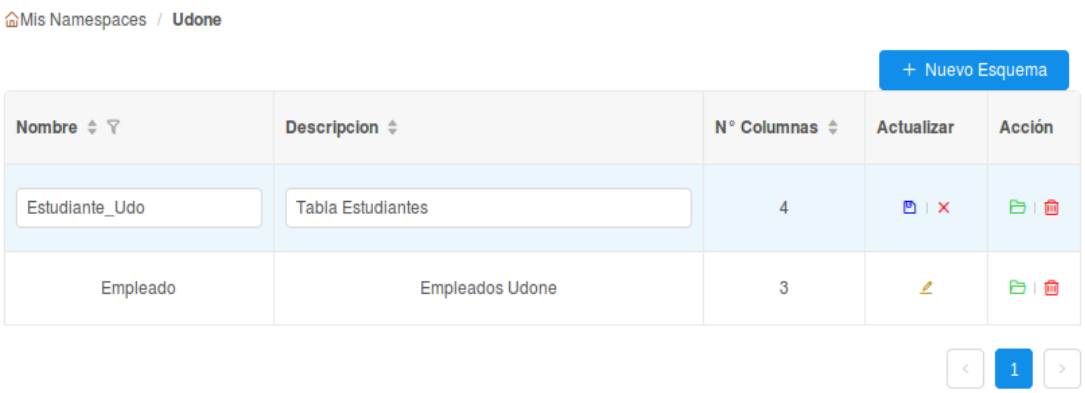


Gráfico 37. **Editar Esquema.**

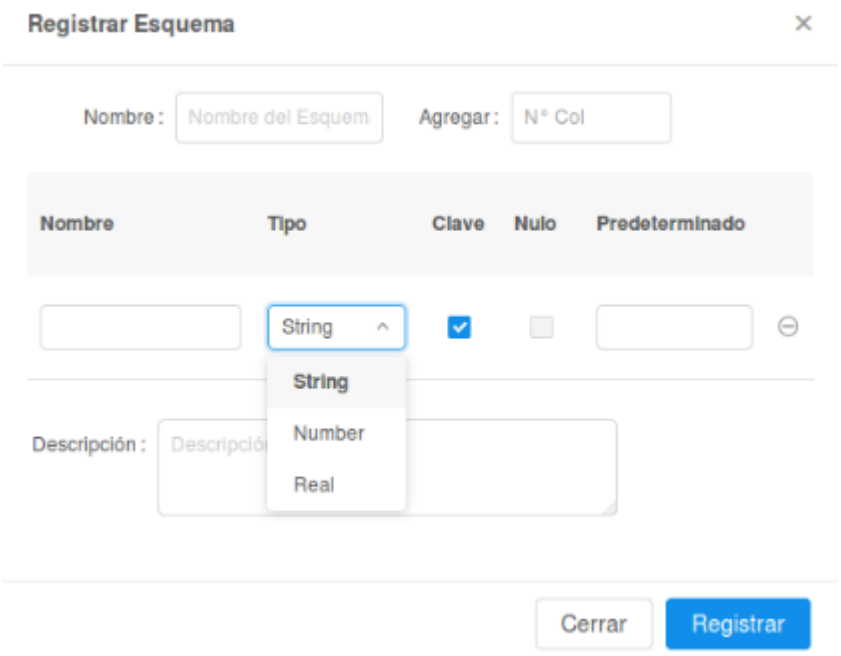


Gráfico 38. **Modal Registrar Esquema.**

Cedula	Nombre	Apellido	Promedio	Actualizar	Acción
12345678	Maria	Perez	9		
019875	Cesar	Hernandez	9		
332211	Luis	Salazar	1		
112233	Juan	Lopez	2		
22996846	Yulialci	Adrian	5		
110010	Carmen	Guerra	3		
87654321	Pedro	Suárez	1.2		

Gráfico 39. Tabla Data.

En esta tabla es posible actualizar cualquier campo excepto la clave primaria (ver Gráfico 40). Además, se cuenta con un grupo de botones que permitirán realizar acciones para el esquema, tales como:

- Nuevo Registro: Agregar nuevo registro al esquema (ver Gráfico 41).
- Importar: Importar data en JSON, XML, CSV o EXCEL (ver Gráfico 42).
- Consultas: Para consultas en lote (ver Gráfico 43).
- Estructura: Muestra Columnas del Esquema (ver Gráfico 44).

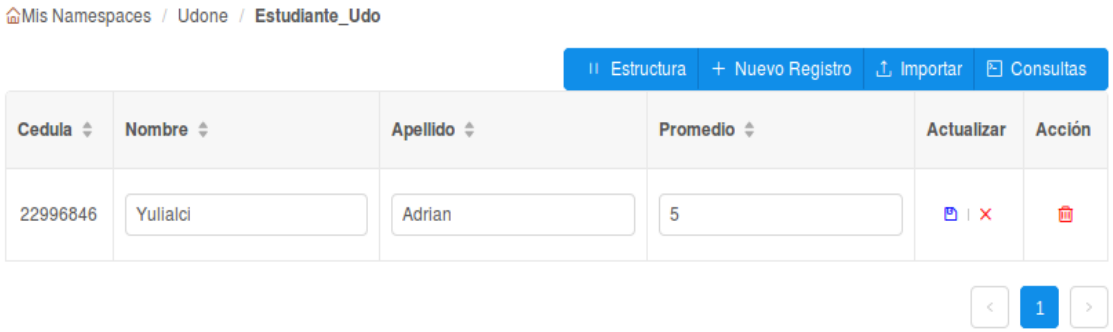


Gráfico 40. **Editar Data.**

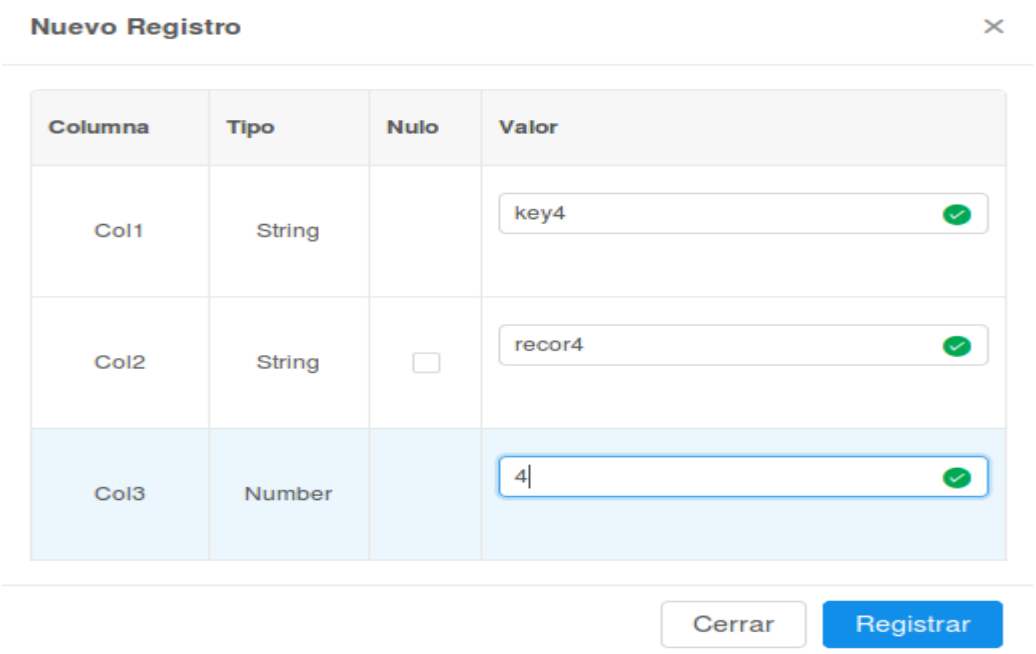


Gráfico 41. **Modal Registrar Data.**

Gráfico 42. **Importar Data.**

Para importar data solo se debe cargar el archivo y luego importarlo. Cabe destacar que en el content de importar data aparece en el grupo de botones la opción examinar, ésta permite regresar a la tabla data.

Por su parte, bajo la opción consultas el usuario indica los parámetros necesarios para realizar la misma, tales como, paginación, campos de búsqueda (a través de un transfer indicar los campos o columnas a buscar) y filtros (a través de una tabla indica los parámetros de filter y ffilter).

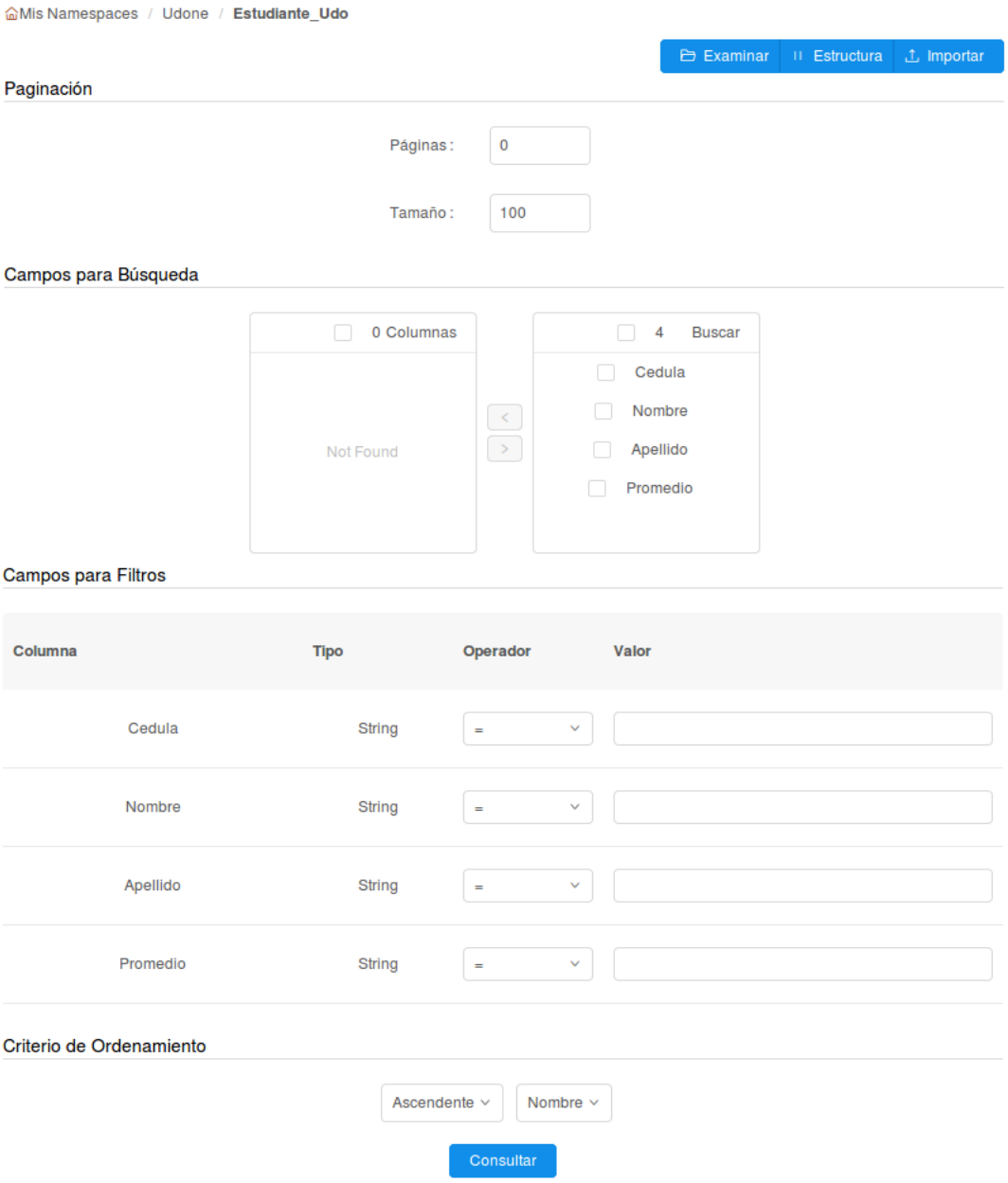


Gráfico 43. Consultas.

Para las columnas todos los campos serán editables si el esquema no tiene datos. De lo contrario, sólo es posible actualizar el nombre (Ver Gráficos 45 y 46). Además, se puede agregar una columna al esquema indicando las opciones respectivas (Ver Gráfico 47).

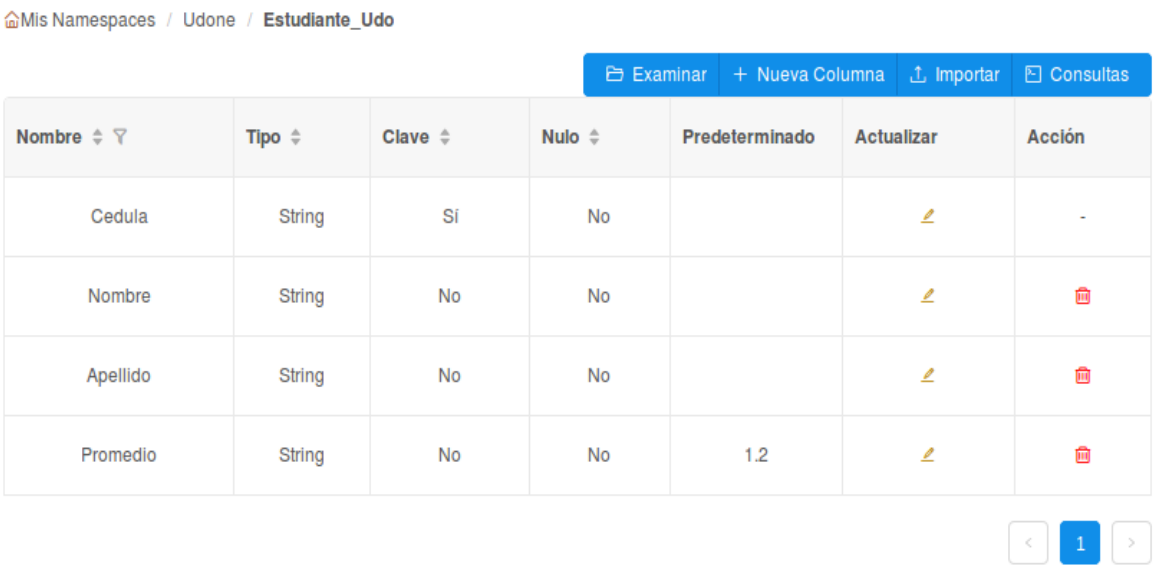


Gráfico 44. **Tabla Columnas.**

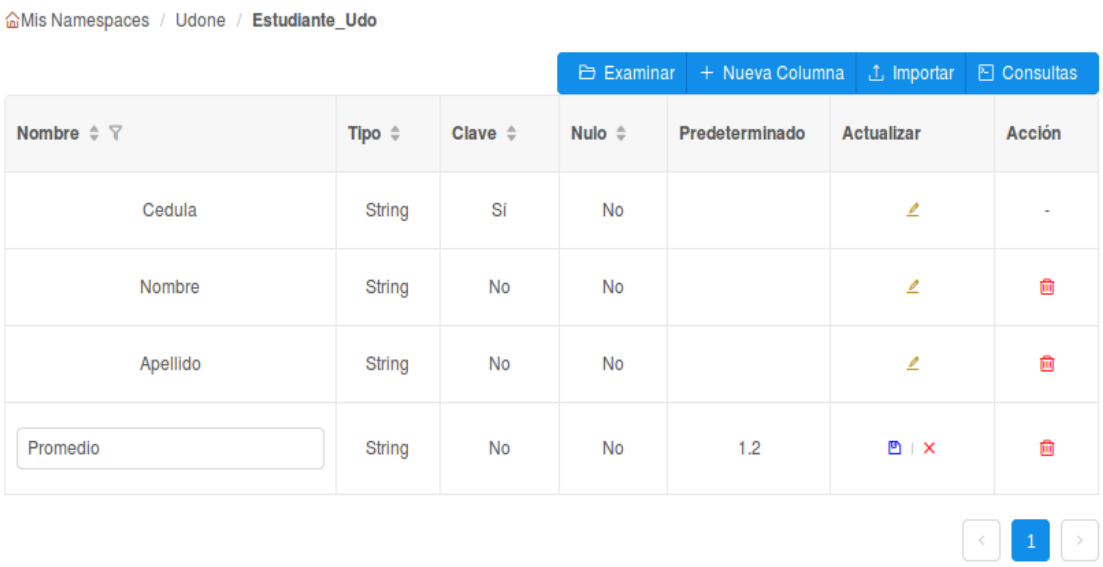


Gráfico 45. **Editar Columna de Esquema con Datos.**

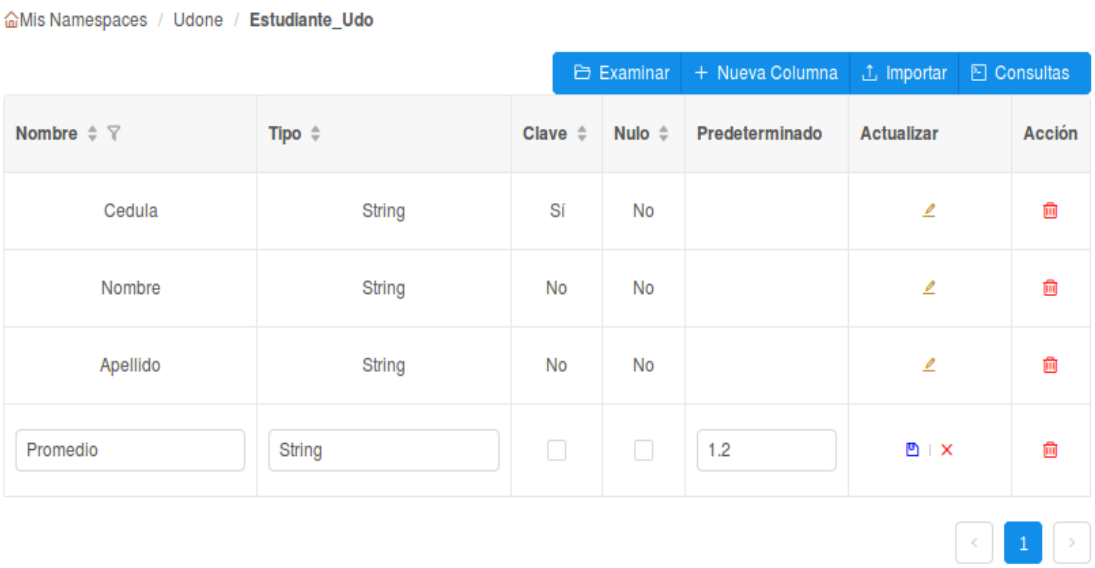


Gráfico 46. **Editar Columna de Esquema sin Datos.**

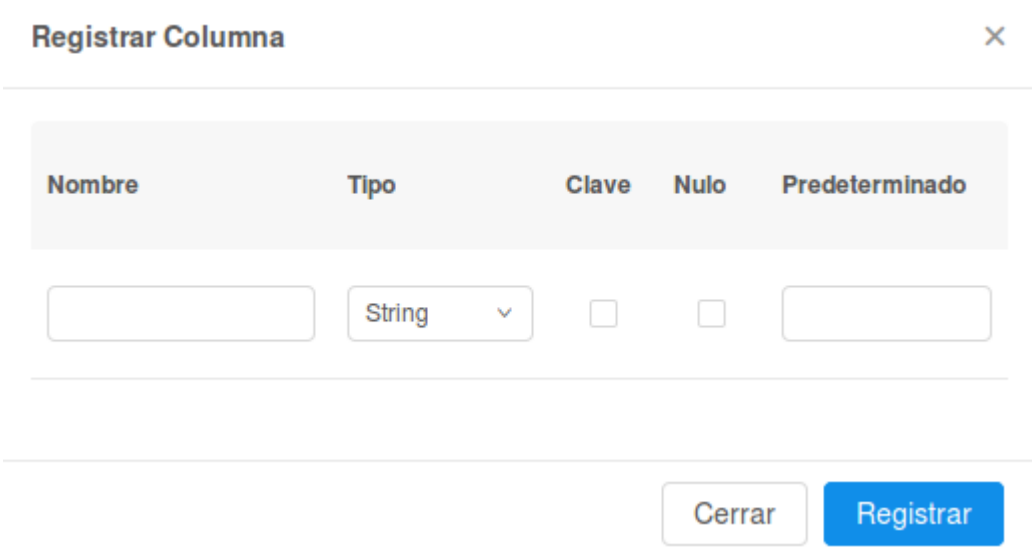


Gráfico 47. **Registrar Columna.**

Finalmente, tenemos la opción perfil (ver Gráfico 48) que permite consultar los datos de usuario a través de una tabla, la cual puede expandir y consultar su apikey activa (ver Gráfico 49)

Esta apikey se puede bloquear bajo el icono , generando una nueva apikey. Por su parte, el único campo editable para la tabla perfil es el nombre del usuario.

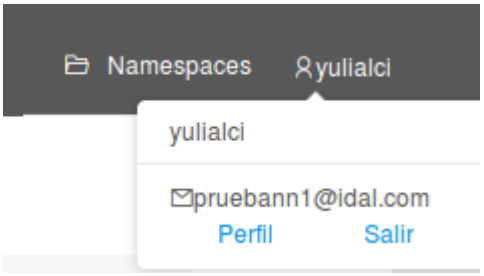


Gráfico 48. **Popover en Header – Perfil Usuario**

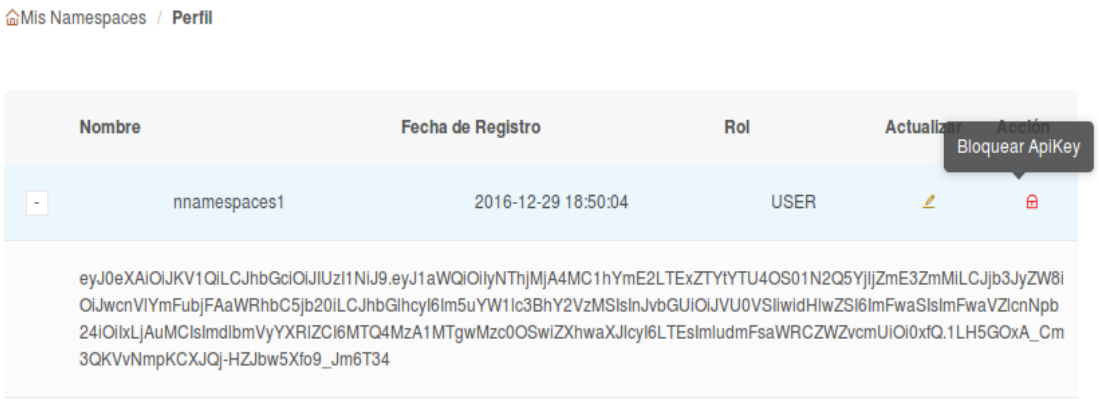


Gráfico 49. **Tabla Perfil.**

Una vez finalizado el diseño de interfaz correspondiente al frontend se procedió hacer el mismo proceso con el backend. En éste el administrador cuenta con todas las funcionalidades del usuario en el frontend. Sin embargo, el listado de namespaces a visualizar le permite nota a que usuario pertenece (ver Gráfico 50). Además, cuenta con una tercera opción en el menú ubicado en el header que le permite ver listado de usuarios (ver Gráfico 51). En dicho listado se utilizan iconos, tales como:

- Bloquear Usuario

- Desbloquear Usuario.
- Consultar Namespaces (Equivalente a Tabla del Gráfico 30).
- Consultar Apikeys (ver Gráfico 52). Muestra Listado de Apikeys
- Eliminar.

Los campos editables para la tabla usuario son Nombre y Rol (ver Gráfico 53).

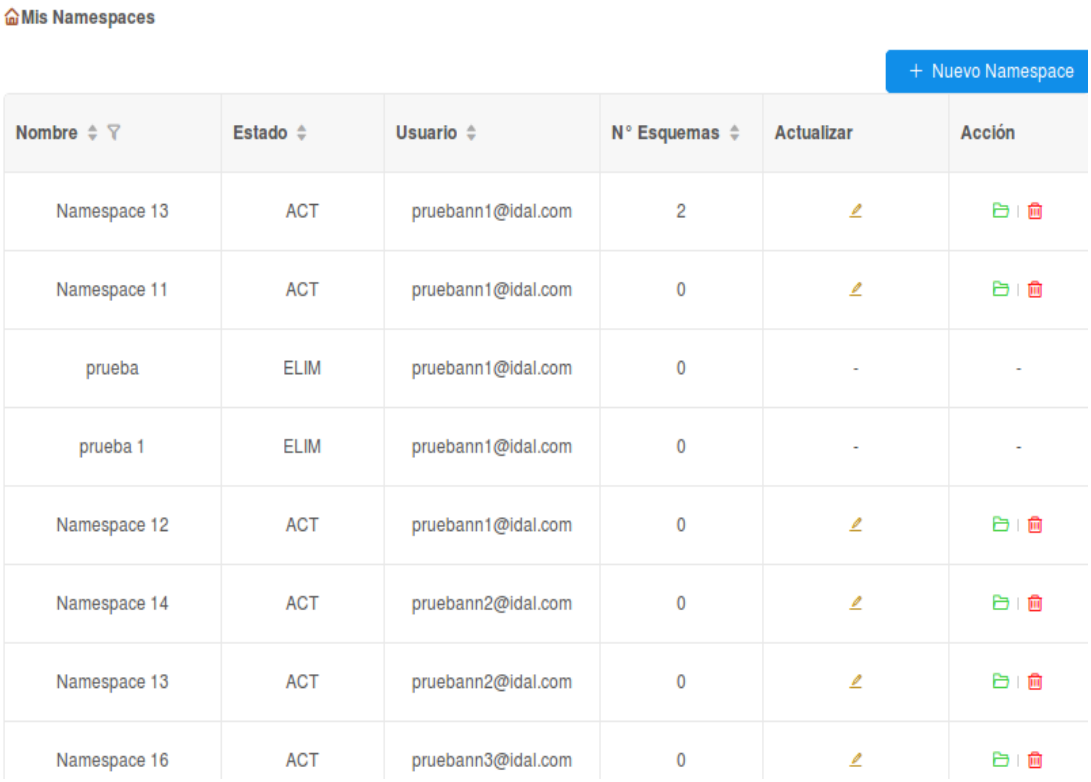


Gráfico 50. **Tabla Namespace para Administrador.**

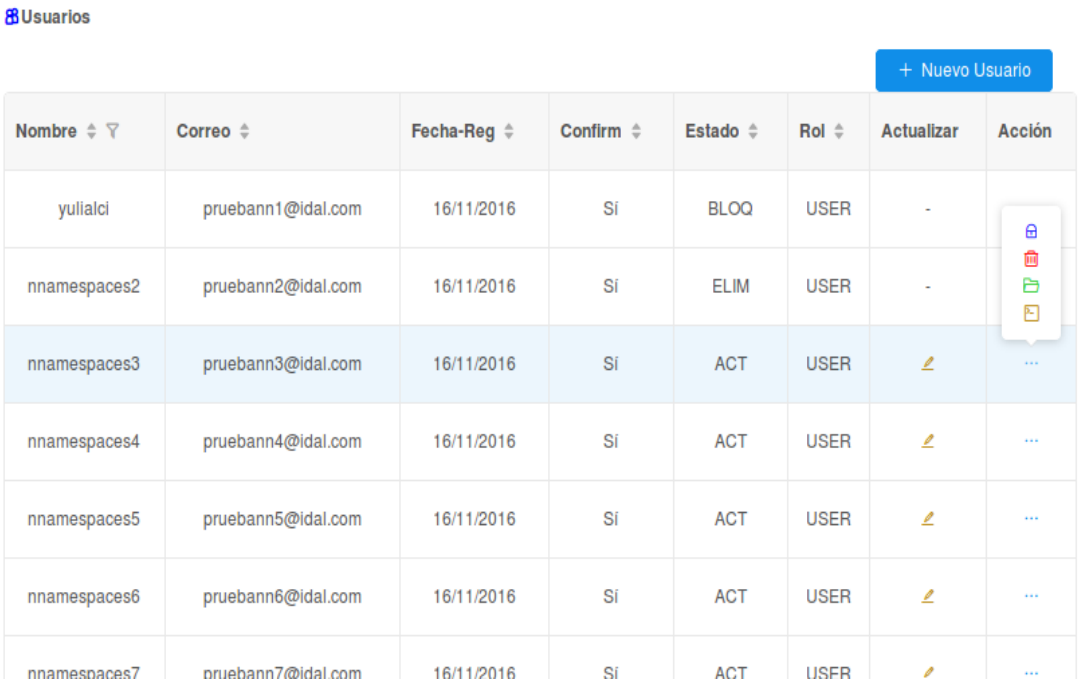


Gráfico 51. **Tabla Usuarios.**

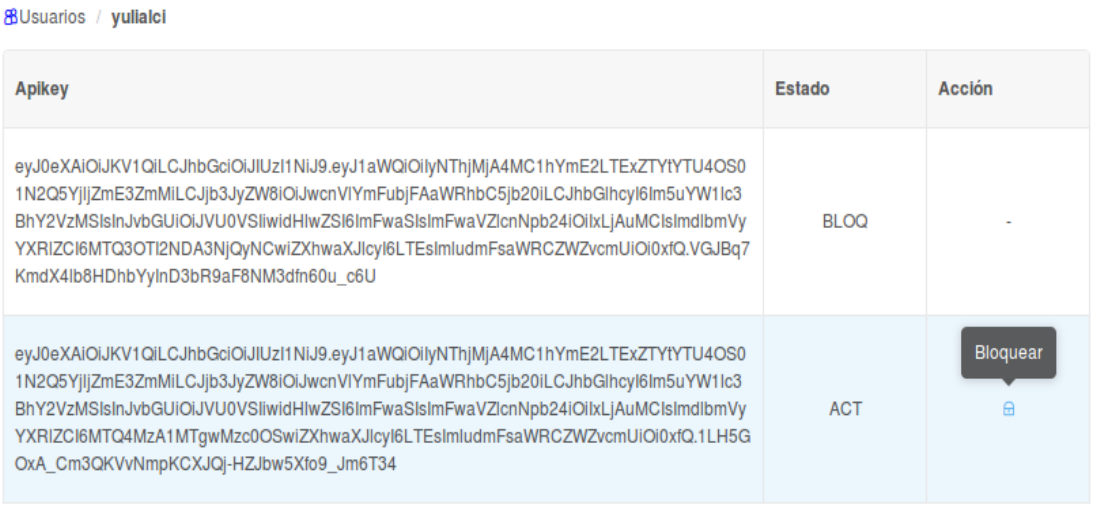


Gráfico 52. **Tabla Apikeys.**

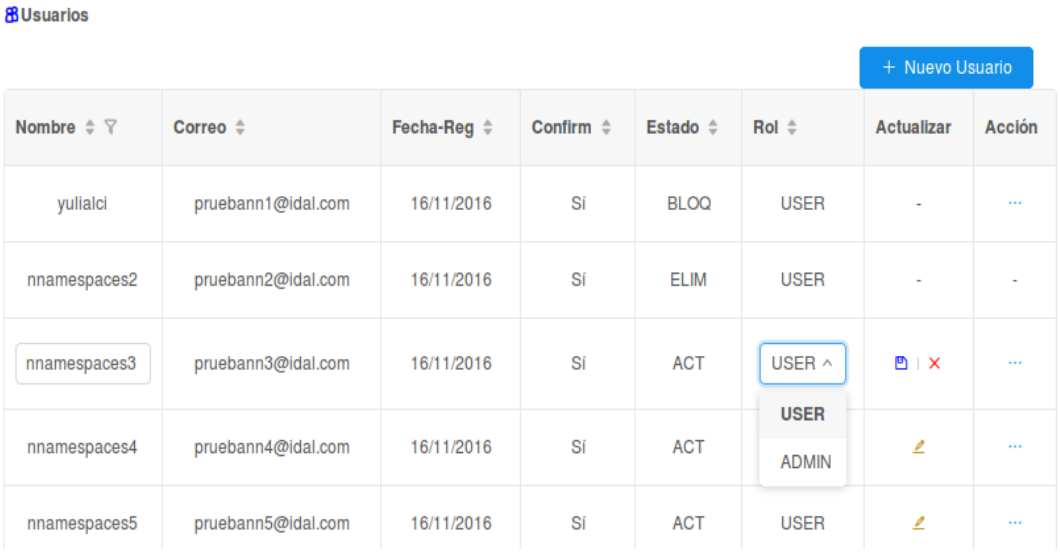


Gráfico 53. **Editar Usuario.**

Fase 3. Post-Juego

En esta fase se realiza la preparación para el lanzamiento de la versión, para la aplicación la versión uno (1), incluyendo la documentación final (realizada en swagger) y pruebas realizadas antes del lanzamiento de la versión. Actualmente se están realizando estas pruebas para pasar a la etapa de producción. Esto excede del ámbito de la investigación que comprende sólo la etapa de desarrollo. Sin embargo, cabe resaltar que Inventame Data Access Layer será comercializado en Chile, pues la empresa se encuentra en dicho país. Particularmente, podrá ser utilizado por programadores para gestionar Big Data utilizando la API correspondiente a IDAL-CORE, la cual podrá conectarse a través de librerías en Node JS a otras plataformas; tales como Inventame Studio. En cuanto a IDAL-WEB-SERVER podrá ser utilizado a nivel empresarial para el manejo de cantidades masivas de datos.

Respecto a las estrategias de mercadeo y comercialización aún están siendo analizadas por la empresa. Sin embargo, cabe destacar que la política para agregar nuevas características, atributos, mejoras, funciones y utilidades será manejada a través de versiones. Además, el costo a pagar por los usuarios será mensual y dependerá de la cantidad de datos y transacciones realizadas. Se estima el costo de cien (100) dólares por cada 10 Gb.

Por otra parte, la descripción mostrada para IDAL-WEB-SERVER en cuanto a interfaz en la fase dos (2) será utilizada posteriormente como parte del Manual de Usuario correspondiente a Inventame Data Access Layer.

CAPÍTULO V

CONCLUSIONES Y RECOMENDACIONES

Conclusiones

Al momento de realizar el idal-core se observaron distintas características que intervinieron en su desarrollo y permitieron que éste se codificara fácilmente, estos se mencionan a continuación:

- Al usar base de datos no relacional y realizar las consultas el tiempo de respuesta en la visualización de los resultados disminuye considerablemente. Esto debido a la escalabilidad horizontal. Esto es resaltante, debido a que el tamaño para la base de datos no relacional es mayor por la cantidad masiva de datos que en ella se almacena. Sin embargo, los tiempos de ejecución son más eficientes, aunque estos no se midieron se observó en la práctica.
- Uno de los principales beneficios de almacenar los datos en un base de datos no relacional es la flexibilidad que proporciona, si se requiere por ejemplo una campo extra en una columna del esquema no es necesario cambiar la estructura de la base de datos, tal como es el caso de la base de datos relacional. Sólo se agrega a la estructura map del campo columns en Cassandra.
- Aunque la base de datos no relacional proporciona muchas ventajas, la base de datos relacional permitió mantener el control sobre la data

almacenada en Cassandra, por lo que ésta también cumple un papel importante. De manera que, cada tipo de base de datos cumple su función.

- También se pudo observar la ventaja que suministra el trabajar con la programación en capas, ésta facilita la organización de la plataforma. De hecho, gracias a esta arquitectura Inventame Data Access Layer soporta la expansión, es decir, pueden agregarse nuevas funcionalidades (nuevas rutas) en el futuro sin que sea necesario cambiar las rutas codificadas.
- Scrum como metodología permitió corroborar que puede construirse software hecho a la medida de las exigencias del usuario final, sin la pretensión innecesaria de desarrollar excesivos diseños cargados de impresiones, muchas veces, abstractas y subjetivas, y que además alargan los procesos de desarrollo de aplicaciones informáticas. Cabe destacar que, se utilizó un grupo de diagramas (de clase y de interacción) pero los mismos no forman parte de SCRUM, éstos se realizaron a petición del Scrum Master.

Todas estas herramientas permitieron que el web-server se diseñara rápidamente, sin tener que realizar un mayor trabajo. Lo que indica que realmente la programación en capas, permite el desarrollo organizado de cualquier plataforma.

Recomendaciones

En vista de lo práctico y organizado que resulta el uso de SCRUM se recomienda utilizar metodologías ágiles y de vanguardia para tener la posibilidad de conocer nuevas formas de emprender la construcción de

aplicaciones informáticas y, tomando en cuenta el buen desarrollo de esta investigación, se recomienda esta metodología para desarrollos futuros.

Por otra parte, al notar los beneficios que aporta el uso de base de datos no relacional se recomienda para futuras investigaciones apoyarse en las nuevas tendencias que facilitan la gestión de información. No limitarse al uso de base de datos relacional. Particularmente en campos como:

- Redes sociales.
- Desarrollo Web.
- Desarrollo Móvil.
- BigData.
- Cloud Computing.

En cuanto al sistema, en vista de su gran utilidad puede ampliarse de diversas maneras, tales como:

- Agregar tipos de datos complejos a los esquemas.
- Agregar permisos de lectura y escritura a los esquemas.
- Realizar join entre esquemas.
- Desarrollar librerías que permitan la conexión de la API a cuentas externas.

REFERENCIAS

Aguilera, P. (2010). *Seguridad Informática.* [Libro en línea]. España: Editex. Disponible: https://books.google.co.ve/books?id=Mgvm3AYIT64C&pgPA69dq=contrase%C3%B1as+hash&hl=es-419&sa=X&ved=0ahUKEjfw9DvAhUw1QKHdiNBLoQ6AEIGDAA#v=onepage&q=contrae%C3%B1s%2hh=false [Consulta: 2017, Marzo 7].

Amazon Web Services (s.f.). *¿Qué es la Cloud informática en la nube?.* [Página web en línea]. Disponible: https://aws.amazon.com/es/what-is-cloud-computing/?nc2= h_l2 _cc [Consulta: 2016, Diciembre 20].

Andreu, F; Pellejero, I. y Lesta, A. (2006). *Fundamentos y aplicaciones de seguridad en redes WLAN: de la teoría a la práctica.* [Libro en línea]. España: Marcombo. Disponible:https://books.google.co.ve/boos?id=k3JuG2D9lMC&pg=PA75&dq=autenticacion+basada+en+token&hl=es419&sa=X&ved=0ahUKEwiUrpadwsXSAhXpjVQKHb1XAIQQ6AEIIzAC#v=onepage&q=autenticacion%20basada%20en%20token&f=false [Consulta: 2017, Marzo 7].

Arias, F. (2006). *El proyecto de investigación: Introducción a la metodología científica*. Caracas. Episteme.

Barragan, A., Forero, A. (2013). Implementación de una base de datos NoSQL para la generación de la Matriz O/D. [Trabajo de Grado]. Disponible: http://repository.Ucatolica.edu.co/bitstream/10983/690/2/IMPLEMENTACION%20DE%20UNA%20BASE%20DE%20DATOS%20NOSQL%20PARA%20LA%20GENERACION%20DE%20LA%20MATRIZ%20OD.pdf [Consulta: 2016, Abril 5].

Campoverde, B. (2012).*Desarrollo de Aplicación Web mediante HTML5 y la base de datos NoSQL MongoDB*. [Trabajo de Grado]. Disponible: http://dspace.uazuay.edu.ec/bitstream/datos/527/1/09451.pdf [Consulta: 2016, Abril 5].

Cardador, A. (2014). *Implantación de aplicaciones web en entornos internet, intranet y extranet.* [Libro en línea]. Málaga: IC. Disponible: https://books.google.co.ve/books?id=Lj91CQAAQBAJ&pg=PT54&lpg=PT54&dq=porque+usar+aplicacione+con+capas+de+datos&source=bl&ots=b7_hMJabLX&sig=05VikkFfI98krJFMg_p7VYmq5ZQ&hl=es419&sa=X&ved=0ahUKEwiNmdyCta3LAhVF0h4KHcTOBwgQ6AEITjAJ#v=onepage&q=porque%20usar%20aplicaciones%20con%20capas%20de%20datos&f=false [Consulta:2016, Marzo 14].

Cardador, A. (2015). *Desarrollo de Aplicaciones web distribuidas*. [Libro en línea]. Málaga: IC. Disponible: https://books.google.co.ve/books?id=CT91CQAAQBAJ&pgPT39&dq=endpoint+que+es&hl=es419&sa=X&ved=0ahUKEwiQgbnHmsXSAhUDTSYKHZA_BNcQ6AEIGDAA#v=onepage&q=endpoint%20que%20es&f=false [Consulta: 2017, Marzo 7].

Cierco, D. (2011). *Cloud Computing: Retos y Oportunidades.* [Libro en línea]. Madrid: Fundación IDEAS. Disponible: https://books.google.co.ve/books?id=_fTJXVjOD90C&pg=PA5&dq=computacion+en+la+nube&hl=es419&sa=X&ved=0ahUKEwj85PLAxcTSAhWCJiYKHXAWDVcQ6AEIKTAD#v=onepage&q=computacion%20en%20la% 20nube&f=false [Consulta: 2017, Marzo 7].

Corrales, M. (2015). *Selección, elaboración, adaptación y utilización de materiales, medios y recursos didácticos en formación profesional para el empleo.* [Libro en línea]. España: Paraninfo. Disponible: https://books.google.co.ve/books?id=8xlODQAAQBAJ&pg=PA142&dq=localizador+de+recursos+uniformes+URL+que+es&hl=es419&sa=X&ved=0ahUKEwj_8IHrs8XSAhWngFQKHWXOBOAQ6AEIHjAB#v=onepage&q=localizador%20de%20recursos%20uniformes%20URL%20que%20es&f=false [Consulta: 2017, Marzo 7].

Daián, G., López, M. y Florencia, M. (2015).*Utilización de NoSQL para resolución de problemas al trabajar con cantidades masivas de datos*. [Trabajo de Grado]. Disponible:http://sedici.unlp.edu.ar/bitstream/handle/10915/45514/Documento_completo.pdf?sequence=1 [Consulta: 2016, Abril 5].

Díaz, W. (2013). *Bases de datos NoSQL: Llegaron para quedarse.* [Página Web en Línea]. Disponible: http://basesdedatosnosql.blogspot.com [Consulta: 2016, Marzo 14].

Dickey, J. (2014). *Write Modern Web Apps with the MEAN Stack: Mongo, Express, AngularJS, and Node.js* [Libro en línea]. Disponible: https://books.google.co.ve/books?id=UEOZBAAAQBAJ&dq=json+web+token+que+es&source=gbs_navlinks_s [Consulta: 2017, Marzo 7].

Fernández, F. (2004). *El documento electrónico en el derecho civil chileno. Análisis de la Ley 19.799.* [Revista en Línea]. Disponible: http://www.scielo.cl/scielo.php?script=sci_arttext&pid=S071800122004000200005 [Consulta: 2017, Marzo 7]

Fernández, G. (2015). *Utilidad de las bases de datos nosql en relación con las técnicas de big data.* [Trabajo de Grado]. Disponible: http://repositorio.utp.edu.co/dspace/bitstream/handle/11059/5379/0057565F363.pdf?sequence=1 [Consulta: 2016, Diciembre 20]

Gilfillan, I. (2011). La Biblia de MYSQL. [Libro en línea]. España: SYBEX. Disponible:http://didepa.uaemex.mx/clases/Manuales/MySql/MySqlLa%20biblia%20de%20 mysql.pdf [Consulta: 2017, Marzo 7].

Hurtado, J. (2000). *Metodología de la investigación holística.* Caracas: SYPAL/YUTE.

Jervis, P. (2006). *La Regulación del Mercado de Datos Personales en Chile.* [Trabajo de Grado]. Disponible: http://repositorio.uchile.cl/bitstream/handle/2250/114258/de-jervis_p.pdf?sequence=1 [Consulta: 2017, Marzo 7].

Joyanes, L. (2016). *Big Data, Análisis de grandes volúmenes de datos en organizaciones.* [Libro en línea]. México: Alfaomega Grupo Editor. Disponible: https://books.google.co.ve/books?id=1GywDAAAQBAJ&pg=PT228&dq=base+de+datos+nosql&hl=es419&sa=X&ved=0ahUKEwi8trDk0MTSAhVE7yYKHUt0D8UQ6AEIMzAF#v=onepage&q=base%20de%20datos%20nosql&f=false [Consulta: 2017, Marzo 7]

Kurzweil, R. (2012). *La Singularidad está cerca. Cuando los humanos transcendamos la biología*. Berlín: Lola Books

Márquez, M. y De La Cruz, M. (2010). *Interfaz de monitoreo para máquinas de hemodiálisis Fresenius*. [Trabajo de Grado]. Disponible: http://132.248.52.100:8080/xmlui/bitstream/handle/132.248.52.100/937/Tesis.pdf?sequence=1 [Consulta: 20 16, Abril 5].

Martínez, D. (2013). *Taller Apache Cassandra. Curso Big Data.* [Documento en Línea]. Disponible: http://eventos.citius.usc.es/bigdata/workshops/Cassandra.pdf [Consulta: 20 16, Diciembre 20].

Mora, S. (2002). *Programación de Aplicaciones Web: Historia, Principios Básicos y Clientes Web*. [Página web en línea] Disponible en: http://rua.ua.es/dspace/bitstream/10045/16995/1/sergio_lujanprogramacion_de_aplicaciones_web.php

Pérez, H. (2010). *Propuesta de análisis y diseño basada en UML y UWE para la migración de arquitectura de software centralizada hacia Internet.* [Trabajo de Grado]. Disponible: http://biblioteca.usac.edu.gt/tesis/08/08_0470_CS.pdf [Consulta: 2016, Abril 5].

Piñeiro, J. (2013). *Bases de datos relacionales y modelado de datos.* [Libro en línea]. España: Paraninfo. Disponible: https://books.google.co.ve/books?id=udFECQAAQBAJ&printsec=frontcover&dq=base+de+datos+relacional&hl=es419&sa=X&ved=0ahUKEwjC8K4zcTSAhXDPiYKHQWADVkQ6AEILzAE#v=onepage&q=base%20de%20datos%20relacional&f=fals e [Consulta: 2017, Marzo 7].

Schwaber, K. y Sutherland, J. (2011). The Scrum Papers: Nut, Bolts, and Origins of an Agile Framework. [Libro en Línea]. Disponible: http://www.hbagency.com/scrum/tlfiles/scrum_inc/documents/ScrumPapers.pdf [Consulta: 2017, Marzo 9].

Schwaber, K. y Sutherland, J. (2016). *The SCRUM Guide.* [Página Web en Línea]. Disponible: http://www.scrumguides.org/scrum-guide.html [Consulta: 2016, Abril 5].

Sierra, M. (2006). *Qué es una base de datos y cuáles son los principales tipos de bases de datos*. [Página Web en Línea]. Disponible: http://www.aprenderaprogramar.com/index.php?option=com_attachments&task=download&id=500 [Consulta: 2016, Marzo 14].

TechTarget (2015). *Base de datos relacional.* [Página Web en Línea]. Disponible: http://searchdatacenter.techtarget.com/es/definicion/Base-de-datos-relacional [Consulta: 2016, Diciembre 20].

Vargas, R. y Maltés, J. (s.f.). *Programación en Capas.* [Página Web en Línea]. Disponible: http://www.di-mare.com/adolfo/cursos/2007-2/pp-3capas.pdf. [Consulta: 2016, Marzo 14].

METADATOS PARA TRABAJOS DE GRADO, TESIS Y ASCENSO:

TÍTULO	Inventame Data Access Layer
SUBTÍTULO	

AUTOR (ES):

APELLIDOS Y NOMBRES	CÓDIGO CULAC / E MAIL
Adrián H., Yulialci.	**CVLAC:** 22.996.846 **E MAIL:** yulialciadrian@gmail.com
	CVLAC: **E MAIL:**
	CVLAC: **E MAIL:**
	CVLAC: **E MAIL:**

PALÁBRAS O FRASES CLAVES:

- Cassandra
- MySQL
- API REST
- SCRUM
- Node JS
- React JS
- Web

METADATOS PARA TRABAJOS DE GRADO, TESIS Y ASCENSO:

ÀREA	SUBÀREA
Ingeniería y Ciencias Aplicadas	Informática

RESUMEN (ABSTRACT):

El propósito de esta investigación fue brindar soporte a las herramientas desarrolladas por Inventame Group LLC, utilizando la programación en capas. En vista del crecimiento en la gestión de los datos, se hizo necesario utilizar base de datos escalable horizontalmente, denominadas NoSql, particularmente Cassandra bajo a una estructura llave- valor. Para el control de los datos almacenados en Cassandra se utilizó base de datos relacional (MySQL) para guardar la estructura de los namespaces, los cuales contienen esquemas y estos a su vez columnas. Siendo los valores de las columnas los almacenados en Cassandra. Para la gestión de los mismos, el núcleo de Invéntame Data Access Layer consiste en una API REST desarrollada en Node JS (lenguaje de programación utilizado por la empresa), documentada al estilo Swagger; la cual permite al usuario gestionar sus datos por medio de endpoints. Además se diseñó una interfaz web en React JS bajo la metodología SCRUM para que los usuarios que no conocen de programación (clientes) gestionen sus datos. La investigación es proyectiva basada en Hurtado (2000), La metodología de desarrollo utilizada fue SCRUM desarrollada por Schwaber y Sutherland en el año 2014.

METADATOS PARA TRABAJOS DE GRADO, TESIS Y ASCENSO:

CONTRIBUIDORES:

APELLIDOS Y NOMBRES	ROL / CÓDIGO CVLAC / E_MAIL				
Ing.:Ugueto, Eyamir	**ROL**	**CA**	**AS**	**TU** X	**JU** X
	CVLAC:	8.730.054			
	E_MAIL	uguetor@gmail.com			
	E_MAIL				
Msc.:Malavé, Braumalis	**ROL**	**CA**	**AS**	**TU**	**JU** X
	CVLAC:	12.673.143			
	E_MAIL	Bmalave1@hotmail.com			
	E_MAIL				
Dra.: Zabala, Suhail	**ROL**	**CA**	**AS**	**TU**	**JU** X
	CVLAC:	12.067.253			
	E_MAIL	SuhailZabala@hotmail.com			
	E_MAIL				

FECHA DE DISCUSIÓN Y APROBACIÓN:

2017	03	13
AÑO	**MES**	**DÍA**

LENGUAJE. SPA

METADATOS PARA TRABAJOS DE GRADO, TESIS Y ASCENSO:

ARCHIVO (S):

NOMBRE DE ARCHIVO	TIPO MIME
Trabajo_de_grado_Adrian_Yulialci	.docx
Trabajo_de_grado_Adrian_Yulialci	.pdf

CARACTERES EN LOS NOMBRES DE LOS ARCHIVOS: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z. a b c d e f g h i j k l m n o p q r s t u v w x y z. 0 1 2 3 4 5 6 7 8 9.

ALCANCE

ESPACIAL: _____ (**OPCIONAL**)

TEMPORAL: _____ (**OPCIONAL**)

TÍTULO O GRADO ASOCIADO CON EL TRABAJO:

Licenciada en Informática

NIVEL ASOCIADO CON EL TRABAJO:

Pregrado

ÁREA DE ESTUDIO:

Informática

INSTITUCIÓN: Universidad de Oriente, Núcleo Nueva Esparta

METADATOS PARA TRABAJOS DE GRADO, TESIS Y ASCENSO:

DERECHOS

Artículo 41 del reglamento de trabajo de pregrado (Vigente a partir del II semestre 2009, Según comunicado CU-034-2009).*"Los Trabajos de Grado son Propiedad exclusiva de la Universidad y sólo podrán ser utilizados para otros fines con el consentimiento del Consejo de Núcleo respectivo, quien lo participará en Consejo Universitario"*

Yulialci Adrián
C.I.: 22.996.846
AUTOR

Ing. Eyamir Ugueto	Msc. Braumalis Malavé	Dra. Suhail Zabala
C.I.: 8.730.054	C.I.:12.673.143	C.I.: 12.067.253
TUTOR	**JURADO**	**JURADO**

POR LA COMISION DE TRABAJO DE GRADO