Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

IMPLEMENTACIÓN DE UN SISTEMA WEB PARA PLANIFICACIÓN DE PROYECTOS INDIVIDUALES

José Manuel De Paz Estrada
Josue Daniel Pirir Morales
Asesorado por la Inga. Floriza Ávila Pesquera de Medinilla

Guatemala, agosto de 2012

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

IMPLEMENTACIÓN DE UN SISTEMA WEB PARA PLANIFICACIÓN DE PROYECTOS INDIVIDUALES

TRABAJO DE GRADUACIÓN

PRESENTADO A JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR

JOSÉ MANUEL DE PAZ ESTRADA
JOSUE DANIEL PIRIR MORALES
ASESORADO POR LA INGA. FLORIZA ÁVILA PESQUERA DE MEDINILLA

AL CONFERÍRSELES EL TÍTULO DE

INGENIERO EN CIENCIAS Y SISTEMAS

GUATEMALA, AGOSTO DE 2012

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANO	Ing. Murphy Olympo Paiz Recinos
VOCAL I	Ing. Alfredo Enrique Beber Aceituno
VOCAL II	Ing. Pedro Antonio Aguilar Polanco
VOCAL III	Ing. Miguel Ángel Dávila Calderón
VOCAL IV	Br. Juan Carlos Molina Jiménez
VOCAL V	Br. Mario Maldonado Muralles
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANO	Ing. Murphy Olympo Paiz Recinos
EXAMINADOR	Ing. Juan Álvaro Díaz Ardavin
EXAMINADOR	Ing. Pedro Pablo Hernández Ramírez
EXAMINADOR	Ing. José Ricardo Morales Prado
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

José Manuel De Paz Estrada

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANO	Ing. Murphy Olympo Paiz Recinos
VOCAL I	Ing. Alfredo Enrique Beber Aceituno
VOCAL II	Ing. Pedro Antonio Aguilar Polanco
VOCAL III	Ing. Miguel Ángel Dávila Calderón
VOCAL IV	Br. Juan Carlos Molina Jiménez
VOCAL V	Br. Mario Maldonado Muralles
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANO	Ing. Murphy Olympo Paiz Recinos
EXAMINADOR	Ing. César Augusto Fernández Cáceres
EXAMINADOR	Ing. Oscar Alejandro Paz Campos
EXAMINADOR	Ing. José Alfredo González Díaz
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

Josue Daniel Pirir Morales

HONORABLE TRIBUNAL EXAMINADOR

En cumplimiento con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presentamos a su consideración nuestro trabajo de graduación titulado:

IMPLEMENTACIÓN DE UN SISTEMA WEB PARA PLANIFICACIÓN DE PROYECTOS INDIVIDUALES

Tema que nos fuera asignado por la Dirección de la Escuela de Ingeniería en Ciencias y Sistemas, con fecha noviembre de 2011.

Jose Manuel De Paz Estrada

Josue Daniel Pirir Morales

Universidad de San Carlos de Guatemala

Facultad de Ingeniería
Escuela de Ciencias y Sistemas

Guatemala, 29 de mayo de 2012

Ingeniero
Carlos Alfredo Azurdia Morales
Coordinador de Privados y Trabajo de Graduación

Respetable Ingeniero Azurdia:

Por este medio le informo como asesor del trabajo de graduación de los estudiantes universitarios de la carrera de Ingeniería en Ciencias y Sistemas, JOSÉ MANUEL DE PAZ ESTRADA, carné 200611316 y JOSUE DANIEL PIRIR MORALES, carné 200611089, que he revisado el trabajo de graduación titulado: "IMPLEMENTACIÓN DE UN SISTEMA WEB PARA LA PLANIFICACIÓN DE PROYECTOS INDIVIDUALES", y a mi criterio el mismo está completo y cumple con los objetivos propuestos para su desarrollo según el protocolo.

Agradeciendo su atención a la presente,

Atentamente,

Floriza Avila
INGENIERA EN CIENCIAS Y SISTEMAS
COL. No. 4333

Inga. Floriza Felipa Avila Pesquera de Medinilla
Asesor de trabajo de graduación
Colegiado: 4333

Universidad San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

Guatemala, 06 de Junio de 2012

Ingeniero
Marlon Antonio Pérez Turk
Director de la Escuela de Ingeniería
En Ciencias y Sistemas

Respetable Ingeniero Pérez:

Por este medio hago de su conocimiento que he revisado el trabajo de graduación de los estudiantes **JOSÉ MANUEL DE PAZ ESTRADA** carné **2006-11316** y **JOSUE DANIEL PIRIR MORALES** carné **2006-11089**, titulado: **"IMPLEMENTACIÓN DE UN SISTEMA WEB PARA PLANIFICACIÓN DE PROYECTOS INDIVIDUALES"**, y a mi criterio el mismo cumple con los objetivos propuestos para su desarrollo, según el protocolo.

Al agradecer su atención a la presente, aprovecho la oportunidad para suscribirme,

Atentamente,

Ing. Carlos Alfredo Azurdia
Coordinador de Privados
y Revisión de Trabajos de Graduación

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
INGENIERIA EN CIENCIAS Y SISTEMAS
COORDINACION

ESCUELA DE CIENCIAS Y SISTEMAS

UNIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERÍA
ESCUELA DE CIENCIAS Y SISTEMAS
TEL: 24767644

El Director de la Escuela de Ingeniería en Ciencias y Sistemas de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer el dictamen del asesor con el visto bueno del revisor y del Licenciado en Letras, del trabajo de graduación titulado ***"IMPLEMENTACIÓN DE UN SISTEMA WEB PARA PLANIFICACIÓN DE PROYECTOS INDIVIDUALES,*** *presentado por los estudiantes JOSÉ MANUEL DE PAZ ESTRADA Y JOSUE DANIEL PIRIR MORALES, aprueba el presente trabajo y solicita la autorización del mismo.*

"ID Y ENSEÑAD A TODOS"

Ing. Marlon Antonio Pérez Turk
Director, Escuela de Ingeniería en Ciencias y Sistemas

Guatemala, 09 de agosto 2012

Universidad de San Carlos
de Guatemala

Facultad de Ingeniería
Decanato

Ref.DTG.394.2012

El Decano de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer la aprobación por parte del Director de la Escuela de Ingeniería en Ciencias y Sistemas, al trabajo de graduación titulado: **IMPLEMENTACIÓN DE UN SISTEMA WEB PARA PLANIFICACIÓN DE PROYECTOS INDIVIDUALES**, presentado por los estudiantes universitarios: **José Manuel De Paz Estrada y Josue Daniel Pirir Morales,** procede a la autorización para la impresión del mismo.

IMPRÍMASE.

Ing. Murphy Olympo Paiz Recinos
DECANO

Guatemala, agosto de 2012

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DECANO
FACULTAD DE INGENIERIA

/cc

ACTO QUE DEDICO A:

Dios	Por darme las fuerzas para seguir adelante y nunca darme por vencido.
Mis padres	Ana Lucrecia Estrada y Eddy De Paz, por creer siempre en mí, apoyarme en cualquier decisión que tome, enseñarme los principios y valores que son fundamentales para alcanzar mis metas, además de su amor y cariño que me brindan día a día.
Mis hermanos	Ana y Héctor De Paz, por darme su apoyo en todo momento y hacer lo que estuviera a su alcance para ayudarme durante mis estudios.
Mis abuelos	Leticia Mérida, Marta Veliz y René Estrada, por sus enseñanzas, apoyo y cariño.
Mis tíos y tías	Ronald De Paz y en especial a Genoveva Estrada por su gran apoyo durante los primeros años que pasé en la universidad.

José Manuel De Paz Estrada

AGRADECIMIENTOS A:

Mis amigos y compañeros	Que me apoyaron durante todos mis estudios y que de alguna forma ayudaron a que siguiera adelante. En especial a mi hermano Raúl Ávila, Elder Prado, Josue Pirir, Anabela Díaz, Emilio Méndez, Alicia Ruano, Pablo Ramírez, Eduardo Quetzales, Gabriela Díaz, Michael Colindres, Honard Bravo, Anibal Chicojay, Balvino López, Jhon Granados, Ivan Chalí, David Freddy González, Jorge Maldonado, Juan Cano y Oliver Ramos. Junto con otros amigos de la infancia y demás compañeros.
Mi asesora	Ingeniera Floriza Ávila, por su guía y tiempo para la elaboración de este trabajo, además de sus consejos y apoyo durante el tiempo que fui auxiliar en su curso.
Mis catedráticos	Por todas sus enseñanzas que son el cimiento del conocimiento de muchos profesionales.
A la Universidad de San Carlos de Guatemala	Por ser la casa de estudios donde logré alcanzar mi sueño de ser un profesional de bien para mi país, y enseñarme una nueva forma de vida.

José Manuel De Paz Estrada

ACTO QUE DEDICO A:

Dios	Por todas las bondades y dones concedidos, así como la voluntad y fe en los momentos difíciles.
Mis padres	Florencio Pirir y Jesús Morales, por brindarme todo su amor y cariño, enseñarme los principios y valores que fundamentan mi vida; juntos lo hemos logrado.
Mi hermano	Luis Alberto, por darme su apoyo en todo momento; pronto lograremos alcanzar esa meta.
Mis tías	María Mercedes y Verónica, por brindarme todo su apoyo y cariño incondicionalmente.
Mis tíos	Edgar Rolando y Basilio, por ser como hijos de mis abuelos.
Mis abuelos	Carlota, Paulina y en especial a Secundino Morales, quien fuera como mi segundo papá; sé que desde allá arriba cuidarás siempre de mí.

Josue Daniel Pirir Morales

AGRADECIMIENTOS A:

Mis amigos y compañeros	Que de alguna u otra manera compartimos este largo viaje que aún no termina. En especial a Elder Prado, José De Paz, César Montes, Anabela Díaz, Iván Chalí, Gabriela Díaz, Daniel Paiz, Emilio Méndez, Michael Colindres, Honard Bravo, Eduardo Quetzales, Marlin Arévalo, Juan Roldán, Oliver Ramos, Balvino López, Silvia Rossana, Nathalie Morán, María José Palacios, Juan Pablo Barrera, Raúl Sánchez, Jorge López, Freddy González, Pedro Quemé, Juan Luis Cano y Jorge Maldonado.
Mi asesora	Por apoyarme en la realización de este trabajo de graduación, por guiarme y motivarme para la realización del mismo.
Mis catedráticos	Por todas sus enseñanzas que son el cimiento del conocimiento de muchos profesionales y el mío. En especial a la Inga. Virginia Tala tutora de este trabajo de Graduación y al Ing. Marlon Orellana y Javier Gramajo, por sus consejos.
A la Universidad de San Carlos de Guatemala	Por abrirle las puertas y formarme como profesional, enseñándome que el aprendizaje no termina en sus aulas.

Josue Daniel Pirir Morales

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

FIGURAS

TABLAS

GLOSARIO

AJAX	Asynchronous JavaScript And XML (JavaScript asíncrono y XML).
API	Application Programming Interface (Interfaz de programación de aplicaciones), es una biblioteca que permite integrar un conjunto de funciones y procedimientos a un nuevo *software*.
Asíncrono	Un proceso que no tiene lugar en total correspondencia temporal con otro que le precede.
Browser	Es un navegador de Internet; un intérprete de los lenguajes utilizados, aceptados por W3C.
Cliente	Es la capa donde el usuario accede a la ejecución de operación en una arquitectura cliente-servidor.
CPM	Método para el cálculo de la ruta crítica.
CSS	Cascading Style Sheets (hojas de estilo en cascada).
Datastore	Relativo al almacén de datos, nombre que se le da al almacén de datos de Google.

Diagrama de Gantt	Es una herramienta visual que tiene como objetivo mostrar el tiempo que tomará realizar una tarea a lo largo del tiempo.
DOM	Document Object Model (Modelo de Objetos del Documento).
EDT	Estructura de desglose de trabajo.
Feedback	Retroalimentación.
Framework	Es una estructura conceptual y tecnológica de soporte, definida normalmente con artefactos o módulos de *software* concretos, con base en la cual otro proyecto de *software* puede ser organizado y desarrollado.
GAE	Google App Engine. Servicio de *cloud computing* de Google.
Hosted	Medio donde se encuentra alojada una aplicación Web.
HTML	HyperText Markup Language (Lenguaje de Marcado de Hipertexto).
HTTP	HyperText Transfer Protocol (Protocolo de transferencia de hipertexto).

I/O	Input/Output, acrónimo de dispositivos de entrada y salida.
IDE	Integrated Development Environment (Entorno de desarrollo integrado), permite desarrollar de una manera más clara las aplicaciones.
IEEE	Institute of Electrical and Electronics Engineers (Instituto de Ingenieros Eléctricos y Electrónicos).
Internet	Es un conjunto descentralizado de redes de comunicación, interconectados a través de protocolos TCP/IP.
Java	Lenguaje de programación orientado a objetos creado por Sun Microsystems.
Javascript	Lenguaje de programación interpretado, que se ejecuta del lado del cliente en una aplicación Web.
JDO	Java Data Objects, especificación de almacenamiento en formato de objetos de Java.
JPA	Java Persistence API, API de persistencia de Java.
JSON	JavaScript Object Notation.
PERT	Program Evaluation and Review Technique, (Técnica de revisión y evaluación de programas).

Plugin	Es un complemento de una aplicación que permite agregar nuevas funcionalidades a la misma.
Request	Término del inglés que significa solicitud, utilizado en informática para representar una solicitud de datos.
Response	Término del inglés, respuesta.
RIA	Aplicaciones de Internet enriquecidas.
Script	Archivo de órdenes o de procesamiento por lotes; es un programa usualmente simple, que por lo regular se almacena en un archivo de texto plano.
Servidor	Es la capa donde se ejecutan las operaciones solicitadas por el cliente y encapsula la lógica del negocio.
Sistema	Conjunto de partes relacionadas en un medio, que interactúan entre sí para alcanzar un objetivo en común.
W3C	World Wide Web Consortium, consorcio internacional que dirige la forma correcta para la WWW.
Widget	Pequeña aplicación o programa reutilizable, que debe ser ejecutado por un motor especifico.

WWW	World Wide Web, sistema de distribución basado en hipertexto accesible a través de Internet.
XHTML	Extensible HyperText Markup Language (Lenguaje de marcas de hipertexto extensible).
XML	Extensible Markup Language (Lenguaje de marcas extensible).

RESUMEN

El proyecto de tesis, Implementación de un sistema Web para planificación de proyectos individuales, tiene como fin principal, presentar una solución que se adapte a las necesidades de los usuarios que se dirigen hacia una Web más semántica.

Debido a que el uso de las tecnologías de la información tiene un alto impacto en el rendimiento de la ejecución de tareas, se busca crear un sistema con el cual el usuario pueda llevar una correcta administración de sus actividades.

Este sistema se desarrolla aplicando nuevas tecnologías y herramientas para el desarrollo Web, entre la cuales se encuentran: las relacionadas a las aplicaciones de Internet enriquecías (RIA por sus siglas en inglés), como AJAX, JavaScript, HTML 5, etc.

Las tecnologías RIA se combinan perfectamente con las implementaciones provistas por Google, entre las que se encuentran Google Web Toolkit (GWT) y Google App Engine (GAE), además se hará uso de una de las metodologías de desarrollo ágil, la que será eXtreme Programming de la cual se tendrá como resultado final la descripción del sistema a través de diagramas UML.

OBJETIVOS

General

Desarrollar un sistema Web minimalista para la planificación de proyectos individuales, utilizando tecnologías propias de RIA, como herramientas y lenguajes de programación libres para documentar el desarrollo, implementación y publicación final, para que el sistema pueda ser utilizado por cualquier usuario a través de Internet.

Específicos

1. Desarrollar un sistema Web que aproveche las tecnologías AJAX, junto con las tecnologías de desarrollo ofrecidas por Google.

2. Desarrollar un sistema intuitivo para el uso de usuarios inexpertos tanto en planificación de proyectos como en la utilización de aplicaciones Web.

3. Publicar un sistema que permita al usuario gestionar varios proyectos a la vez, utilizando la misma interfaz.

4. Implementar la usabilidad al permitir ejecutar acciones, minimizando la cantidad de movimientos con el *mouse* a tres.

5. Implementar la característica para la exportación del plan de proyecto en formato PDF.

INTRODUCCIÓN

La Internet mantiene un ciclo de evolución constante hace apenas un lustro atrás se hablaba de la Web 2.0 como una evolución de la Web 1.0, esta se diferenciaba por ser una red encargada del manejo y transferencia de información, hasta que la Web 2.0 se consolidó como una red erigida por y para los usuarios, almacenando una gran cantidad de información, que se comparte a diario por medio de blogs, redes sociales, páginas personales y otros medios digitales.

El futuro de la Internet se encuentra encausado hacia una red semántica, es decir inteligente, donde se utiliza la red no solamente como un medio para intercambiar información, sino que también, ahora cuenta con herramientas y tecnologías de desarrollo que les permite a los programadores acercarse al desarrollo de sistemas más humanos.

Cada día los requerimientos de los usuarios aumentan, porque existe la necesidad de encontrar soluciones precisas, por lo tanto es necesario que las aplicaciones cumplan con; un proceso iterativo el cual tenga lugar en intervalos de tiempo cada vez más cortos.

En el siguiente trabajo de tesis: Implementación de un sistema Web para la planificación de proyectos individuales, se desarrollará una solución de *software*, que permita a los usuarios organizar sus proyectos, de tal manera que podrán optimizar el recurso tiempo para ayudarlos a alcanzar mejores resultados.

El sistema será implementado en una plataforma Web, esta brindará entre otras cualidades, ubicuidad, disponibilidad, y usabilidad lo cual le permitirá contar con un sistema amigable e inteligente que le ayudará durante su proceso de aprendizaje en la gestión de proyectos, apoyado de tecnologías que forman parte de RIA; dichas tecnologías son explicadas y documentadas a lo largo del presente trabajo.

En la primera parte, se describirán los fundamentos de la plantificación de proyectos y los problemas que el sistema podrá solucionar; en la segunda parte se encuentra el marco conceptual de las tecnologías a aplicar durante el desarrollo; en la tercera parte se encontrará al conjunto de artefactos de salida, propios de las fases de análisis y diseño que ayudarán a la explicación del desarrollo de la aplicación; mientras que el capítulo final hará descripción de los resultados, el análisis y verificación del cumplimiento del alcance y los objetivos, teniendo como base del mismo un sistema funcional.

1. TÉCNICAS DE PLANIFICACIÓN DE PROYECTOS

1.1. Dirección de proyectos

Un proyecto se define como, un esfuerzo temporal, que es llevado a cabo para la creación de un producto, el cual debe de ser único. Se dice que es un esfuerzo, porque se requiere de la utilización de recursos para su desarrollo y es temporal porque tiene un principio y un final, en el cual dan lugar a un conjunto de actividades que son administrables.

En la dirección de proyectos es necesario reunir un conjunto de: cualidades, habilidades, herramientas y técnicas en el desarrollo de actividades, para cumplir con los objetivos de los mismos, los cuales son alcanzados mediante la aplicación e integración de los procesos de dirección de proyectos; estos son: la planificación, ejecución, seguimiento y control, cierre. Texto de la Guía de los fundamentos para la dirección de proyectos del Project Management Institute.

En la dirección de un proyecto, existe un responsable de llevar el mismo hasta alcanzar sus expectativas; este es el director de proyectos, quien debe identificar los requerimientos, establecer objetivos y límites del mismo, así también debe de determinar cómo administrar adecuadamente los recursos a su disposición, tomando en cuenta que el proyecto estará sujeto a cambios, los cuales en la mayoría de los casos son dictados por el cliente. En el caso del sistema a implementar, el director de proyecto será el mismo ejecutor, ya que es el único responsable en la administración de proyectos individuales.

El director de proyecto a menudo debe de luchar contra tres factores intrínsecos durante el desarrollo del mismo, los cuales son: alcance, tiempo y costo. Para que un proyecto sea catalogado como de alta calidad este debe haber cumplido a cabalidad el alcance, en el tiempo justo y con los costos dentro del presupuesto. Además, se debe tomar en cuenta que cada una de las variaciones en cualquiera de estos factores, afectará a otros.

1.2. Ciclo de vida de un proyecto

- Un proyecto se caracteriza por ser temporal, es decir que tiene un inicio y un final. El ciclo de vida de un proyecto se puede definir como un conjunto de fases en los cuales se divide el mismo, cada una con un enlace establecido que servirá para su integración; regularmente se divide en una fase inicial donde se determina el alcance del proyecto; en las fases intermedias se detalla el plan y el avance, se realizan las validaciones y verificaciones, mientras que en la fase final se aprueba y entrega el mismo.

- Uno de los principales motivos por lo cual un proyecto es separado en fases es para llevar un control del avance total del proyecto y determinar en qué momento puede ser modificado o abandonado, debido a que no cumple con los requerimientos o este está consumiendo demasiados recursos, por lo cual se hace insostenible.

Durante el ciclo de vida de un proyecto, las transiciones entre fases están definidas por un conjunto de entregables técnicos. En cada transferencia estos se revisan, para verificar si se encuentran completos e íntegros; asimismo, se realizan las respectivas pruebas, verificaciones y validaciones necesarias para dar pie a la siguiente fase.

Es importante mencionar que no siempre es necesario terminar una fase para iniciar otra; regularmente existen fases que necesitan ser ejecutadas en paralelo para luego encontrar un punto de intersección común. A esta técnica de superposición de fases se le conoce como ejecución rápida.

Debido a que no existe una única manera de definir cómo se desarrollará el ciclo de vida, este se puede definir al responder las siguientes interrogantes:

- ¿Qué trabajo técnico se debe realizar en cada fase?
- ¿Cuándo se deben generar los productos entregables en cada fase y cómo se revisa, verifica y valida cada producto entregable?
- ¿Quién está involucrado en cada fase?
- ¿Cómo controlar y aprobar cada fase?

Las descripciones van de lo general a las detalladas, en la cuales se encuentran formularios, diagramas y listas de control. Sin embargo entre las características comunes del ciclo de vida se puede encontrar:

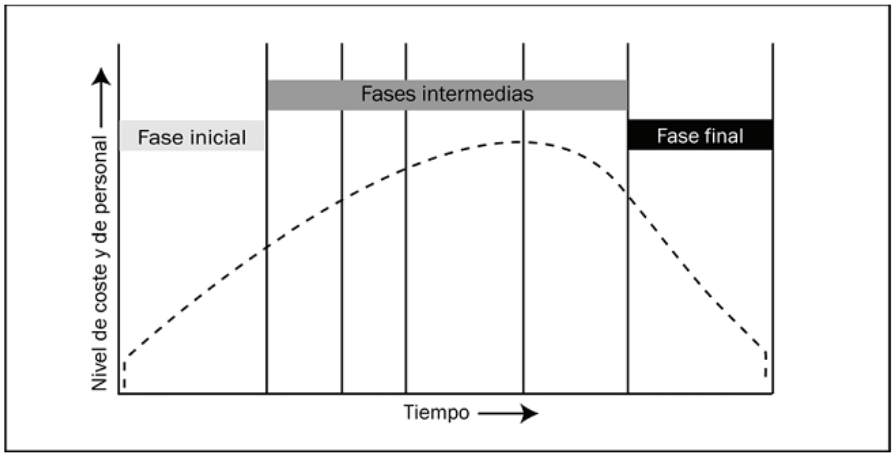
- Fases secuenciales, definidas por la transferencia de entregables o componentes técnicos.

- La cantidad de recursos utilizados en el comienzo es bajo, alcanza su nivel máximo en fases intermedias y cae rápidamente cuando el proyecto está por concluir.

- El nivel de incertidumbre se encuentra en su punto más alto, al inicio del proyecto, pero decae a medida que se va avanzando en el mismo.

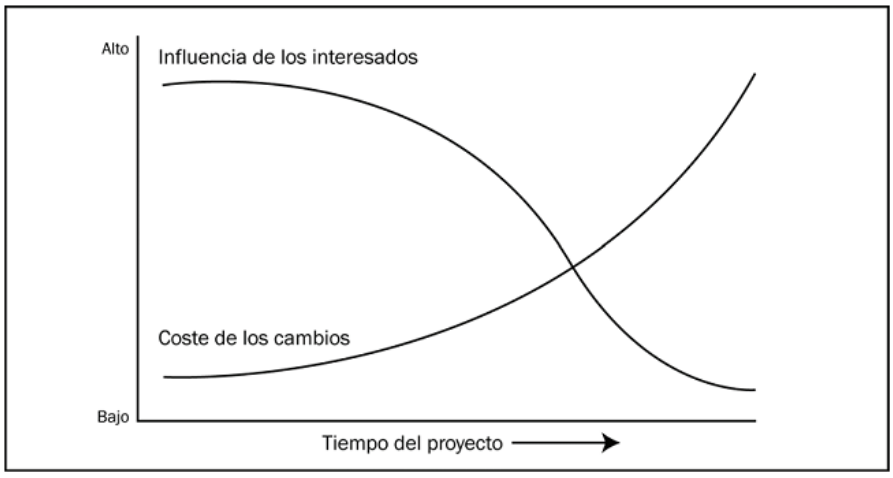
- Los costos en los cambios se encuentran en su nivel más bajo al inicio del proyecto y se incrementan a medida que va avanzando.

Figura 1. **Gráfico recursos versus tiempo**



Fuente: PMI. Guía del PMBOK. P.21.

Figura 2. **Gráfico costos versus tiempo**



Fuente: PMI. Guía del PMBOK. P.21.

El número de fases de un proyecto, es un tema que se deja a criterio del director del mismo, el cual define la cantidad de acuerdo con sus necesidades. Por ejemplo si se trata de un proyecto de desarrollo, la cantidad de fases en el ciclo de vida depende de la metodología de desarrollo seleccionada.

1.3. Planificación de proyectos

- Un proyecto, es una técnica de planificación que le permitirá organizar actividades; un proyecto se convierte así como un medio para elaborar un plan estratégico en búsqueda de un objetivo. Normalmente se planea un proyecto por las siguientes razones:

- Una demanda del mercado (por ejemplo, una compañía solicita la instalación de un nuevo servidor debido a que la cantidad de solicitudes por parte de los clientes ha sobrepasado el límite).

- Una necesidad de la organización (por ejemplo, la organización necesita centralizar su información por lo cual necesita implementar una solución de inteligencia de negocios).

- Una solicitud de un cliente (es decir, una empresa de Internet, amplía su red de cobertura debido a una zona demográfica en crecimiento).

- Un avance tecnológico (es decir, una compañía de fármacos autoriza el proceso de investigación de una nueva droga para la ansiedad).

- Un requisito legal (es decir, una empresa bancaria le solicita mejorar el nivel de seguridad en su portal Web, pues es requisito según la nueva legislación del *cibercrimen*).

1.4. Programación de tareas

Una tarea se puede definir como una actividad que debe de realizarse en un tiempo determinado; del mismo modo se puede definir a la programación como al conjunto de procedimientos y técnicas estructuradas, con las cuales, se realizarán de forma ordenada un conjunto de actividades.

Al definir estos conceptos, el paso siguiente se refiere a la programación de tareas como la acción de listar un conjunto de actividades, definiendo los procedimientos y técnicas a utilizar, para que pudiesen realizarse en un tiempo determinado.

El sistema a implementar, le permitirá al usuario programar tareas de forma tal que se pueda llevar un correcto seguimiento y control de las mismas. Es importante enfatizar que la planificación de tareas, está alineada con el alcance del proyecto, es decir que al concluir con el desarrollo del proyecto, el alcance debe de ser medible.

La escritura de este documento, puede convertirse en un buen ejemplo para englobar cada uno de los elementos que lo conforman. En primer lugar se tiene una tarea a realizar que será la elaboración de la tesis, la cual está enfocada en el área de desarrollo.

Antes de comenzar con la misma se debieron de contestar varias preguntas, aparte de las que se plantearon durante la elección del tema, entre ellas se encuentran:

- ¿Conoce la fecha límite para la entrega?

- ¿Cuáles son las tareas que deben de ejecutarse?
- ¿Cómo se repartirán las tareas?
- ¿Cómo se separarán los capítulos y contenidos del documento?
- ¿De cuánto tiempo dispone cada autor?
- ¿Cada uno de los autores realiza las tareas con igual eficiencia?
- ¿Los autores realizan las tareas utilizando un intervalo de tiempo intermedio y no siempre están dedicados a la actividad?
- ¿Qué modificaciones y con qué tiempo se cuenta para ser presentado al asesor, realizar las correcciones y presentársela al tutor?

Si se detiene a analizar, quizá desde su punto de vista, las respuestas a algunas de estas interrogantes son afectadas por variables exógenas, como por ejemplo la fecha límite de entrega, las cuales están afuera del alcance de cada uno de los involucrados en el desarrollo de la misma.

A continuación se indican algunas de las tareas que deberán ser realizadas:

- Definición del tema
- Selección de teoría
- Elaboración de matriz de contenidos
- Elaboración de protocolo

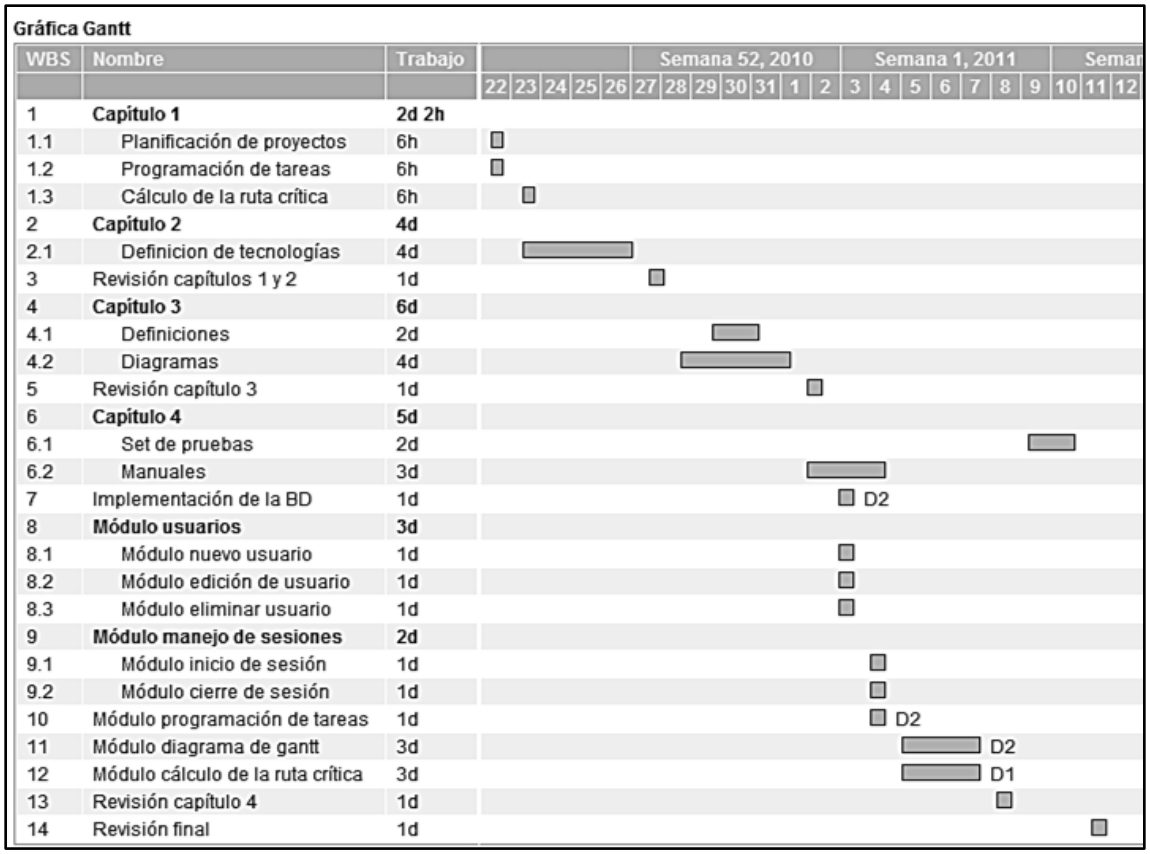
- Escritura de capítulos iniciales
- Definición de tecnologías
- Elaboración de módulos
- Escritura de capítulos finales
- Publicación de la solución
- Revisión final

Se pueden programar dichas tareas apoyándose con el uso de un diagrama de Gantt; en la figura 3, se puede observar el “borrador” de la programación de tareas de este trabajo de tesis.

En el mismo podrá observarse que existen tareas que dependen de la terminación de otras y no pueden ser elaboradas en paralelo; una vez realizada la distribución de tareas, se observa que estas se pueden separar, de acuerdo con la experiencia.

Luego se determinó que se podían realizar tareas en paralelo que en un principio se creyó que no eran posibles; además, no se tomaron en cuenta los recesos como los asuetos, debido a las fiestas de fin de año, dichas situaciones solo pueden ser previstas, a medida que auméntala experiencia, la cual solamente se gana con el paso de los años y el involucramiento en ejecución y administración de proyectos.

Figura 3. **Diagrama de Gantt, desarrollo de tesis 2010-2011**



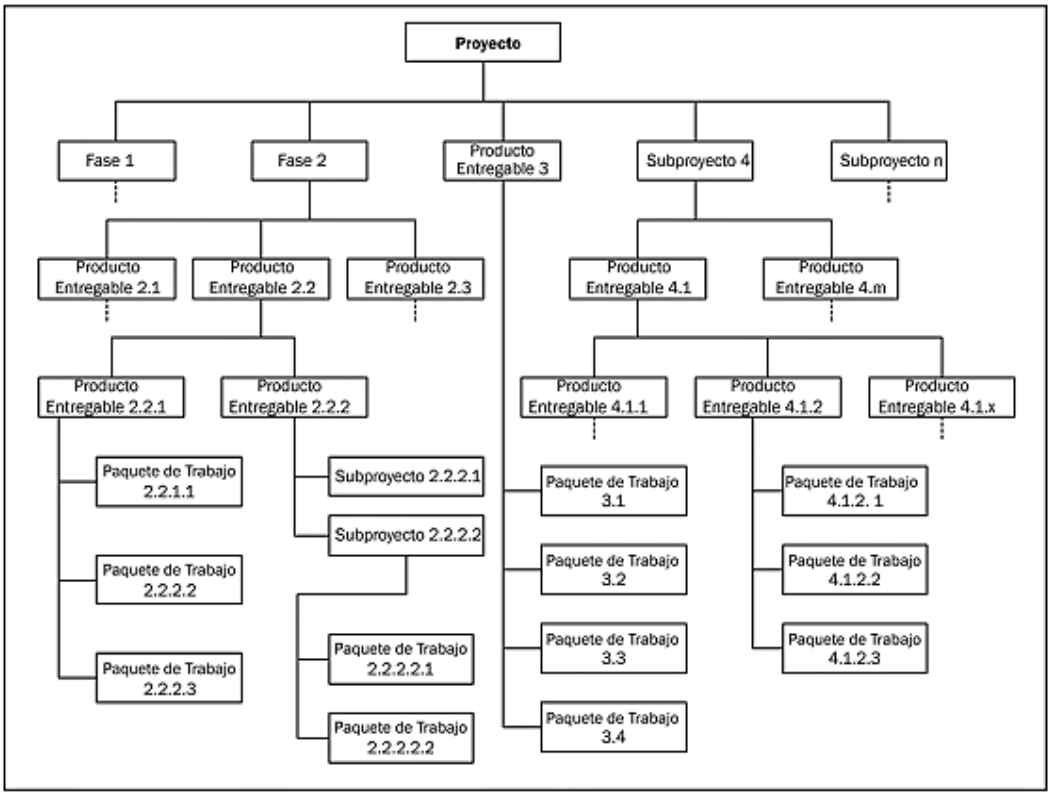
Fuente: elaboración propia.

1.4.1. EDT

La estructura de desglose de trabajo (EDT) o WBS (Work Breakdown Structure por sus siglas en inglés), es una descomposición jerárquica, orientada al producto entregable del trabajo que será ejecutado por el equipo del proyecto, para lograr los objetivos del mismo y crear los productos entregables requeridos. La EDT organiza y define el alcance total del proyecto.

La EDT subdivide el trabajo del proyecto en porciones de trabajo más pequeñas y fáciles de manejar, donde cada nivel descendente representa una definición cada vez más detallada del trabajo del proyecto. “El trabajo planificado comprendido dentro de los componentes de la EDT del nivel más bajo, denominados paquetes de trabajo, puede programarse, supervisarse, controlarse y estimarse sus costes”[1]

Figura 4. **Estructura genérica de EDT**



Fuente: PMI. Guía del PMBOK. P.21.

[1]Project Management Institute. *Guía de los fundamentos para la dirección de proyectos*. P. 112.

2. TECNOLOGÍAS DE DESARROLLO

2.1. RIA

Rich Internet Application(RIA), es un conjunto de tecnologías utilizadas para la implementación de aplicaciones Web, entre las cuales se pueden encontrar: las tecnologías AJAX, Flex de Adobe Flash, GWT de Google, lenguajes basados en *scripts*, los cuales brindan a las aplicaciones Web características que anteriormente solo se encontraban en aplicaciones de escritorio, como validaciones automáticas, *drag and drop*, *cursor busy*, *in page edit*, etc. Texto de Professional Rich Internet Applications de Dana Moore.

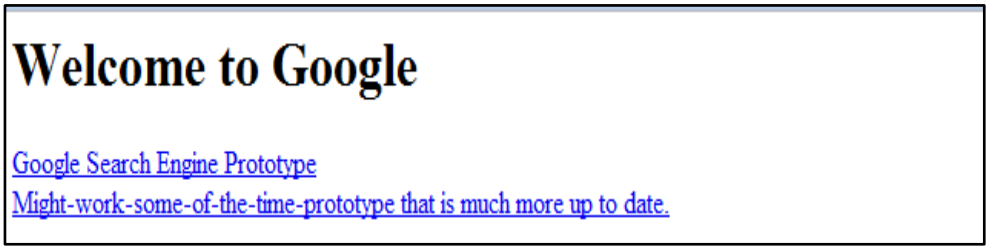
El sistema a implementar en este trabajo de tesis se encuentra basado en RIA, por lo cual la misma gana en facilidad de uso, *feedback* y mayor interacción con el usuario, colocando la información en un eje central, obteniendo *richness* (riqueza), características propias de las nuevas tendencias en los sistemas de información y aplicaciones en Internet.

RIA nace como una necesidad de los usuarios y desarrolladores que buscaban una nueva generación de plataformas y aplicaciones de Internet, capaz de darle extensibilidad y capacidad al permitir ser reconstruida en diferentes formas, brindándole a las aplicaciones *look and feel.*

Desde ese entonces, cada página o aplicación Web debía de contar con un “estilo”, ya que los cambios son notoriamente visibles; solo basta con observar una simple página escrita en HTML y una que contiene una hoja de estilos dinámica CSS.

En las figuras 5 y 6, puede observar un ejemplo sobre "plástica"; concepto que se le atribuye a la utilización de tecnologías RIA por su dinamismo y su característica de centrarse en la apariencia. Texto de Professional Rich Internet Applications de Dana More.

Figura 5. **Primera página de inicio de Google 1997**



Fuente: http://web.archive.org/web/19981111184551/http://google.com/. Consulta: 10 de enero de 2011.

Figura 6. **Página de inicio Google en 1998, todo en un solo HTML**

Fuente: http://web.archive.org/web/19981202230410/http://www.google.com/. Consulta: 10 de enero de 2011.

Figura 7. **Página de inicio de Google 2011, con AJAX**

Fuente: http://www.google.com.gt. Consulta: 11 de enero de 2011.

En las figuras anteriores se puede observar la evolución de la página Web, del buscador de Internet más utilizando a nivel mundial: Google. Véase la evolución desde un HTML simple, hasta una versión que es más que HTML; es un conjunto de tecnologías que le brindan una interfaz más vistosa y que puede adaptarse a toda esa plasticidad que los usuarios requieren.

2.2. AJAX

AJAX "acrónimo de Asynchronous JavaScript + *XML* es traducido al español como JavaScript Asíncrono + XML".[2]

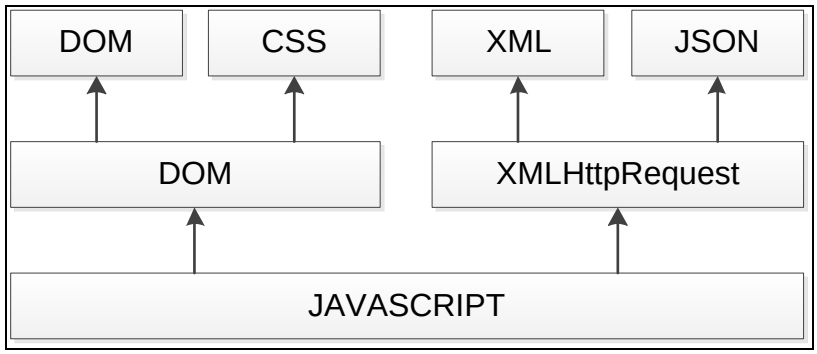
[2] PÉREZ, Javier, *Introducción a AJAX. P. 5.*

AJAX, fue por primera vez definido como: “AJAX no es una tecnología en sí mismo. En realidad, se trata de varias tecnologías independientes que se unen en formas nuevas y sorprendentes”.[3]

Las siguientes son las tecnologías que conforman el concepto de AJAX:

- XHTML y CSS: capa de presentación
- DOM: interacción y manipulación dinámica de la presentación
- XML, XSL, JSON: intercambio y manipulación de la información
- XMLHttpRequest: intercambio asíncrono de la información
- JavaScript: capa de enlace de las tecnologías

Figura 8. **Arquitectura de AJAX**



Fuente: elaboración propia.

[3] *AJAX: a new approach to Web applications*. http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications. Consulta: 10 de enero de 2011.

Recordar que la transmisión asíncrona es aquella que se da entre la participación de un emisor y un receptor donde no existe un canal temporal entre las mismas, es decir que no existe una línea física autónoma que controle el tiempo de transmisión.

2.2.1. AJAX contra aplicaciones tradicionales

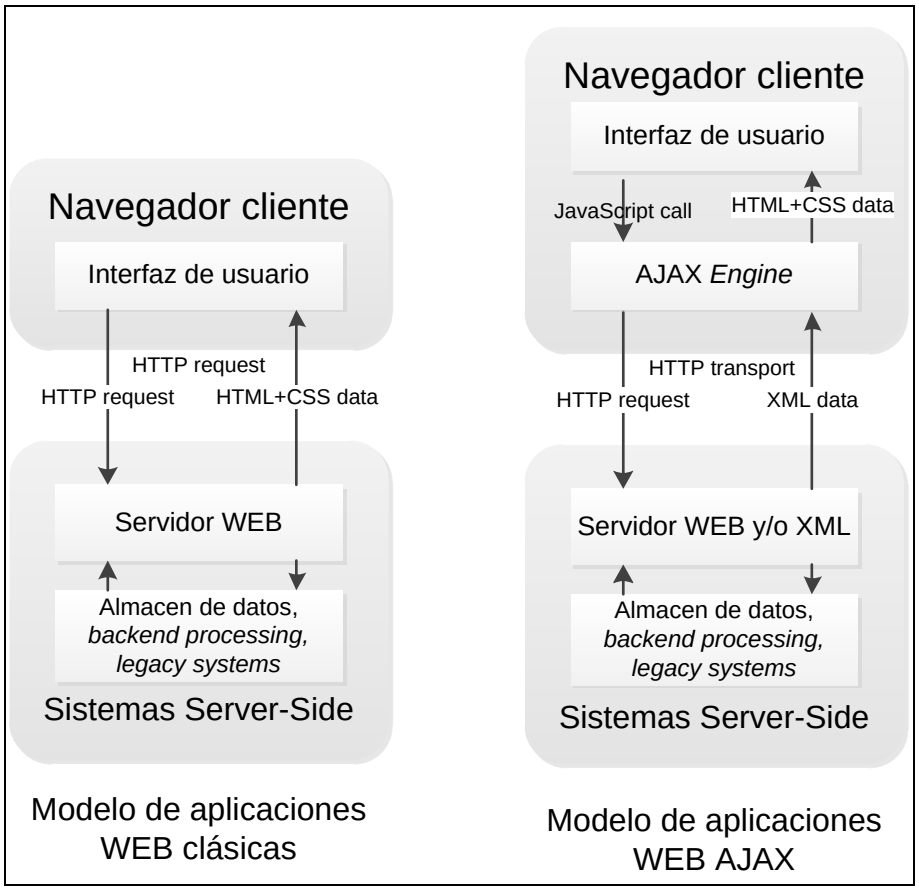
Para el desarrollo de aplicaciones basadas en AJAX, se requiere un conocimiento avanzado de cada una de las tecnologías que la conforman, para el aprovechamiento de las bondades que las mismas ofrecen.

En el desarrollo de este sistema para la administración de proyectos individuales, se tendrá un enfoque en el uso de dichas tecnologías, buscando con ello mejorar significativamente la experiencia del usuario sobre el uso del mismo.

En la figura 9, puede observarse que, en las aplicaciones Web tradicionales, por cada petición de usuario, se necesitaba realizar una llamada al servidor. La respuesta a esta llamada por parte del servidor es una nueva página HTML al cliente; la misma también describe el modelo de *request*, *response* (solicitud, respuesta) utilizado por el modelo de aplicaciones a estudiar e implementar.

De igual forma en la figura 9 se muestra un cuadro comparativo en el cual se observa que AJAX implementa una capa extra en el navegador del cliente, el AJAX Engine; la cual se dedica a realizar las peticiones al servidor, con lo que este ya no devolverá el contenido HTML, sino contenido en formato XML; este a la vez será interpretado por la AJAX Engine y enviado al cliente en formato HTML.

Figura 9. **Aplicaciones AJAX y aplicaciones tradicionales**

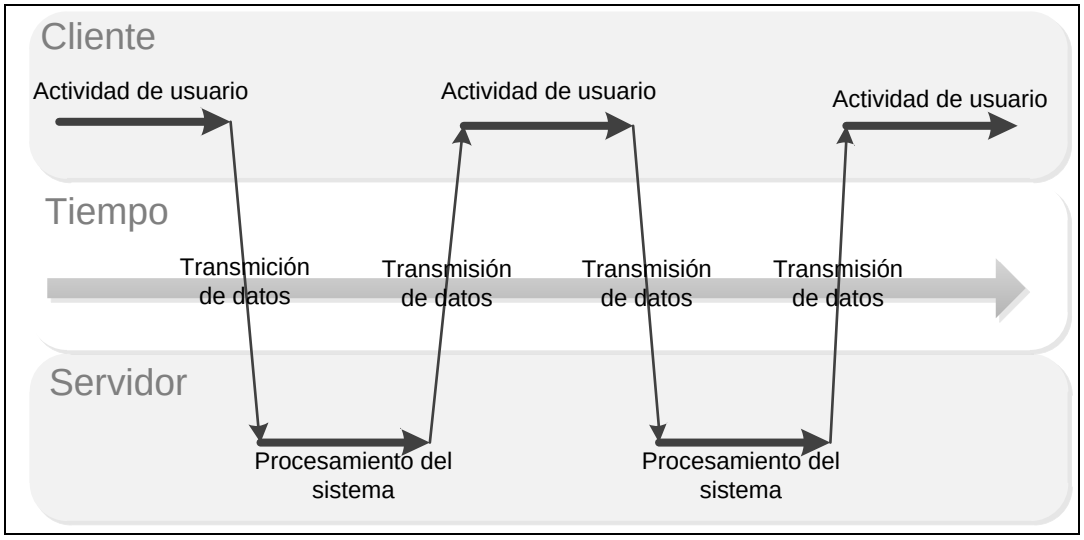


Fuente: elaboración propia.

AJAX, mejora entonces la usabilidad de las aplicaciones y la interacción con el usuario, evitando la constante recarga de las páginas, debido a que el AJAX Engine se dedica a la recarga de contenidos del lado del servidor y esta se realiza en segundo plano; lo que permite que el usuario nunca se encuentre con una página en blanco, mientras espera la respuesta del servidor.

La situación anterior se puede visualizar en páginas como Picasa[4], Gmail o Facebook, al visualizar las fotos en Picasa y Facebook, no hay necesidad de recargar la página por completo; de igual manera ocurre para la visualización de correos electrónicos en Gmail, debido a que la recarga solamente se realiza por medio de una solicitud de JavaScript, que es procesada por la capa intermedia entre el interfaz de usuario y el servidor de aplicaciones, lo que en el navegador se conoce como la AJAX Engine.

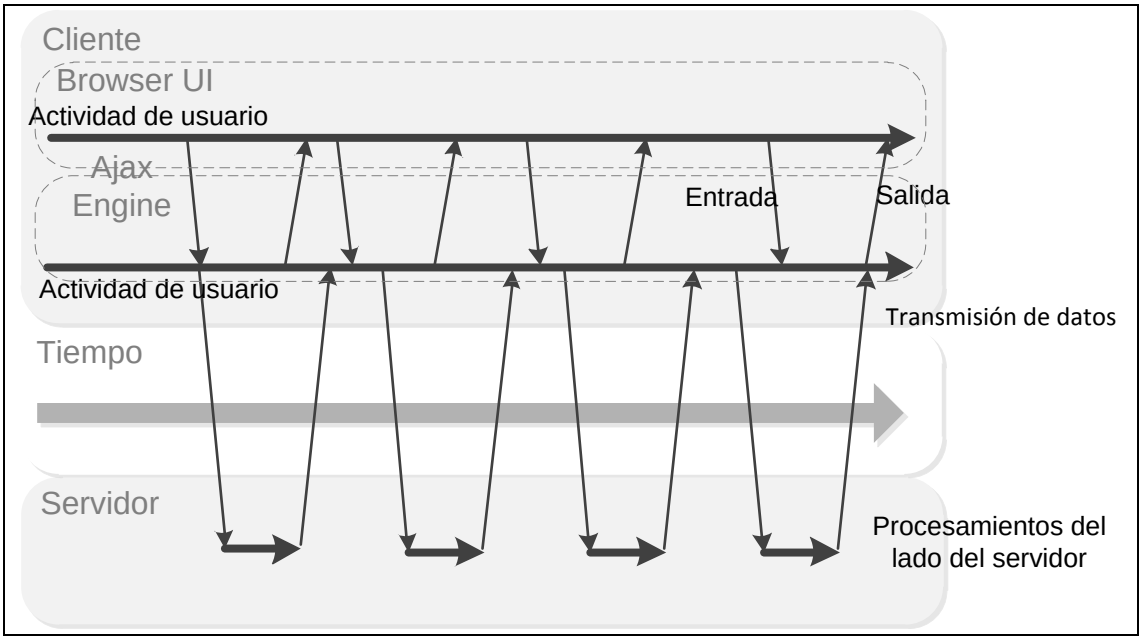
Figura 10. **Transmisión síncrona de aplicaciones Web tradicionales**



Fuente: elaboración propia.

[4] Servicio de almacenamiento de imagen Picasa http://picasaWEB.google.com/. Consulta: 12 de enero de 2011.

Figura 11. **Transmisión asíncrona de aplicaciones Web con AJAX**



Fuente: elaboración propia.

En las figuras 10 y 11 se observa que a diferencia del modelo síncrono, en el modelo asíncrono las peticiones HTTP[5] son sustituidas por peticiones JavaScript que se realizan a la AJAX Engine; dichas peticiones son simples debido a que no requieren la intervención del servidor.

En caso que una solicitud la requiriera, se realizará una petición de manera asíncrona mediante AJAX, sin interrumpir la interacción del usuario debido a cargas de páginas o largas esperas en la respuesta.

[5] Definición de protocolo http: http://www.w3.org/Protocols/. Consulta: 12 de enero de 2011.

Hasta esta parte del documento se ha descrito a AJAX como una tecnología, que tiene su esencia principal en el componente XMLHttpRequest del navegador, o por su capacidad de implementar una herramienta llamada AJAX Engine entre el navegador y la aplicación, para proveer un sistema de comunicación asíncrona; sin embargo esta no es la totalidad de la esencia de AJAX, sino su esencia es la capacidad que permite integrar diversas tecnologías para mejorar la interfaz de usuario y darle riqueza a las aplicaciones.

2.3. HTML

HyperText Markup Language (HTML) es un lenguaje interpretado del lado del cliente por medio de un navegador Web, es un lenguaje estándar para la elaboración de páginas Web; dicho lenguaje brinda flexibilidad y adaptabilidad para ser utilizado de manera conjunta con lenguajes que pueden ser embebidos. Texto de Pro HTML5 programing de Peter Lubbers.

2.3.1. Breve historia

La historia de HTML se remonta a su primera publicación en 1993[6], durante esta década, el desarrollo del lenguaje fue extenuante, aparecieron las versiones 2.0, 3.2 y 4.0 en el mismo año, hasta que en 1999 apareció la versión 4.01[7] que es el estándar de mayor utilización y presencia en las páginas Web hasta el día de hoy.

---

[6] Primera especificación de HTML: http://www.w3.org/History/19921103-hypertext/hypertext/WWW/MarkUp/Tags.html. Consulta: 13 de enero de 2011.
[7] Especificación 4.01 de HTML. http://www.w3.org/TR/1999/REC-html401-19991224/: Consulta 13 de enero de 2011.

Así como se dio el rápido crecimiento de HTML, se pensó en el fin del mismo como lenguaje; en ese entonces la Web se enfocaba más en XML y XHTML. Pero HTML no se resistía a morir y por lo cual surgió una nueva especificación.

Se inició con el desarrollo de HTML 5(HyperText Markup Language, versión 5). Con el cual se busca llevar las plataformas Web a un nuevo nivel; se inició el desarrollo con el nombre del proyecto llamado Web HyperText Application Working Group (WHATWG)[8] en 2004. W3C retomó los cambios en HTML en 2006, publicando por primera vez el estándar en 2008.

Los desarrolladores consideraban a HTML 5 como: la "segunda nueva Web", donde a los sitios estáticos se le convertían en sitios dinámicos, era esa dinámica que sitios como las redes sociales necesitaban, para brindarle al usuario esas características que solo encontraban, en aplicaciones de escritorio. Texto de Professional Rich Internet Applications de Dana More.

HTML5 aún tiene varios problemas, pero el principal es la estandarización, pues aún no todos los navegadores soportan por completo dicha especificación, y los usuarios de Internet se niegan en cambiar su explorador Web antiguo, es más, muchas de las características de la especificación del HTML5 no son soportadas por las versiones "estables" de los navegadores tradicionales (los instalados por defecto en los sistemas operativos en su mayoría Windows XP o Windows Vista).

[8]Comunidad WHATWG: http://www.whatwg.org/. Consultado el 15 de enero de 2011.

2.3.2. Principios de HTML5

Los principios de diseño en los que está basado el desarrollo de HTML 5, según lo describe su especificación (WHATWG, 2004); HTML 5 está elaborado sobre una nueva visión de posibilidad y viabilidad, dichos principios son:

- Compatibilidad: esta característica está en proceso, por ello HTML5 aún se encuentran en desarrollo, grupos como WHATWG, W3C y la Internet Engineering Task Force (IETF), la primera trabajando junto con grupos de desarrollo de Apple, Mozilla, Google y Opera trabajando en un API y Web *Browser* compatibles; la segunda realizando cambios y distribuyendo la especificación; mientras que la última, desarrollando los componentes necesarios para su soporte con protocolos como el HTTP.

- HTML 5, no pretende reinventar la rueda, por ejemplo se ha invertido gran cantidad de esfuerzo analizando comportamientos, donde Google, luego de analizar millones de páginas, se ha descubierto que es común que el nombre de ID para las etiquetas DIV, las personas utilicen DIV id= *"header*" para enmarcar el contenido de la cabecera. HTML5 trata de brindarle alternativas a los desarrolladores, implementando una etiqueta que se llame <*header*>, según lo expresa Pro HTML5 programming de Peter Lubbers.

- Utilidad*:* la escritura y definición de los estándares de HTML5, están enfocados en los usuarios y los desarrolladores Web, por encima de los que crean la especificación W3C o WHATWG. Esto conlleva a la creación de un código, demasiado práctico, pero dejando la perfección de lado.

Por ejemplo las siguientes líneas de declaración son válidas en HTML5:

id = “ejemplo”
Id = ejemplo
ID = “ejemplo”

Aunque esto solamente sería válido para el intérprete del navegador, pues sí se utiliza XHTML5 para la validación de cadenas XML. Aunque la colocación en serie de HTML o XHTML podrían producir los mismos árboles DOM, con mínimas diferencias, pero debido a que la sintaxis de XHTML es más estricta, las últimas dos líneas no serían válidas.

- Seguridad de diseño: el desarrollo de HTML5 se basa en la implementación de la seguridad, cada una de las partes de la especificación poseen secciones de consideración de seguridad. Los modelos de seguridad permiten hacer cosas como comunicarnos de manera segura a través de dominios, sin necesidad de tener que realizar pasos extenuantes e ingeniosos. Dichos modelos ya se han utilizado en diferentes API.

- Separación de presentación de contenido se hace uso de las ya conocida hojas de estilo CSS, de la cual se expondrá más adelante. Aunque hay que utilizar las nuevas características que se implementan a CSS, debido a que las que eran compatibles con HTML 4.01, ya no son nativas de HTML5, pero incluso así funcionan gracias al principio de compatibilidad de diseño.

HTML5 busca subsanar carencias que se venían dando como:

- Pobre accesibilidad.
- Complejidad innecesaria (código difícil de interpretar, debido a los códigos de estilo).
- Documentos de código demasiado largos (repetición de contenido, código más lento).

- Interoperabilidad*:* simplificación de la interoperabilidad, “simple es mejor”; se debe simplificar a la medida que fuera posible, por ejemplo.

  - Sustituye a códigos complejos de JavaScript
  - Un nuevo y simplificado Doctype
  - Un nuevo y simplificado set de declaración de caracteres
  - Una API poderosa y sencilla de HTML5

  Todo lo anterior aunado con un documento de especificaciones detalladas, para evitar malas interpretaciones en un documento de más de 900 páginas. Asimismo como un intérprete robusto llamado Error handling planes. Dicho intérprete permite la recuperación de errores.

- Accesibilidad: la accesibilidad universal es un principio que se divide en tres conceptos:

  - Accesibilidad para los usuarios con discapacidades: HTML5 trabaja en relación con un estándar llamado Iniciativa de accesibilidad Web (WAI, por sus siglas en inglés).

AccessibleRich Internet Application (ARIA), WAI-ARIA[9].El cual brinda compatibilidad con lectores de pantalla.

- Independencia de los medios de comunicación: HTML5 debe de funcionar de igual manera en todo tipo de dispositivos, ya sea computadoras de escritorio, tabletas o celulares.

- Soporte para todos los lenguajes del mundo: por ejemplo, la nueva etiqueta <ruby> soporta las anotaciones Ruby que utilizan tipografía del este de Asia.

En el 2007 aún se creía en la propuesta de un lenguaje libre de *plugin* privativo, pero debido a presiones de la industria esto no pudo ser, debido a que los objetos dinámicos y el vídeo llegaron a la Web con estándares cerrados como Quick Time y Flash. La idea era buena, en la especificación se esperaba utilizar formatos de código abierto, como el OggTheora[10], pero la definición no estuvo del todo correcta, por ejemplo, las etiquetas *object* y *embed* no se retiraron aunque si desparecen algunos atributos de *object* como *archive*, *classid*, *codebase*, *codetype*, *declare* y *stamby*.

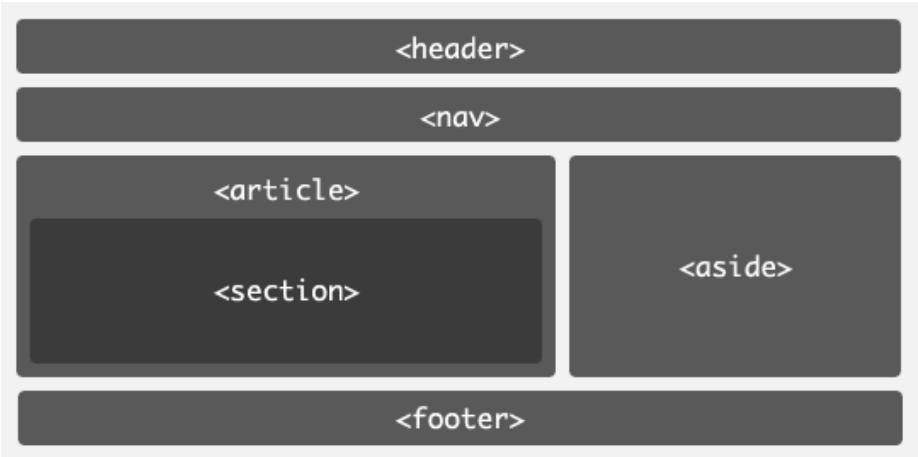
En la figura 12, se muestra un esquema comparativo que presenta cambios en la especificación HTML y HTML 5.

[9] Revisión WAI-ARIA http://www.w3.org/WAI/intro/aria. Consulta: 20 de enero de 2011.
[10] Acerca de OggTheora: http://www.theora.org/. Consulta: 20 de enero de 2011.

Figura 12. **Esquema de una página en HTML 4.0.1**

Fuente: http://www.johandebruin.com/wp-content/uploads/2010/11/paradigma-html5.jpg. Consulta: 20 de enero de 2011.

Figura 13. **Esquema de una página en HTML 5**



Fuente: http://www.johandebruin.com/wp-content/uploads/2010/11/paradigma-html5.jpg. Consulta: 20 de enero de 2011.

2.4. JavaScript

JavaScript (Odell, 2009), es el lenguaje dinámico que le da vida a las aplicaciones RIA. El navegador Netscape fue el primero en soportar las funcionalidades dinámicas de dicho lenguaje de programación, que originalmente se llamó LiveScript, creado por Brendan Eich[11] de Netscape e incluido en la versión de su navegador en 1995.

Cambiando el paradigma de la programación Web, reemplazando los tipos estáticos a dinámicos, los programadores consiguieron reducir las constantes solicitudes del servidor para realizar tareas simples, como la validación de datos, realizando estos procesamientos del lado del navegador, dando el primer paso importante para la creación de las tecnologías RIA.

JavaScript como lenguaje tiene una sintaxis similar a la de C, aunque adopta nombres y convenciones propios del lenguaje Java, es importante definir que ambos lenguajes no tienen relación y están desarrollados para propósitos diferentes. Los navegadores cuentan con una implementación de DOM la cual permitirá interactuar con el lenguaje de programación JavaScript.

2.5. GWT

El Google Web Toolkit trabaja como un *framework* de desarrollo sobre lenguaje Java, es de código abierto; brinda las herramientas necesarias para desarrollar aplicaciones Java, brindando una interfaz de depuración y desarrollo. Según los expresa GAE and GWT Application de Daniel Guermeur.

[11] About Brendan Eich: http://brendaneich.com/. Consulta: 30 de enero de 2011.

Todo esto se logra utilizando un IDE de desarrollo de preferencia personal, GWT permite la traducción de aplicaciones escritas en lenguaje Java, hacía lenguajes JavaScript y HTML, logrando que todo el código en AJAX sea más comprensible y por ende manipulable.

2.5.1. Características de GWT

La siguiente es una lista de las características y bondades que GWT proporciona como herramienta para el desarrollo de aplicaciones basadas en RIA (esgooglewebtoolkit, 2009).

- Reutilización: la reutilización de componentes, para la creación de interfaces de usuario de forma dinámica, ya que a partir de la creación de un *Widget* se pueden construir otros; cada uno de los *Widget* automáticamente se colocan en paneles y son enviados a otros usuarios a través de archivos JAR.

- RPC sencillo: permite ejecutar Remote Procedure Calls de forma sencilla desde el navegador al servidor de aplicaciones, utilizando un objeto serializarle. GET serializa de forma automática las peticiones del servidor y se analizan las respuesta del mismo. Las funciones de RPC le permiten manejar jerarquía, en el polimorfismo de clase, así como llevar un correcto manejo en las excepciones.

- Administración del historial del navegador: se puede ejecutar una aplicación sin necesidad de recargar ni una sola vez el navegador, como una característica de las aplicaciones AJAX; su aplicación implementada con GWT no necesitará usar el botón atrás del navegador.

- Depuración en tiempo real: debido a que la aplicación es escrita en lenguaje Java, la depuración es realizada en dicho lenguaje por las herramientas de depuración propias del IDE que se utilice.

- Compatibilidad: implementada en los navegadores más populares como Firefox, Google Chrome, IE, Safari y Opera, ya que se apega a los estándares de W3C.

- Internacionalización: facilidad para la creación de librerías para aplicaciones de internacionalización rápida y sencilla.

- Interoperabilidad: debido a que las librerías de clases de Java no están implementadas en su totalidad, se puede utilizar lenguaje JavaScript, utilizando la interfaz nativa de *scripts* de Java (JSNI).

2.5.2. Arquitectura de GWT

La arquitectura de GWT, está compuesta por cuatro componentes principales, el primero es un compilador Java a JavaScript, un navegador *hosted*y dos librerías de clases, las cuales son (esgooglewebtoolkit, 2009):

- Compilador GWT Java a JavaScript: este es el encargado de traducir del lenguaje de programación Java a JavaScript; el compilador es utilizado cuando se necesite correr la aplicación en modo Web.

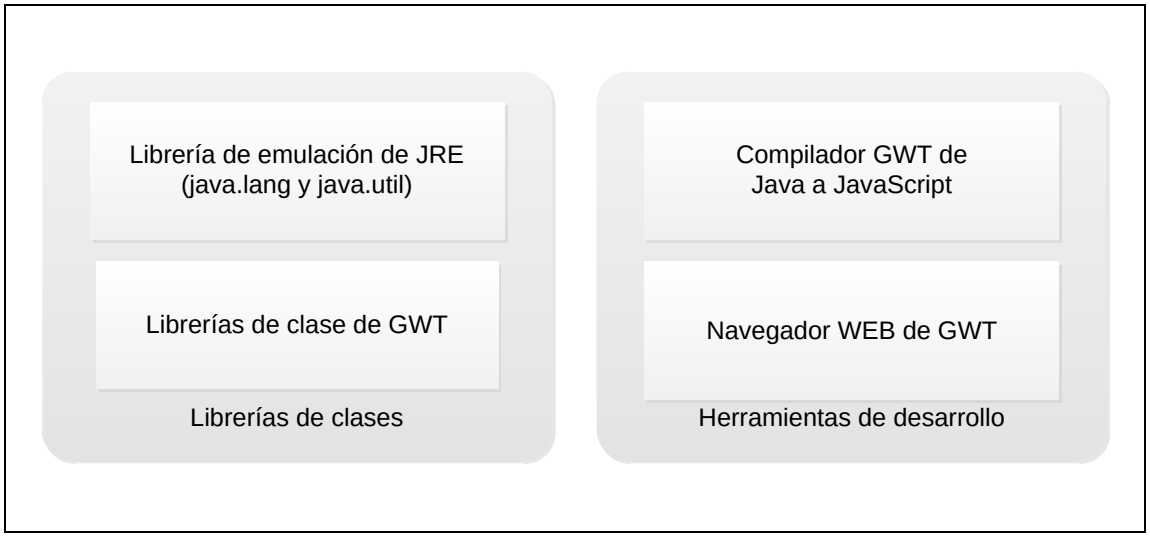
- Navegador Web *hosted* de GWT: este permite correr y ejecutar las aplicaciones GWT en modo *hosted*, es decir que lo que se ejecuta, son los *bytecodes* de Java sobre una máquina virtual sin compilarlos a JavaScript.

Para ello en el navegador, GWT incrusta un controlador de *browser* especial con *hooks*, dentro de una máquina virtual de Java.

- Emulación de librerías JRE: GWT tiene a su disposición las implementaciones en JavaScript de las librerías de clases Java, tales como los casos de las librerías *java.lang* y *java.util*.

- Librerías de clases de interfaz de usuario de GWT: son un conjunto de interfaces y clases que permiten la creación de *widget* en el navegador, tales como botones, imágenes, cajas de texto.

Figura 14. **Arquitectura de GWT**

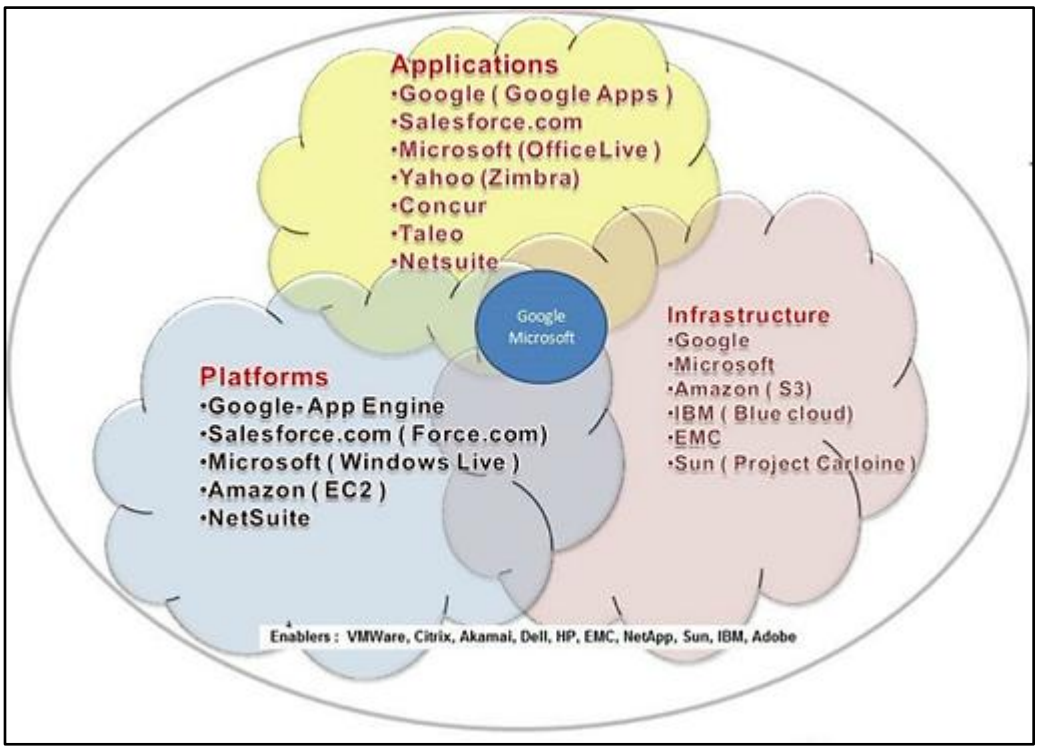


Fuente: elaboración propia.

2.6. *Cloud computing*

El concepto de *cloud computing* se refiere a una tendencia tecnológica que permite un nuevo inicio en la dirección y creación de mercados tecnológicos. *cloud computing* se refiere también a una idea de mensajería personal (*email*, procesadores de palabras, presentaciones, etc.) y aplicaciones productivas para negocios (automatización de fuerza de ventas, servicio al cliente, etc.) desde servidores centralizados que permiten compartir recursos, tanto como almacenamiento, procesamiento y ancho de anda, de forma más eficiente y con costos menores.

Figura 15. **Arquitectura de *cloud computing***



Fuente: LYNCHMerril, IDC Saas Report, Mayo, 2008.

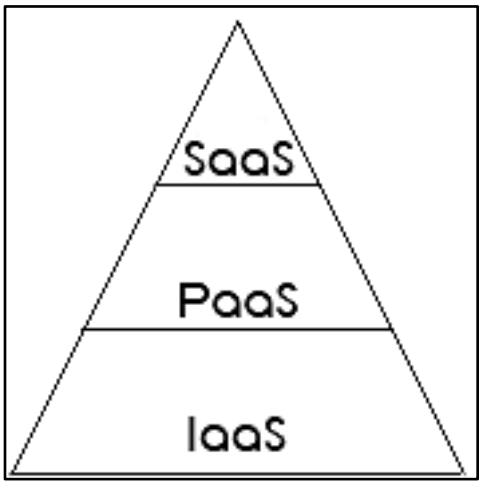
En la figura 15, puede observarse que el concepto de computación en nube está dividido en 3 segmentos, los cuales son: infraestructura, plataforma y aplicaciones; a continuación se presenta una breve descripción:

- IaaS(Infrastructure as a Service): en el nivel de infraestructura como servicio, se encuentra la infraestructura física de la nube, computación en *grid*, una enorme red de ordenadores organizados en *datacenters*, con sistemas operativos reducidos, virtualización de *hardware*, con sistemas de ficheros hechos a la medida, balanceadores de carga, sistema de seguridad, para la detección de intrusos.

- PaaS (Platform as a Service):en el nivel de plataforma como servicio, se encuentra una capa que le permite acceder a los servicios que soporta IaaS, ocultando los detalles que conforman el IaaS, al cual nunca podrá acceder. Donde los recursos son tan ilimitados como el proveedor lo permita, en este nivel es donde opera Google App Engine, el cual presenta una plataforma de administración limpia e intuitiva, con informes de recursos utilizados, a partir de estadísticas.

- SaaS (Software as a Service): *software* como un servicio, en esta capa se encuentra una infraestructura y una cantidad de facilidades que tiene como objetivo: el poder crear aplicaciones que puedan ser utilizadas por los usuarios finales y sus clientes.

Figura 16. **Arquitectura de computación en nube**



Fuente: elaboración propia.

2.7. Google APP Engine

Google App Engine (GAE) es una herramienta que permite ejecutar aplicaciones Weben la infraestructura de Google. GAE cuenta con su propia implementación de un SDK para el desarrollo, utilizando los lenguajes Java y Python. El servicio de nube GAE permite no disponer de otro servidor, ya que ahí podrán subir las aplicaciones para que otros usuarios las utilicen inmediatamente.

El servicio cuenta con integración con GWT y un complemento para el IDE Eclipse que le permitirá publicar las aplicaciones Web AJAX; GAE cuenta con 1 GB de almacenamiento gratuito, suficiente recurso con la opción de pagar solamente por los recursos que utilice, así como también con el ancho de banda suficiente y capacidad de procesamiento (CPU).

Es decir, que permite a las aplicaciones tener hasta 5 millones de visitas por mes, (para la disponibilidad que requiere el sistema a implementar esto es más que suficiente).

GAE aumenta el ancho de banda de acuerdo con los requerimientos y necesidades de almacenamiento de datos.

Google App Engine es entonces una plataforma que permite acceder a los recursos de Google con el objetivo de crear aplicaciones que funcionen en la nube, el cual se encuentra en la capa de PaaS; esta le permitirá utilizar recursos casi infinitos, por lo que se puede manejar de manera despreocupada la infraestructura que proporciona Google App Engine.GAE en su versión 1.5.4 cuenta con los siguientes beneficios[12]:

- 1 GB de ancho de banda de salida por día
- 1 GB de ancho de banda de entrada
- Envío de 2000 emails por día
- Tiempo de CPU por día 6.5 horas
- Disponibilidad de 24/7 en la ejecución de instancias
- 1 GB de almacenamiento
- Hasta 657,000 invocaciones en la obtención de URL

2.7.1. App Engine servicios escalables

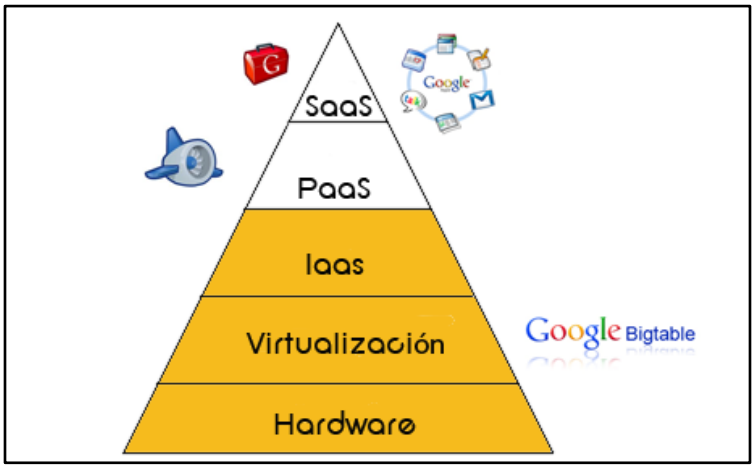
GAE soporta un buen número de útiles servicios, todos diseñados para escalar de la mano con las aplicaciones.

[12]http://code.google.com/intl/es-ES/appengine/docs/quotas.html.

Este listado de aplicaciones sigue creciente, pero las más relevantes son[13]:

- El almacén de datos DataStore
- API para autenticación provista por Google Accounts y otros servicios de OAuth[14].
- API de Java Mail para enviar y recibir e-mail
- Soporte a colas de tareas, permitiendo realizar el trabajo de forma asíncrona sin que el usuario se percate de lo que sucede.
- Soporte de la programación y ejecución de tareas; *Cron Jobs*
- Soporte para el procesamiento de imágenes
- Integración con Google Apps

Figura 17. **Arquitectura de la nube de Google**



Fuente: elaboración propia.

[13] GUERMEUR, Daniel. Google App Engine Java and GWT application developmet. P.35.

[14] Comunidad de OAuth: http://oauth.net/. Consulta: 6 de febrero de 2011.

2.8. El Datastore

El almacén de datos de GAE almacena datos de objetos que trabaja sin esquema, contiene un motor de búsqueda que permite ejecutar transacciones atómicas. El SDK de Java de GAE incluye implementaciones de objetos de datos Java (JDO), la interfaz del API de persistencia de Java (JPA), así como un API de almacén de datos de nivel inferior. Como se lee en la Google App Engine Documentation, 2010.

El Datastore se define a sí mismo como un potente servicio distribuido de almacenamiento de datos, que incluye además, un motor de búsqueda y transacciones que crece con los datos, al igual que un sistema Web distribuido, crece con el tráfico de usuarios.

El Datastore nace como la solución a un problema recurrente en todos los sistemas de información; el manejo de grandes cantidades de información, es un problema que Google se dispuso solucionar al desarrollar Google App Engine, donde implementó el uso del servicio de BigTable para empresas.

Con el Datastore, se deja de utilizar modelos de base de datos relacionales, ya que las tecnologías a utilizar son diferentes a las empleadas por sistemas de gestión de bases de datos como MySQL u Oracle, para lograr este objetivo Google emplea sus propias tecnologías, las cuales están basadas en GFS (Global File System) y BigTable.

2.8.1. GFS

El Google File System, es un sistema distribuido de archivos, escalable para aplicaciones de tratamiento intensivo de datos, el cual ofrece tolerancia a fallos durante la ejecución, así como bajo costo de utilización del *hardware*, aumentar el rendimiento, trabajando aun con un alto número de clientes. Como se lee en The Google File System de Fay Chang.

El Google File System está diseñado con base en los requerimientos, demandas tecnológicas y el rápido crecimiento de la demanda de Google, la cual fue diseñada bajo cuatro conceptos que se describen a continuación:

- Primero: el fallo de los componentes son la regla y no la excepción. Por lo cual el sistema de archivos consiste en miles de ordenadores clientes construidos de componentes baratos, que son accedidos por un comparable número de máquinas. La calidad y la cantidad no son características garantizadas, por lo cual si hay algún desperfecto en el *hardware*, ya sea por error de aplicaciones, *hardware*, humano, red o suministro de energía, estas son remplazadas sin necesidad de realizar respaldos, debido a que se implementó un sistema de monitoreo constante e integral, que permite realizar respaldos automáticos.

- Segundo: los archivos hasta el día de hoy son demasiado grandes para los estándares tradicionales. Los archivos de varios GB son cada vez más comunes, cada uno de ellos típicamente contiene tantos objetos como un documento Web.

Aunque Google trabaja regularmente con conjuntos de datos de gran tamaño y de rápido crecimiento, los archivos de muchos TB que comprimen billones de objetos, ocasiona que sean muy difíciles de manejarlos aunque el sistema de archivos utilizado lo soporte. Como resultado, Google diseñó sus propias asunciones y parámetros para las operaciones I/O y la revisión del tamaño de los bloques.

- Tercero: la mayoría de los archivos cambian añadiendo nuevos datos en lugar de escribir en los datos existentes. La escritura aleatoria dentro de un archivo es prácticamente inexistente. Una vez escritos, los archivos serán de solo lectura y la mayoría de las veces serán secuenciales. Este patrón de accesos a archivos de gran tamaño, añaden rendimiento y optimización, garantizan atomicidad, mientras que los bloques de datos de almacenamiento en caché del cliente van perdiendo su atractivo.

- Cuarto: el codiseño de las aplicaciones y el API de sistema de archivos, beneficia al sistema en general, aumentado con ello la flexibilidad, debido a que se ha rebajado el peso, en el modelo GFS, sin perder con ello su consistencia. Los grupos de GFS están desplegados actualmente para diferentes propósitos. Los más grandes tienen 1000 nodos de almacenamiento, con más de 300 TB de disco y son visitados por cientos de clientes de manera concurrente.

2.8.2. Bigtable

Bigtable es un sistema de almacenamiento distribuido, para el manejo de estructura de datos. Bigtable está diseñado para escalar a *petabytes* de forma fiable, los datos de miles de máquinas.

Con Bigtable se alcanzaron varios objetivos como: obtener una amplia aplicación, escalabilidad y alto rendimiento, debido a que *Bigtable* es utilizada por más de 60 productos de Google, incluyendo: Google Analytics, Google Finance, Google Earth, Google App Engine, entre otros. Como se lee en The Google File System de Fay Chang.

Dichos productos tienen una gran cantidad de carga, que van desde, el rendimiento del procesamiento por lotes, trabajos de latencia sensible, hasta las solicitudes de los usuarios.

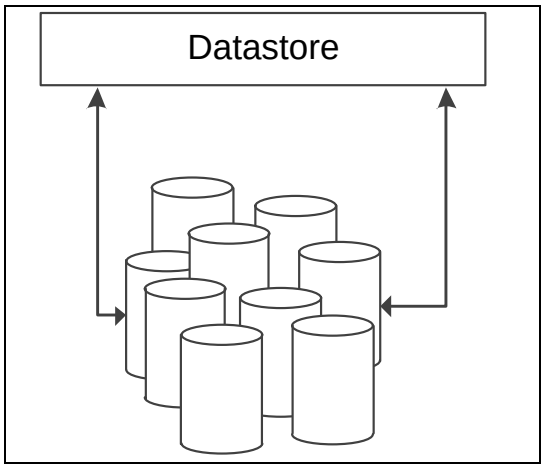
Bigtable presenta similitudes con los bases de datos convencionales como las siguientes: permite bases de datos paralelas y el uso de una memoria principal, proporciona escalabilidad que permite lograr alcanzar un alto rendimiento, no es compatible con el modelo de datos relacional, pero a su vez proporciona un modelo de datos simple, que permite el control dinámico sobre la disposición de datos y su formato y decidir a los clientes decidir sobre las propiedades que representan a los datos y el almacenamiento subyacente, el cual puede ser:

- Magnético
- Óptico
- En nube

Los datos son indexados en filas y columnas con nombres que pueden ser cadenas arbitrarias. Bigtable también trata los datos como cadenas sin interpretar, aunque los clientes suelen serializar, las diversas formas de estructurarlos.

Los clientes pueden controlar la localidad de sus datos a través de la aplicación con minuciosas opciones de esquema. Los parámetros del esquema permiten a los clientes controlar dinámicamente el acceso a los datos de la memoria o del disco.

Figura 18. **Representación del Datastore y *Bigtable***



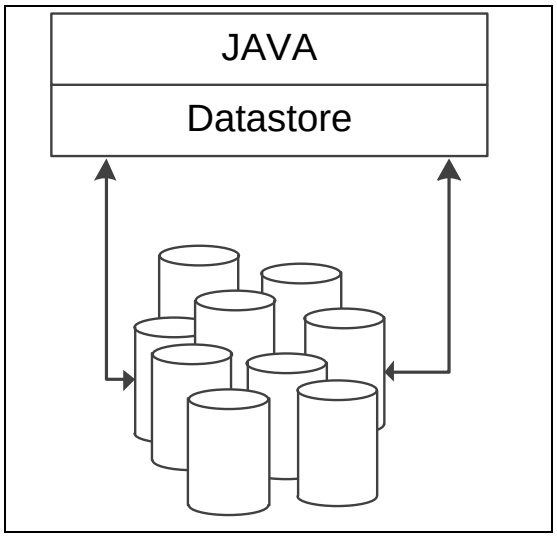
Fuente: http://adictosaltrabajo.com/tutoriales/datastoreAPIBajoNivel/datastore_01.png. Consulta: 15 de febrero de 2011.

2.8.3. Acceder al Datastore

Ahora que ya se han repasado los conceptos de Datastore y cómo está a grandes rasgos constituido, los principios de Global File System y la Bigtable, se define cómo acceder al mismo; actualmente el Datastore se puede acceder a través del Google App Engine, utilizando los lenguajes de programación soportados Java y Python.

Esto se consigue al agregar una nueva capa al diagrama, de acceso al Datastore, el cual se refleja en la figura que se incluye a continuación, la cual en este caso se encuentra escrita en lenguaje Java pues es el utilizado en la implementación del sistema Web para planificación de proyectos individuales.

Figura 19. **Esquema del Datastore y su relación con el lenguaje Java**



Fuente: http://adictosaltrabajo.com/tutoriales/datastoreAPIBajoNivel/datastore_02.png. Consulta: 15 de febrero de 2011.

Ya que el desarrollo está centrado en lenguaje Java, se describe la forma de acceder con este lenguaje, que será de tres maneras:

- El API de nivel inferior: es la interfaz de programación de nivel inferior. Se dice de este modo, porque se encuentra en la capa más cercana al Datastore. Una de las ventajas principales de su utilización, es la de permitir tener un mayor control; la desventaja que conlleva es la de

necesitar un gran conocimiento sobre el funcionamiento del Datastore. Esta API a su vez, puede realizar operaciones primitivas con datos.

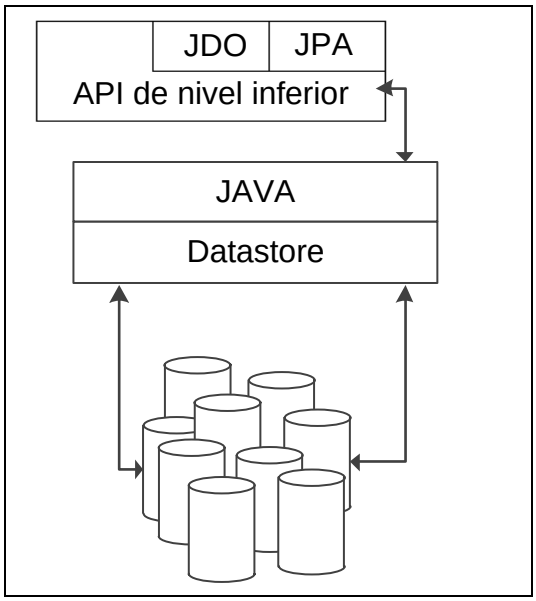
Cuando se programa con dicha API se necesita ser demasiado explicito ya que es necesario indicar no solamente, qué es lo que se quiere almacenar, sino que además, la forma sobre cómo conseguirlo. El API al igual que el Datastore, trabaja con entidades distintas a las que trabaja JDO o JPA; dichas entidades son las que en JAVA se conocen como mapas, pares atributo valor.

- JPA (Java Persistence API): es el API de persistencia de Java, o la interfaz de programación de aplicaciones de persistencia para Java. Esta interfaz le permite definir un estándar de almacenamiento de datos, tal cual se define en una base de datos relacional, para su adaptación y almacenamiento en Datastore; por lo cual se podría decir que se trabaja sobre un falso modelo relacional, donde algunas funciones propias de JPA no están permitidas en App Engine.

- JDO (Java Data Objects): los objetos de datos Java, se definen como una especificación de almacenamiento de objetos en Java; JDO no trabaja bases de datos del tipo relacional, por lo cual encaja con el concepto del Datastore. Debido a que es la herramienta utilizada en el desarrollo del sistema, se ampliará en el siguiente apartado.

En la figura 20 se muestra el esquema de acceso al API de nivel inferior, a la cual se pude ingresar utilizando JDO o JPA; se muestra el lenguaje JAVA pues es el utilizado en el desarrollo de nuestra aplicación, recuerde que también al utilizar las tecnologías que proporciona GAE, podrá utilizar el lenguaje Python.

Figura 20. **Esquema del Datastore y como se accede con un API**



Fuente: http://adictosaltrabajo.com/tutoriales/datastoreAPIBajoNivel/datastore_03.png.
Consulta: 15 de febrero de 2011.

2.8.4. JDO

Objetos de datos Java (JDO),es una interfaz estándar para almacenar objetos, que contienen datos, en una base de datos. El estándar JDO define interfaces para la anotación de objetos Java, la recuperación de objetos a través de consultas y la interacción con una base de datos a través de transacciones. Una aplicación que utiliza la interfaz JDO puede funcionar con diferentes tipos de bases de datos sin utilizar un código específico de bases de datos, incluidas bases de datos relacionales, jerárquicas y de objetos. Al igual que con otros estándares de interfaz, JDO permite que la aplicación migre fácilmente entre diferentes soluciones de almacenamiento.

Lo anterior explica que, JDO no se encuentra restringido al almacenamiento en bases de datos relaciones de forma exclusiva, por esta característica se hace flexible su utilización junto con el Datastore, que de igual manera ya se ha mencionado que no utiliza un modelo relacional en sus bases. JDO sí necesita de una base, la cual es el API de nivel inferior, que como se ha mencionado es la capa más cercana entre el lenguaje y el Datastore.

Pero ¿qué beneficio trae utilizar JDO?, pues la respuesta según lo explica, Expósito, Raúl (2010), JDO te permitirá como desarrollador, el poder centrarse más en "qué" quiere buscar, modificar, ingresar, borrar, etc. En lugar del "cómo". Por lo tanto dicha tecnología permitirá tener un grado de abstracción mayor al que se tendría cuando se trabaja con el API de nivel inferior.

2.8.5. Limitaciones de JDO sobre App Engine

Hacia julio de 2011, el SDK de Java App Engine cuenta con una implementación de JDO 2.3, para el Datastore. La misma está basada en DataNucleus Access Platform, en su versión de *software* libre.

Las siguientes funciones de la interfaz JDO no son compatibles según la Google App Engine documentation, 2011:

- Relaciones sin propiedad (se puede implementar relaciones sin propiedad a través de valores de *key* explícitos: Es posible que se admita la sintaxis de JDO para relaciones sin propiedad en una futura versión).

- Relaciones de propiedad muchos a muchos.

- Consultas con *Join*, (no se puede utilizar un campo de una entidad secundaria en un filtro si realizas una consulta en la entidad principal: Tener en cuenta que es posible comprobar el campo de relación de la entidad principal directamente en la consulta a través de una clave).

- Agrupación JDOQL y otras consultas agrupadas.

- Consultas polimórficas (no se puede realizar una consulta de una clase para obtener instancias de una subclase; cada clase está representada por un tipo de entidad independiente del almacén de datos).

- IdentityType.DATASTORE para la anotación @PersistenceCapable(sólo se admite IdentityType.APPLICATION).

- En la actualidad existe un error que evita que los campos persistentes se guarden en el almacén de datos; esto se solucionará en una futura versión.

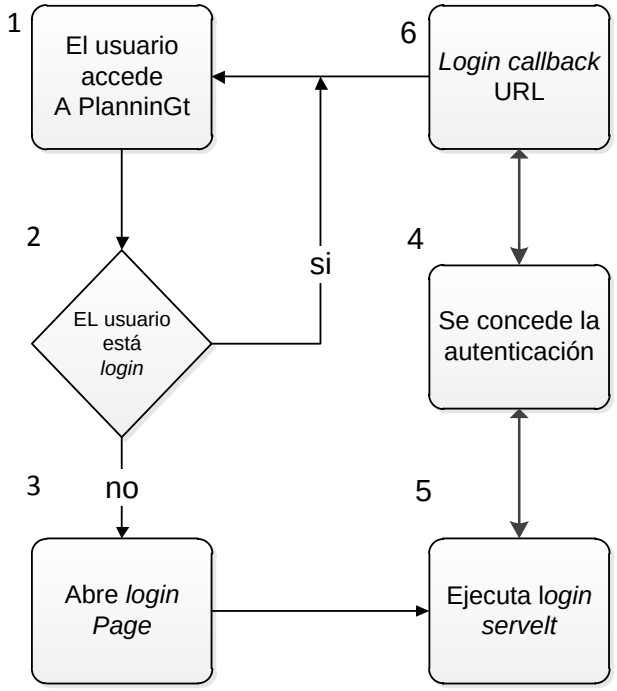
2.9. API Java cuentas de Google

Las aplicaciones que trabajen sobre la arquitectura de Google App Engine, tiene la posibilidad de utilizar el servicio de autenticación de usuarios a través de cuentas de Google. Al utilizar el API de autenticación, una aplicación puede detectar si el usuario actual ha accedido a través de una cuenta de Google, redirigiéndolo inmediatamente a la aplicación o redirigirlo a la página de acceso de cuentas de Google para que acceda a su cuenta o para que cree una cuenta nueva, sino contara con una o así lo desease.

Mientras un usuario se encuentre conectado a la aplicación, esta puede acceder a información como el nombre de usuario y dirección de correo electrónico. La aplicación también puede detectar si el usuario actual es un administrador, lo que facilitará la implementación de áreas exclusivas de administradores en la aplicación. Como se lee en la Google App Engine documentation, 2011.

La implementación de acceso a la aplicación está basada en el diagrama, presentado en la figura 21, donde la aplicación utiliza el API de cuentas Google para redirigir a la aplicación y con ello poder administrar los proyectos con el ID de su cuenta de Google.

Figura 21. **Esquema de autenticación con cuentas Google**



Fuente: elaboración propia.

3. ANÁLISIS Y DISEÑO DE LA APLICACIÓN

Para el desarrollo de la aplicación se optó por adoptar la metodología de programación y desarrollo de *software* eXtreme Programming. (Programación extrema), la cual es una metodología de programación ágil que se enfoca en la satisfacción del cliente y la existencia de pequeños equipos de desarrollo; esta metodología permite a su vez el involucramiento del cliente en el desarrollo del sistema, además XP defiende la filosofía de que el cambio de requerimientos sobre la marcha, es un proceso inevitable.

En un proyecto de investigación los cambios son inevitables y varían proporcionalmente a qué tan nuevas son las tecnologías a utilizar; tal es el caso de las tecnologías de la nube de Google, el *framework* GWT y el uso de Google App Engine, las cuales aún se encuentran en una fase de publicación Beta y deseo de innovar; utilizar al máximo todo este conjunto de tecnologías que proporciona Google, se encuentra aún latente.

3.1. Análisis

En la fase de análisis de documentarán los principales casos de uso de forma gráfica y la definición de cada caso de uso de alto nivel y expandidos.

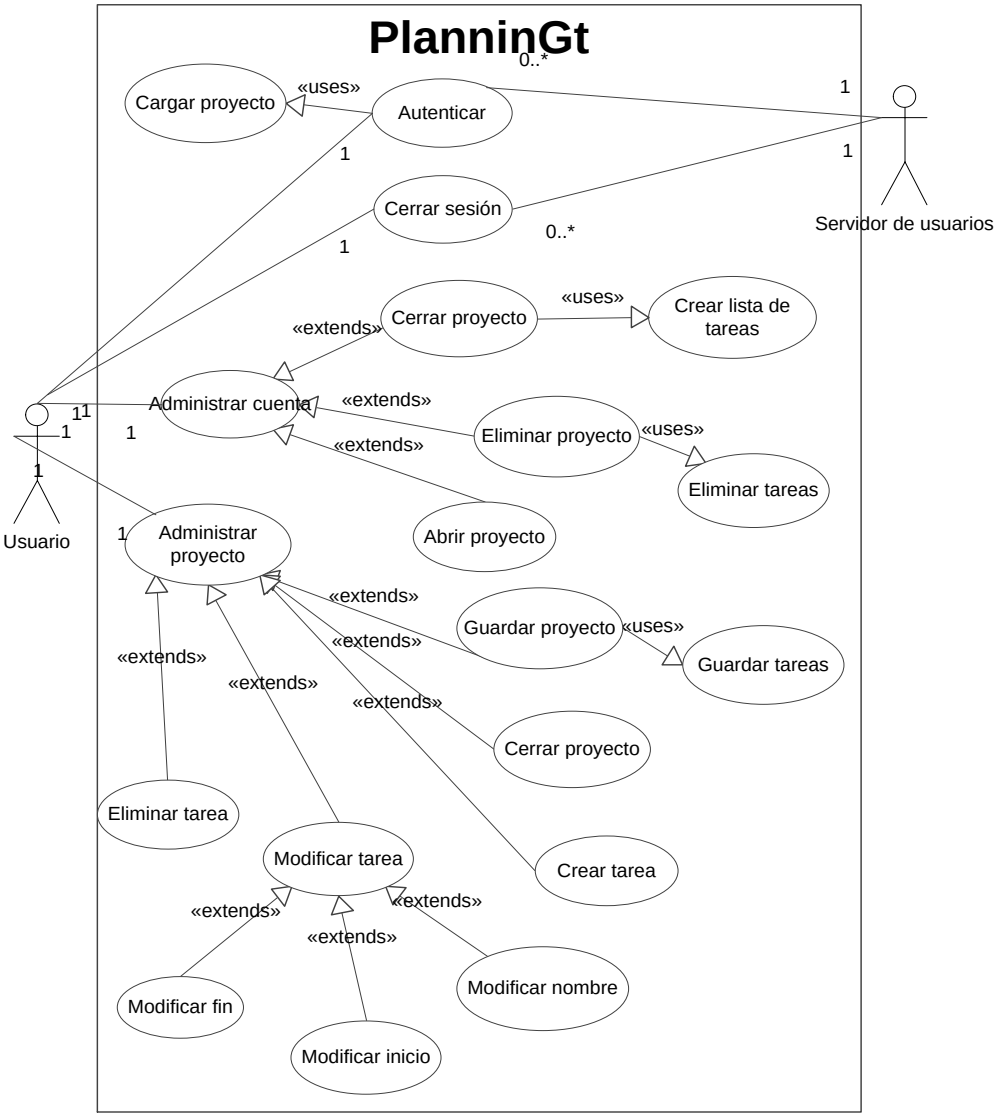
3.1.1. Casos de uso

Cada uno de los usuarios puede ser considerado como un actor, teniendo participación en cada uno de los casos de uso de la aplicación.

3.1.2. Casos de uso de alto nivel

Los casos de uso de alto nivel son utilizados para describir a grandes rasgos los principales elementos del sistema a implementar.

Figura 22. **Diagrama de casos de uso**



Fuente: elaboración propia.

Tabla I. **Caso de uso, autenticar**

Caso de uso	**Autenticar**
Actores	Usuario, servidor de usuarios de Google
Tipo	Primario
Descripción	El usuario entra a la aplicación y se autentica para poder administrar sus proyectos

Fuente: elaboración propia.

Tabla II. **Caso de uso, cerrar sesión**

Caso de uso	**Cerrar sesión**
Actores	Usuario
Tipo	Primario
Descripción	El usuario sale de la aplicación y esta cierra la sesión del mismo

Fuente: elaboración propia.

Tabla III. **Caso de uso, administrar cuenta**

Caso de uso	**Administrar cuenta**
Actores	Usuario
Tipo	Primario
Descripción	El usuario solicita ejecutar altas, bajas y cambios a los proyectos, y el sistema verifica que dicho usuario esté registrado y ejecuta las acciones

Fuente: elaboración propia.

Tabla IV. **Caso de uso, crear proyecto**

Caso de uso	**Crear proyecto**
Actores	Usuario
Tipo	Primario
Descripción	El usuario solicita crear un nuevo proyecto, el sistema verifica que el usuario esté registrado y crea un nuevo proyecto en el Datastore

Fuente: elaboración propia.

Tabla V. **Caso de uso, eliminar proyecto**

Caso de uso	**Eliminar proyecto**
Actores	Usuario
Tipo	Primario
Descripción	El usuario solicita eliminar un proyecto, el sistema busca el proyecto seleccionado en el Datastore y lo elimina

Fuente: elaboración propia.

Tabla VI. **Caso de uso, abrir proyecto**

Caso de uso	**Abrir proyecto**
Actores	Usuario
Tipo	Primario
Descripción	El usuario solicita abrir un proyecto, el sistema lo busca y las tareas asociadas al mismo en el Datastore, para mostrarlas al usuario

Fuente: elaboración propia.

Tabla VII. **Caso de uso, administrar proyecto**

Caso de uso	**Administrar proyecto**
Actores	Usuario
Tipo	Primario
Descripción	El usuario solicita ejecutar altas, bajas y cambios a las tareas de un proyecto seleccionado; el sistema verifica que esté registrado y ejecuta las acciones

Fuente: elaboración propia.

Tabla VIII. **Caso de uso, guardar proyecto**

Caso de uso	**Guardar proyecto**
Actores	Usuario
Tipo	Primario
Descripción	El usuario solicita guardar los cambios realizados en un proyecto y sus tareas, el sistema verifica que el usuario esté registrado y ejecuta las acciones

Fuente: elaboración propia.

Tabla IX. **Caso de uso, cerrar proyecto**

Caso de uso	**cerrar proyecto**
Actores	Usuario
Tipo	Primario
Descripción	El usuario solicita cerrar un proyecto; el sistema advierte que se perderán los cambios no guardados y cierra el proyecto

Fuente: elaboración propia.

Tabla X. **Caso de uso, crear tarea**

Caso de uso	**Crear tarea**
Actores	Usuario
Tipo	Primario
Descripción	El usuario solicita crear una nueva tarea y el sistema la para su disponibilidad

Fuente: elaboración propia.

Tabla XI. **Caso de uso, eliminar tarea**

Caso de uso	**Eliminar tarea**
Actores	Usuario
Tipo	Primario
Descripción	El usuario solicita eliminar una tarea seleccionada, el sistema la busca y la elimina de la tabla

Fuente: elaboración propia.

Tabla XII. **Caso de uso, modificar tarea**

Caso de uso	**Modificar tarea**
Actores	Usuario
Tipo	Primario
Descripción	El usuario selecciona el campo de la tarea que desea modificar e ingresa el nuevo valor. El sistema verifica el nuevo valor y hace los cambios internamente

Fuente: elaboración propia.

Tabla XIII. **Caso de uso, modificar nombre**

Caso de uso	**Modificar nombre**
Actores	Usuario
Tipo	Primario
Descripción	El usuario selecciona el campo nombre de la tarea que desea modificar e ingresa el nuevo valor. El sistema verifica dicho valor y hace los cambios internamente

Fuente: elaboración propia.

Tabla XIV. **Caso de uso, modificar inicio**

Caso de uso	**Modificar inicio**
Actores	Usuario
Tipo	Primario
Descripción	El usuario selecciona el campo inicio de la tarea que desea modificar e ingresa el nuevo valor. El sistema verifica dicho valor y hace los cambios internamente

Fuente: elaboración propia.

Tabla XV. **Caso de uso, modificar fin**

Caso de uso	**Modificar fin**
Actores	Usuario
Tipo	Primario
Descripción	El usuario modifica e ingresa el nuevo valor. Se verifica y el sistema hace los cambios

Fuente: elaboración propia.

3.1.3. Casos de uso expandidos

En los casos de uso expandidos, se brinda una explicación detallada de la secuencia de pasos que un proceso necesita para lograr el funcionamiento del sistema.

Tabla XVI. **Caso de uso expandido, autenticar**

No.	1
Caso de uso	Autenticar
Actores	Usuario (iniciador). Servidor de usuarios de Google.
Propósito	Entrar a la aplicación
Resumen	El usuario entra a la aplicación y se autentica para poder administrar sus proyectos
Tipo	Primario y esencial
Referencias cruzadas	3.2

Curso normal de los eventos; acción del usuario	**Respuesta del sistema**
1. Cuando el usuario visita la aplicación Web	2. Muestra la pantalla de bienvenida en la que aparece el botón de autenticarse
3. Hace click sobre el botón de autenticación	4. Consulta al servidor de usuarios de Google, si el usuario ya ha iniciado sesión
	5. Solicita nombre de usuario y contraseña

Continuación de la tabla XVI.

6. Introduce su nombre de usuario, contraseña y presiona el botón iniciar sesión	7. Solicita al servidor de usuarios que revise los datos y este devuelve el correo y el nombre de usuario para su uso
	8. Muestra el administrador de proyectos y guarda la información del usuario en una variable
Cursos alternos: Línea 2: El usuario ya ha iniciado sesión. Continúa en el paso 7 Línea 4: El usuario no posee una cuenta Google. Indicar registro en Google Línea 4: El nombre de usuario o contraseña incorrectos. Indicar error. Continúa en el paso 5	

Fuente: elaboración propia.

Tabla XVII. **Caso de uso expandido, cerrar sesión**

No.	2
Caso de uso	Cerrar sesión
Actores	Usuario (iniciador), servidor de usuarios de Google
Propósito	Salir de la aplicación
Resumen	El usuario sale de la aplicación y ésta cierra la sesión del mismo
Tipo	Primario y esencial
Referencias cruzadas	R1.8

Continuación de la tabla XVII.

Curso normal de los eventos acción del usuario	**Respuesta del sistema**
1. Este caso de uso comienza cuando el usuario presiona sobre el link “salir”	2. Solicita el cierre de sesión al servidor de usuarios de Google y este cierra la sesión en su servidor
	3. Muestra la página de bienvenida de la aplicación
Cursos alternos: Línea 1: existe un error en la conexión con el servidor de usuarios. Continúa en el paso 2	

Fuente: elaboración propia.

Tabla XVIII. **Caso de uso expandido, crear proyecto**

No.	3
Caso de uso	Crear proyecto
Actores	Usuario(iniciador)
Propósito	Crear un nuevo proyecto y su lista de tareas
Resumen	El usuario solicita crear un nuevo proyecto, el sistema verifica que esté registrado y crea un nuevo proyecto en el Datastore
Tipo	Primario y esencial
Referencias cruzadas	R1.8

Continuación de la tabla XVIII.

Curso normal de los eventos acción del usuario	**Respuesta del sistema**
1. Este caso de uso comienza cuando el usuario ingresa los datos de un nuevo proyecto y presiona sobre el botón crear	2. Verifica que el usuario haya iniciado sesión y que el campo de nombre de proyecto no se encuentre vacío
	3. Crear un nuevo proyecto con los datos proporcionados por el usuario y una lista de tareas vacía, en la que se ingresarán las tareas del proyecto
	4. Crea una lista de tareas vacías relacionada con el nuevo proyecto
	5. Adhiere una nueva fila con el proyecto a la tabla de proyectos y la muestra en pantalla
Cursos alternos: Línea 1: no se ha iniciado sesión. Indica el error Línea 1: se encuentra el campo nombre vació. Indica el error	

Fuente: elaboración propia.

Tabla XIX. **Caso de uso expandido, abrir proyecto**

No.	4
Caso de uso	Abrir proyecto
Actores	Usuario(iniciador)
Propósito	Abrir un proyecto para administrarlo
Resumen	El usuario solicita abrir un proyecto, el sistema lo busca,así como las tareas asociadas al proyecto en el Datastore para mostrarlas al usuario
Tipo	Primario y esencial
Referencias cruzadas	R1.8, R5.3

Curso normal de los eventos; acción del usuario	**Respuesta del sistema**
1. Comienza cuando el usuario selecciona un proyecto	2. Muestra las opciones de abrir o eliminar
3. Selecciona abrir	4. Crea una nueva pestaña con el nombre del proyecto
	5. Lista las tareas del proyecto
6. Selecciona la pestaña del proyecto abierto	

Cursos alternos:

Línea 1: selecciona eliminar. Continúa en el paso 4 de 1.5

Línea 2: no hay proyecto seleccionado. Indica el error y termina el caso de uso

Línea 2: el proyecto ya fue abierto. Indica mensaje y termina el caso de uso

Línea 3: no hay tareas en el proyecto, muestra mensaje y continúa

Fuente: elaboración propia.

Tabla XX. **Caso de uso expandido, eliminar proyecto**

No.	5
Caso de uso	Eliminar proyecto
Actores	Usuario(iniciador)
Propósito	Eliminar un proyecto y sus tareas
Resumen	El usuario solicita eliminar un proyecto, el sistema busca el proyecto seleccionado en el Datastore y lo elimina
Tipo	Primario y esencial
Referencias cruzadas	R1.8, R4.3

Curso normal de los eventos acción del usuario	**Respuesta del sistema**
1. Comienza cuando el usuario selecciona un proyecto	2. Muestra las opciones de abrir o eliminar
3. Selecciona eliminar	4. Pregunta si está seguro de querer eliminar el proyecto
5. Selecciona “Sí”	6. Busca el proyecto en la Datastore y lo eliminar
	7. Busca las tareas relacionadas en el Datastore y las eliminar
	8. Busca el proyecto en la tabla de proyectos y lo elimina
	9. Redibuja la tabla de proyectos

Cursos alternos:

Línea 1: selecciona abrir. Continúa en el paso 4 de 1.4

Línea 3: selecciona “No”. Termina el caso de uso

Fuente: elaboración propia.

Tabla XXI. **Caso de uso expandido, cerrar proyecto**

No.	6	
Caso de uso	Cerrar proyecto	
Actores	Usuario(iniciador)	
Propósito	Cerrar un proyecto y sus tareas	
Resumen	El usuario solicita cerrar un proyecto, el sistema le advierte que se perderán los cambios no guardados y cierra el proyecto si el usuario no desea dar marcha atrás	
Tipo	Primario y esencial	
Referencias cruzadas	R1.8	
Curso normal de los eventos acción del usuario		**Respuesta del sistema**
1. El usuario se encuentra administrando un proyecto y presiona el botón “Cerrar proyecto”		2. Pregunta si está seguro de querer cerrar el proyecto e indica que no se guardarán los cambios
3. Presiona “Si”		4. Elimina los últimos cambios y la pestaña del proyecto
•		5. Regresa al administrador de proyectos
Cursos alternos: Línea 2: el usuario presiona “No”. Termina el caso de uso		

Fuente: elaboración propia.

Tabla XXII. **Caso de uso expandido, guardar proyecto**

No.	7
Caso de uso	Guardar proyecto
Actores	Usuario(iniciador)
Propósito	Guardar un proyecto y sus tareas
Resumen	El usuario solicita guardar un proyecto, el sistema lo guarda, así como las tareas relacionadas con el mismo
Tipo	Primario y esencial
Referencias cruzadas	R1.8

Curso normal de los eventos; acción del usuario	**Respuesta del sistema**
1. Cuando el usuario se encuentra administrando un proyecto y presiona el botón Guardar proyecto	2. Busca el proyecto en el Datastore y lo actualiza; luego busca las tareas relacionadas con el proyecto, las actualiza y crea las tareas que no existían antes
	3. Despliega el mensaje de guardado exitoso

Cursos alternos:

Línea 1: existe un fallo en la conexión con el Datastore. Indica el error y termina el caso de uso

Fuente: elaboración propia.

3.2. Diseño de arquitectura

En el diseño de arquitectura se define el modelo de capas, así también se muestran los diagramas de flujo, estado, secuencias, clases, el modelo entidad relación a implementar en la aplicación.

3.2.1. Capas a utilizar

A continuación se presenta la definición de cada una de las capas a utilizaren la implementación del proyecto y la justificación del porqué la selección de cada una de ellas y las tecnologías relacionadas que las conforman.

3.2.2. Capa de presentación

La capa de *Frontend*, contiene todos los componentes que se relacionan con la interacción del usuario final en las maquinas clientes. A esta capa el usuario accede a través de un navegador Web, que le permita visualizar aplicaciones que utilicen RIA.

3.2.3. Justificación de capa de presentación

PlanninGt, como se ha nombrado al sistema Web, debido a que se desarrolla la gestión de proyectos individuales, administradores de proyectos, harán uso del mismo; esta capa es la más cercana del usuario, es por ello su gran importancia, ya que se debe velar porque cumpla con los requerimientos de usuario y principalmente que esta sea funcional e intuitiva.

Debido a que está desarrollando una aplicación basada en RIA, utilizando el *framework* GWT, se puede optar a construir una aplicación que aproveche los beneficios de la tecnología AJAX, que no necesite recargar el navegador para poder ser utilizada, tal cual una aplicación de escritorio.

3.2.4. Capa de negocio

En esta capa se ejecutan las reglas de negocio, en las cuales aparecen las validaciones, restricciones, protocolos, reglas y requerimientos no funcionales. Aquí se encuentra el servidor de aplicaciones de GAE, que responde a llamadas RPC.

3.2.5. Justificación de capa de negocio

En esta capa se manejan las funciones relacionadas con la interacción de la capa de presentación, con las llamadas a procedimientos de los métodos relacionados con los servicios Web, el almacenamiento y la persistencia de datos. GWT trabaja con GAE para poderse comunicar vía RPC con el cliente, construyendo un canal de comunicación seguro entre cliente y servidor. Se hace uso de estas tecnologías debido a que se está construyendo un sistema nuevo; estale brinda la posibilidad de crear aplicaciones fáciles de gestionar y escalar.

3.2.6. Capa de datos

También llamada capa de *Backend,* es la encargada de conjuntar los elementos dedicados a la persistencia del sistema.

En esta capa se encuentra el Datastore de GAE, y todos los componentes abstraídos en su servicio de nube como IaaS, donde se encuentra la *Bigtable*, balanceo de carga y otros servicios de virtualización por *hardware*.

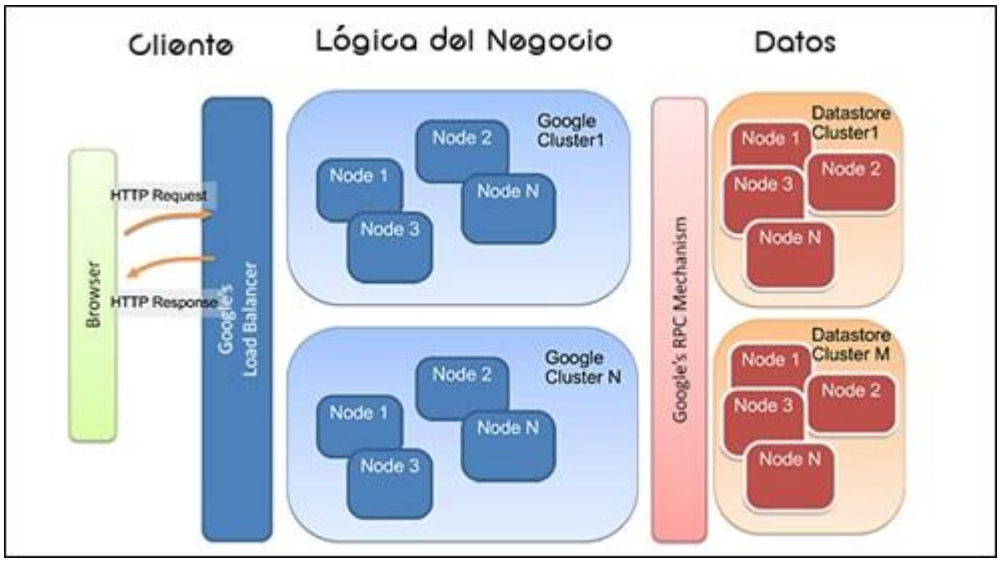
3.2.7. Justificación de capa de datos

La capa de almacenamiento de datos es una de las más importantes, ya que la información es uno de los activos imprescindibles en cualquier organización y los datos de los clientes deben de ser salvaguardados con recelo, ya que los usuarios requerirán de su información, en cualquier momento; la información debe ser protegida para garantizar su persistencia, integridad, confiabilidad, precisión y disponibilidad.

Para esta capa, se ha escogido el Datastore, almacén de datos provisto por GAE, ya que cuenta con soporte para aplicaciones Java, con soporte del API de JDO, así también les ofrece seguridad en el resguardo de los datos y debido a que utiliza escalabilidad horizontal (replicación),esto garantiza disponibilidad.

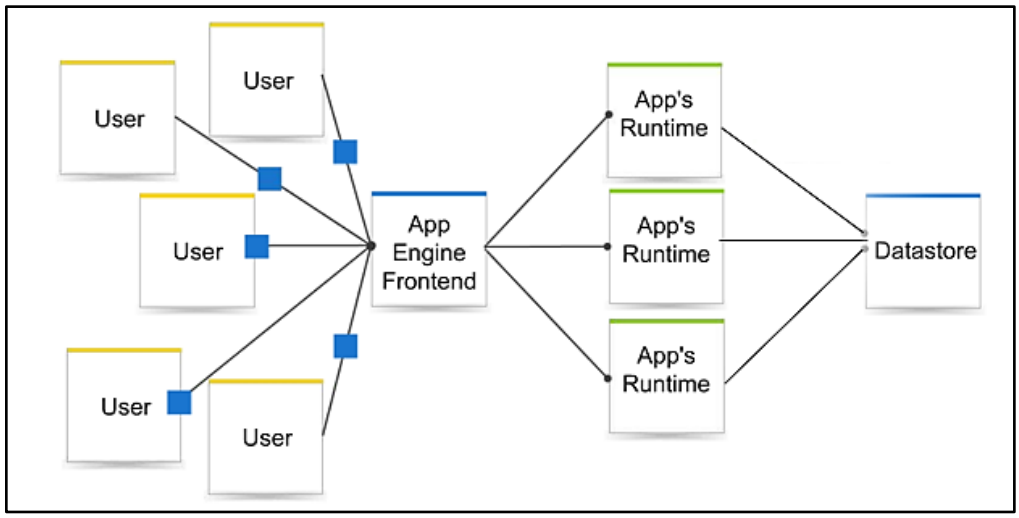
En la figura 23, se muestra el diagrama de capas de la aplicación, donde en el lado del cliente se encuentra el servicio de balanceo de carga; en la lógica del negocio funciona la computación en clúster y el Datastore en la capa de almacenamiento de datos.

Figura 23. **Diagrama de capas de la aplicación**



Fuente: editado de RAJDEPP, 2008.

Figura 24. **Descripción del acceso al sistema**

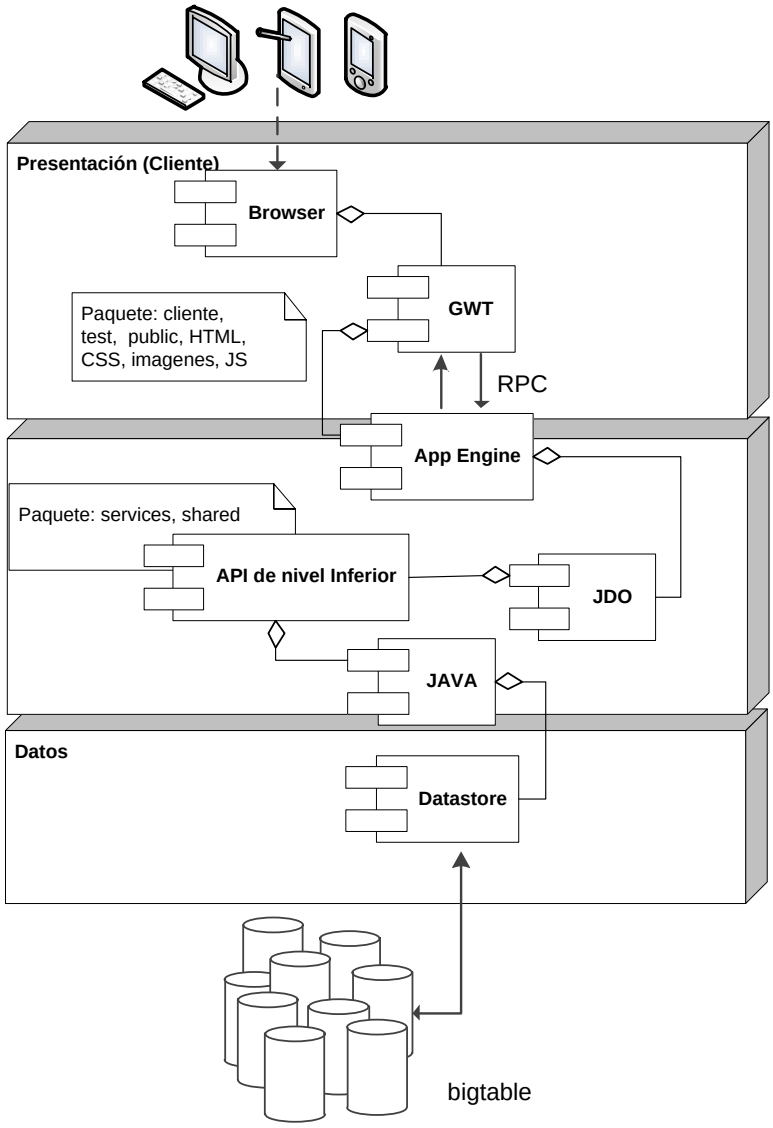


Fuente: editado de RAJDEPP, 2008.

3.3. Diseño de capas

En el siguiente diagrama se puede observar la distribución de los paquetes en cada una de las capas del sistema.

Figura 25. **Diagrama de arquitectura del sistema**

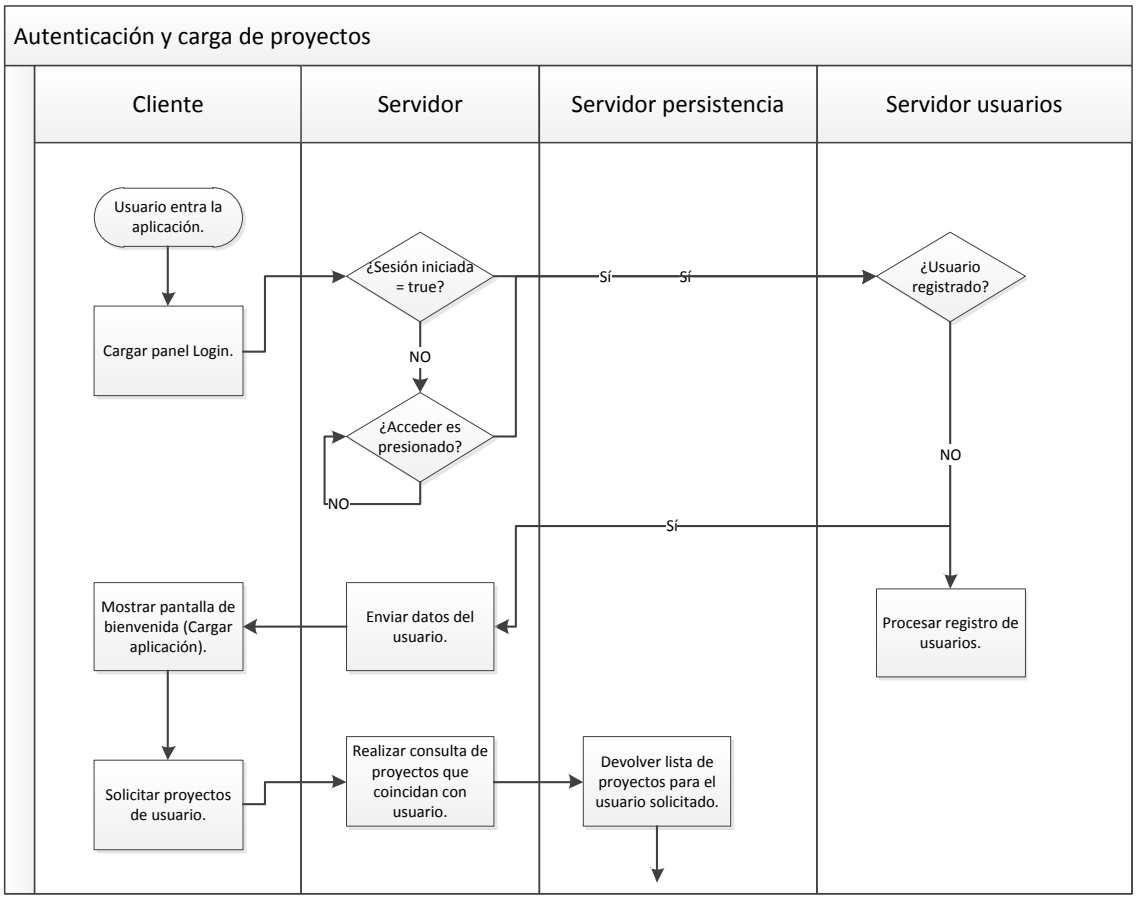


Fuente: elaboración propia.

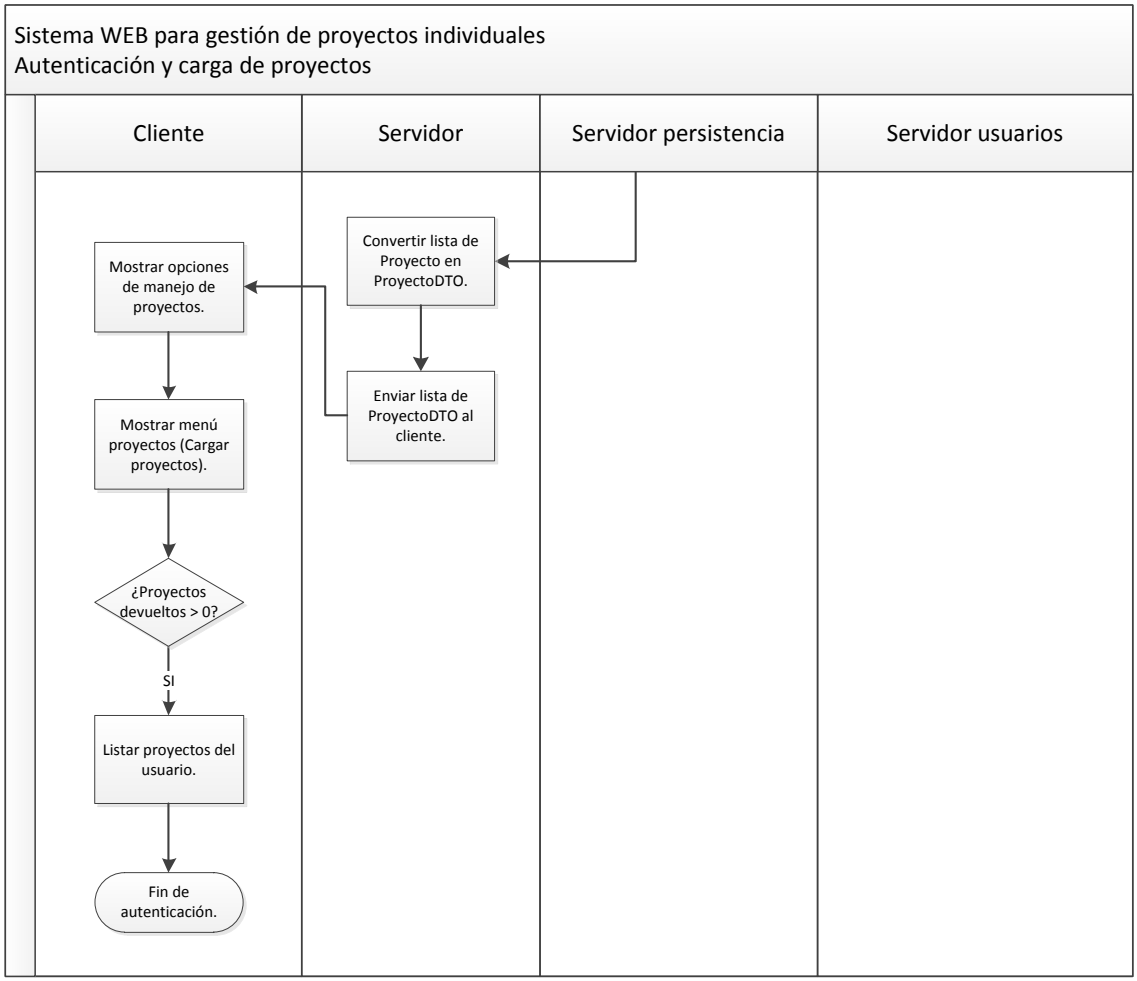
3.4. Diagramas de flujo

Los diagramas de flujo son la representación gráfica de un algoritmo, que muestra el comportamiento de un caso de uso de la aplicación.

Figura 26. **Diagrama de flujo, autenticación y carga de proyectos**

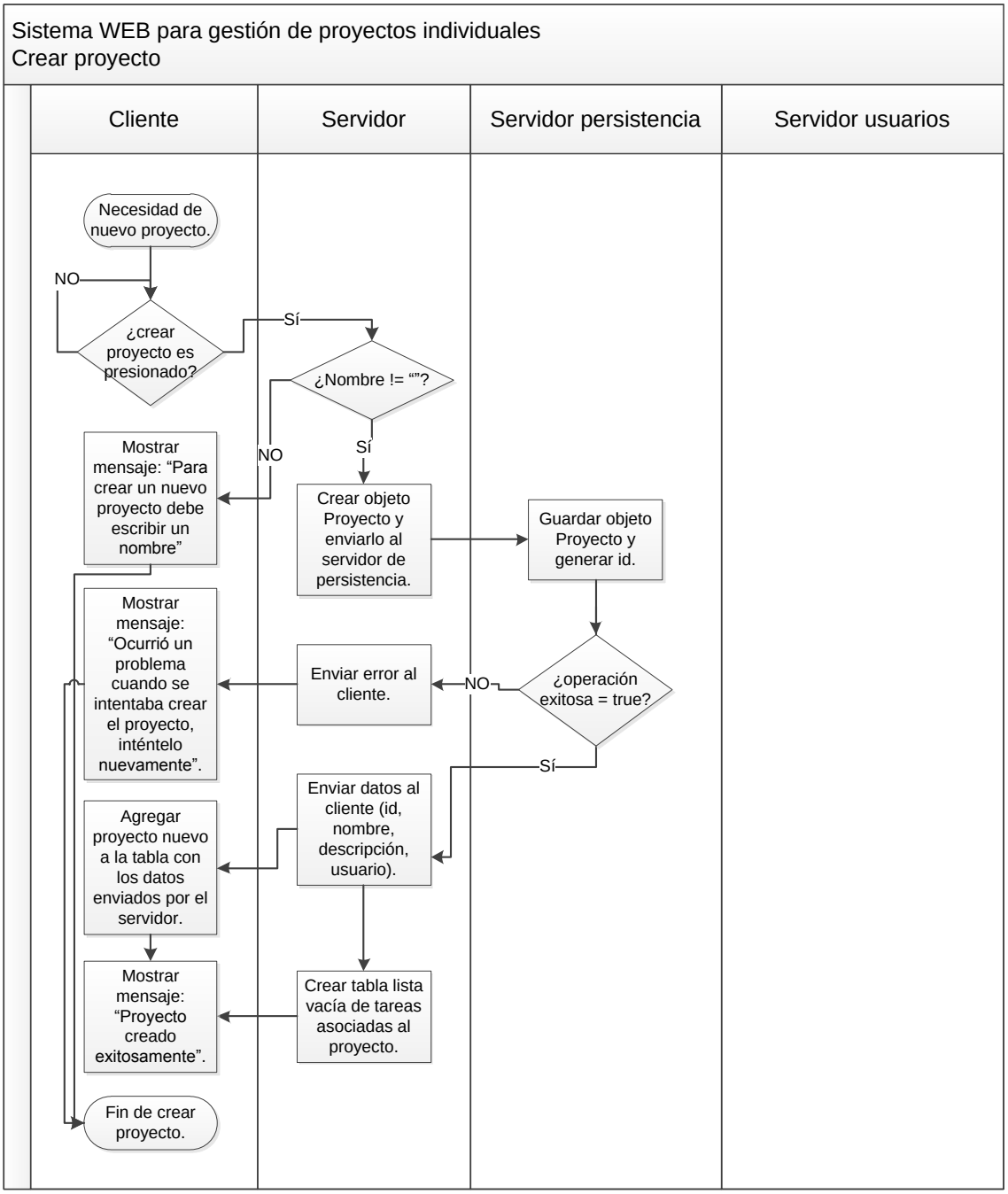


Continuación figura 26.



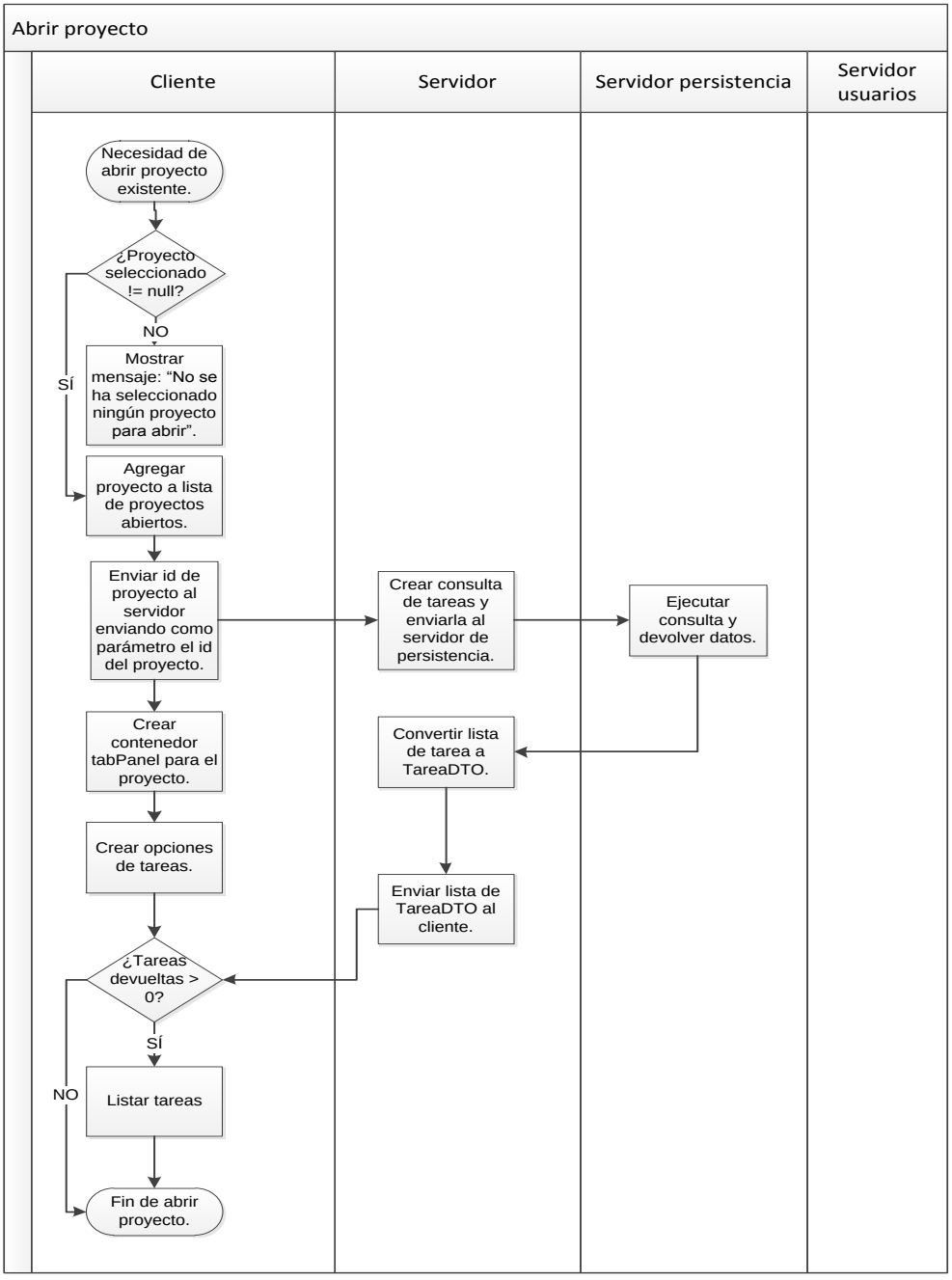
Fuente: elaboración propia.

Figura 27. **Diagrama de flujo, crear proyecto**



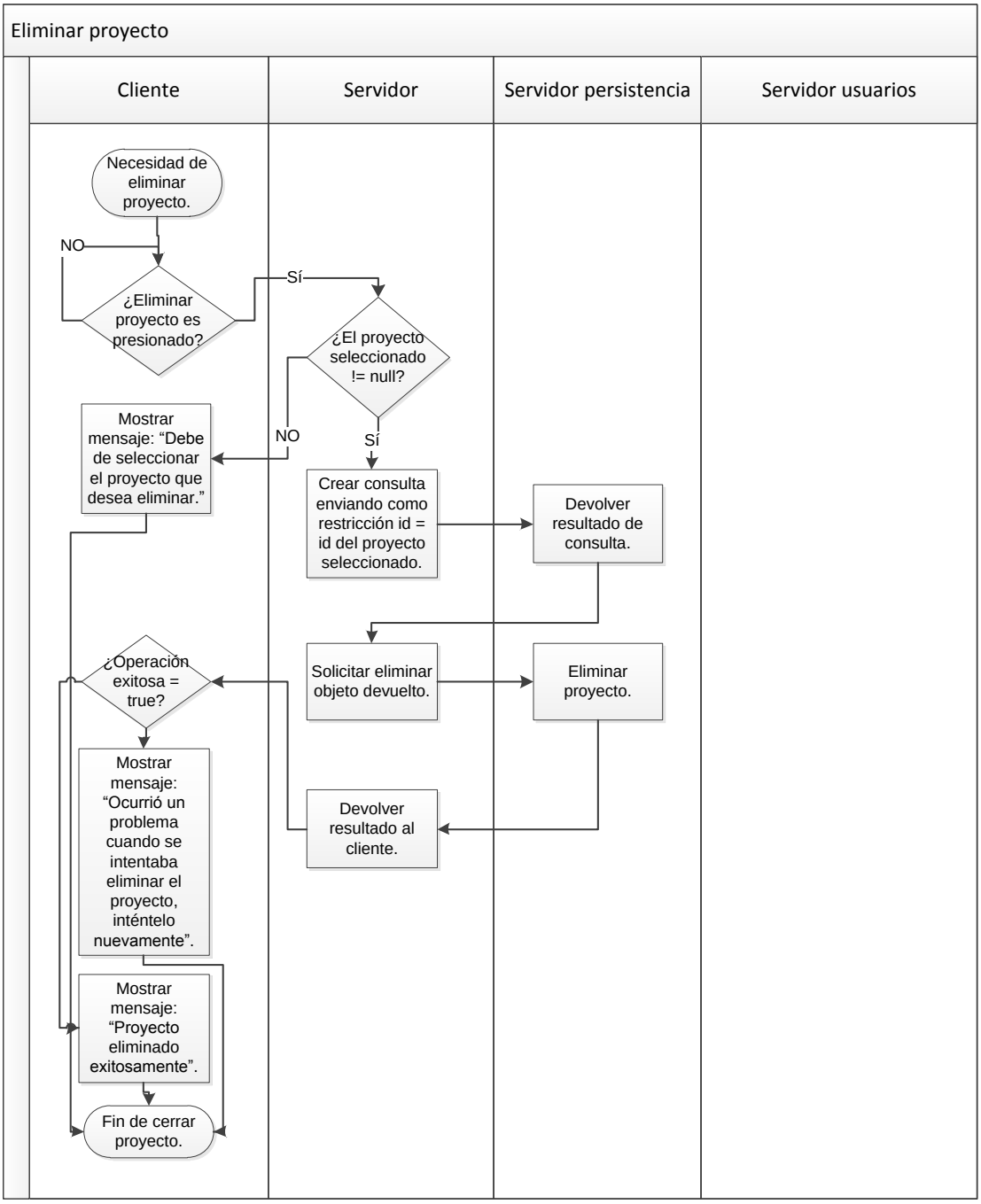
Fuente: elaboración propia.

Figura 28. **Diagrama de flujo, abrir proyecto**



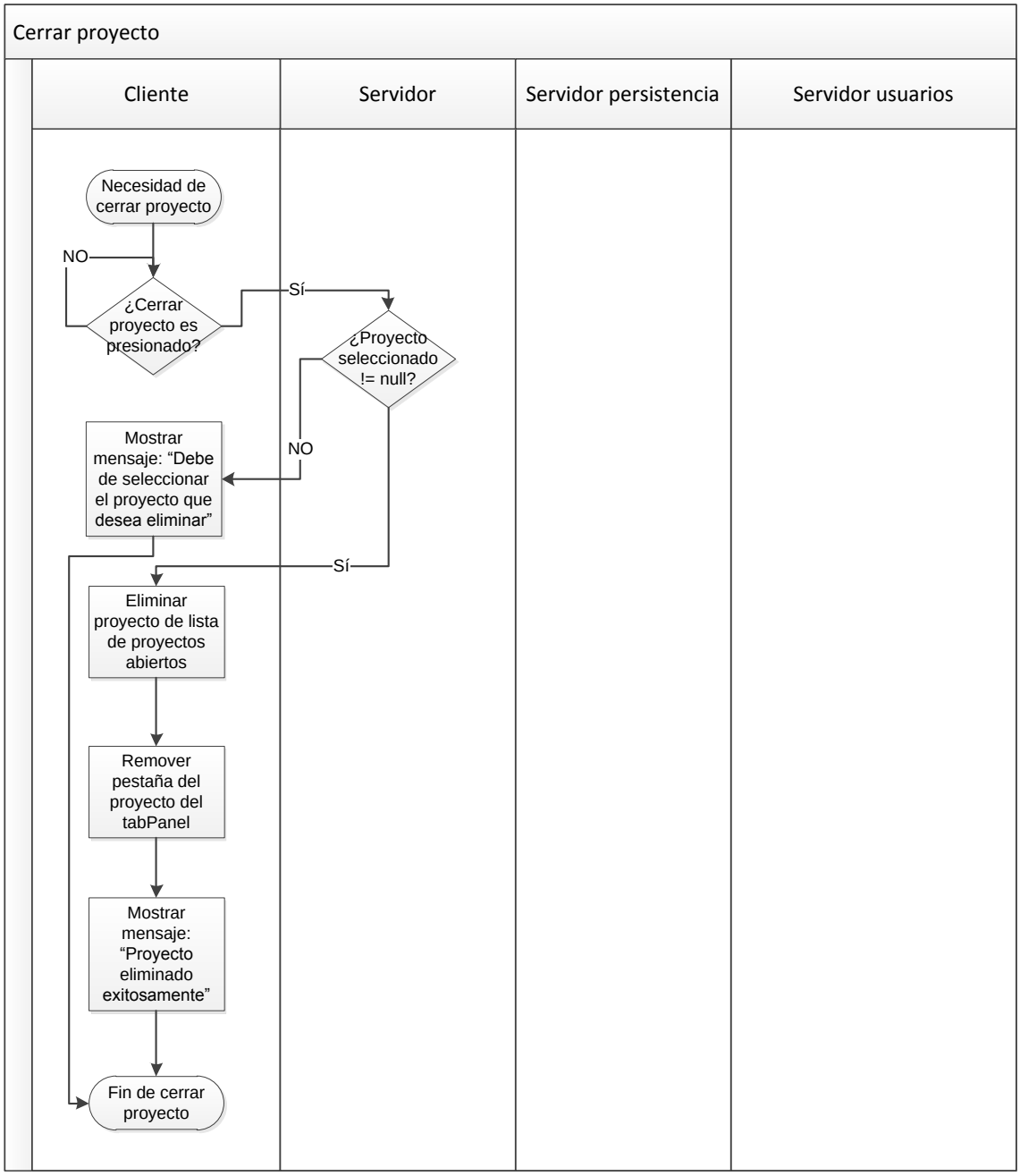
Fuente: elaboración propia.

Figura 29. **Diagrama de flujo, eliminar proyecto**



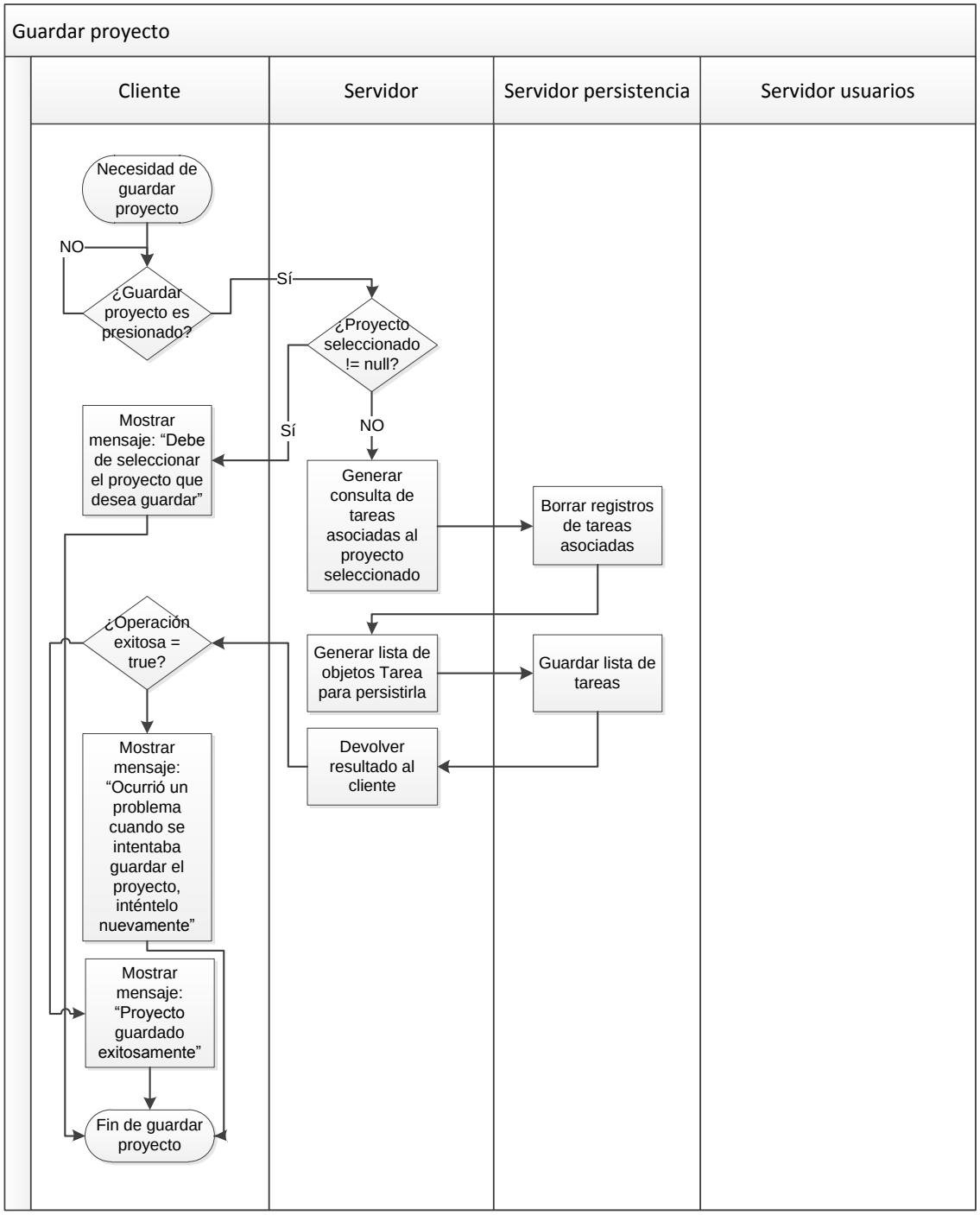
Fuente: elaboración propia.

Figura 30. **Diagrama de flujo, cerrar proyecto**



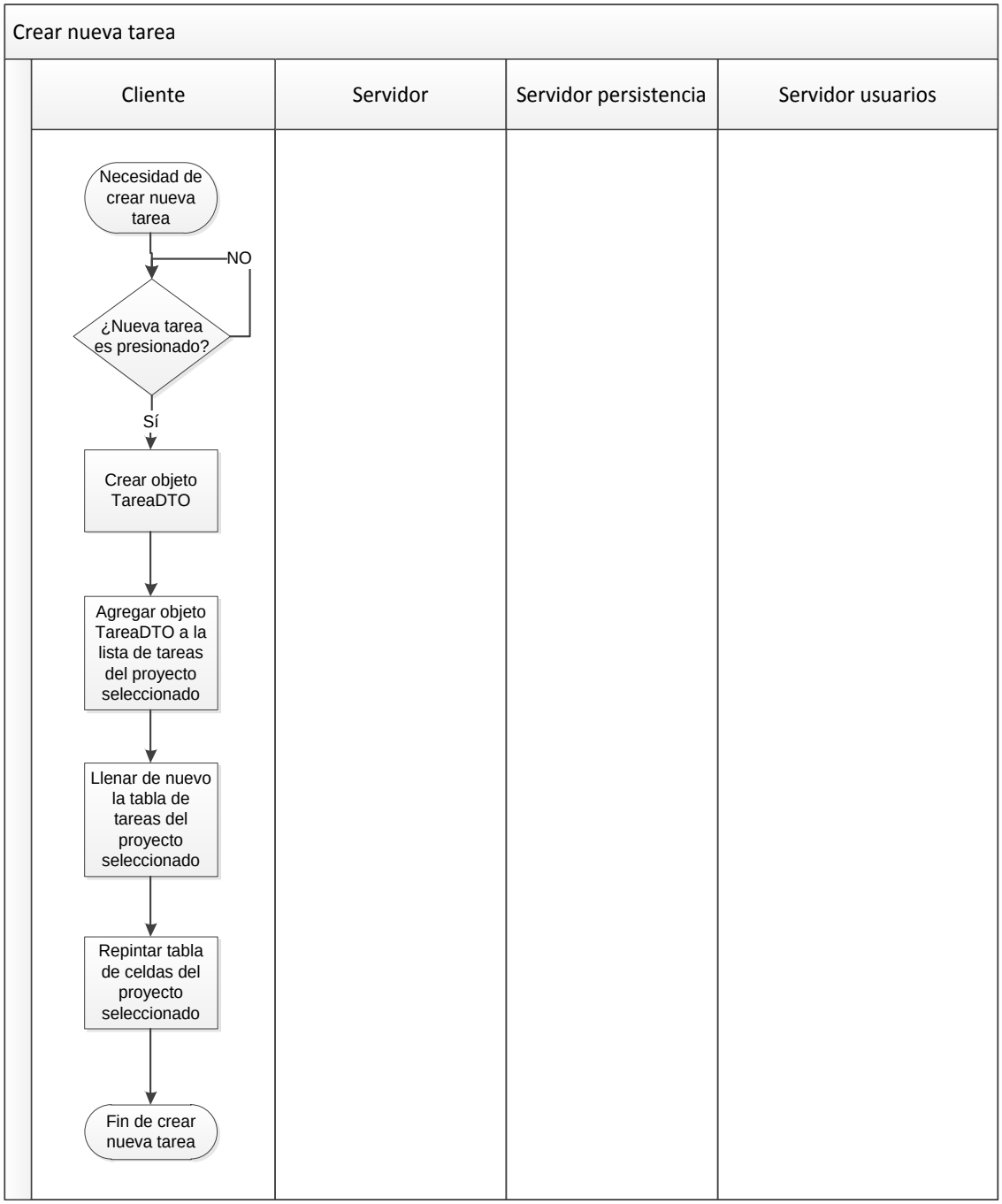
Fuente: elaboración propia.

Figura 31. **Diagrama de flujo, guardar proyecto**



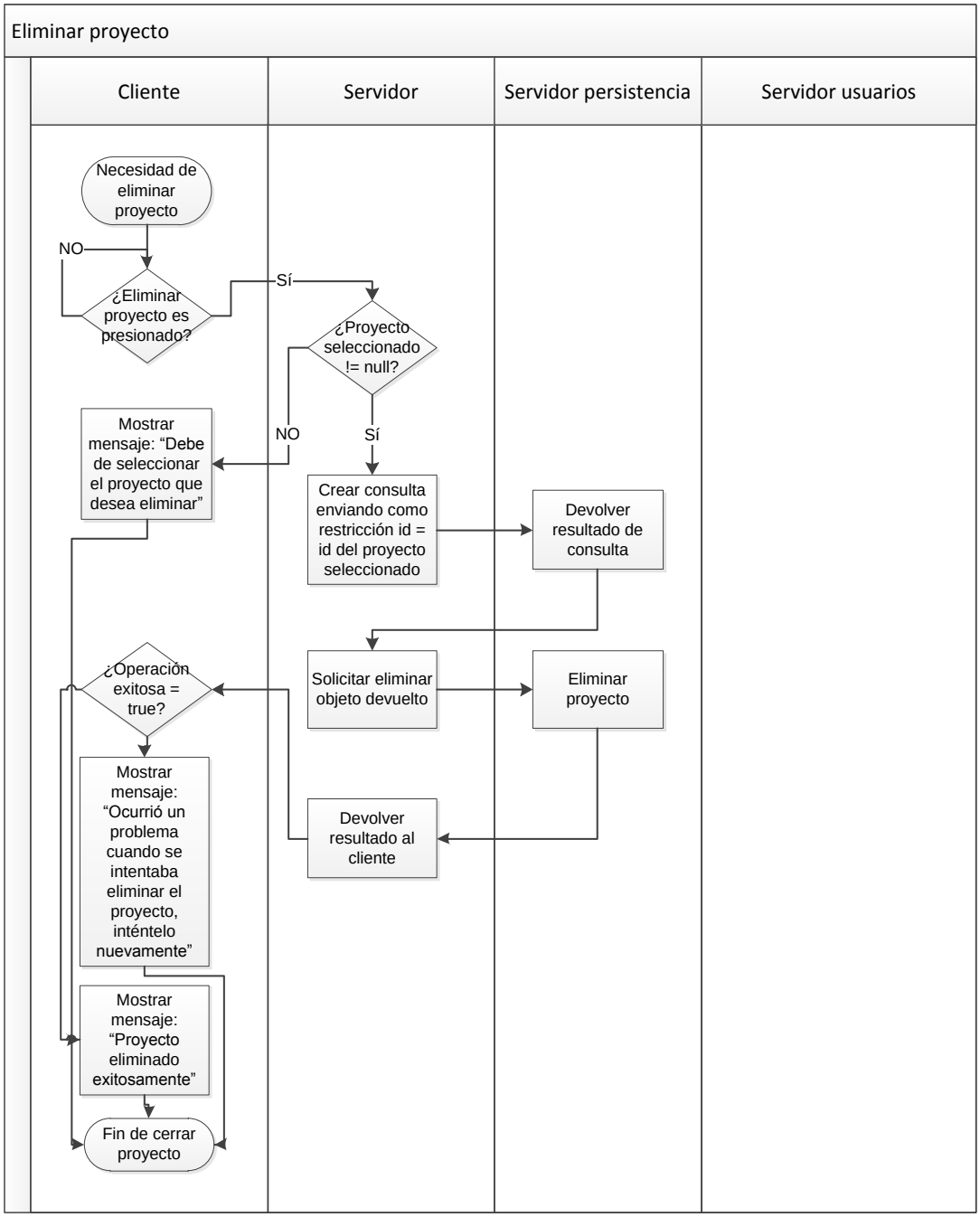
Fuente: elaboración propia.

Figura 32. **Diagrama de flujo, nueva tarea**



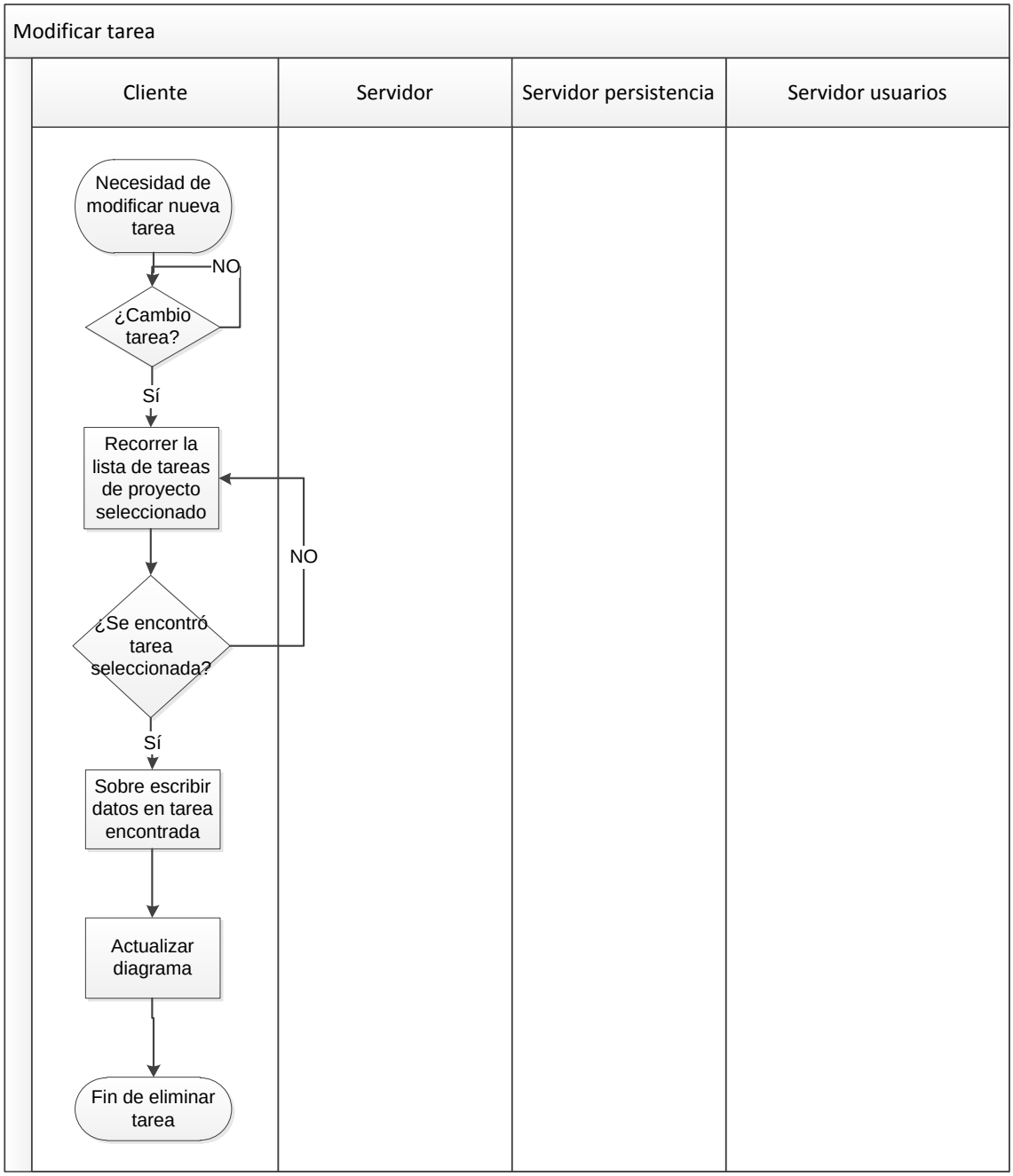
Fuente: elaboración propia.

Figura 33. **Diagrama de flujo, eliminar proyecto**



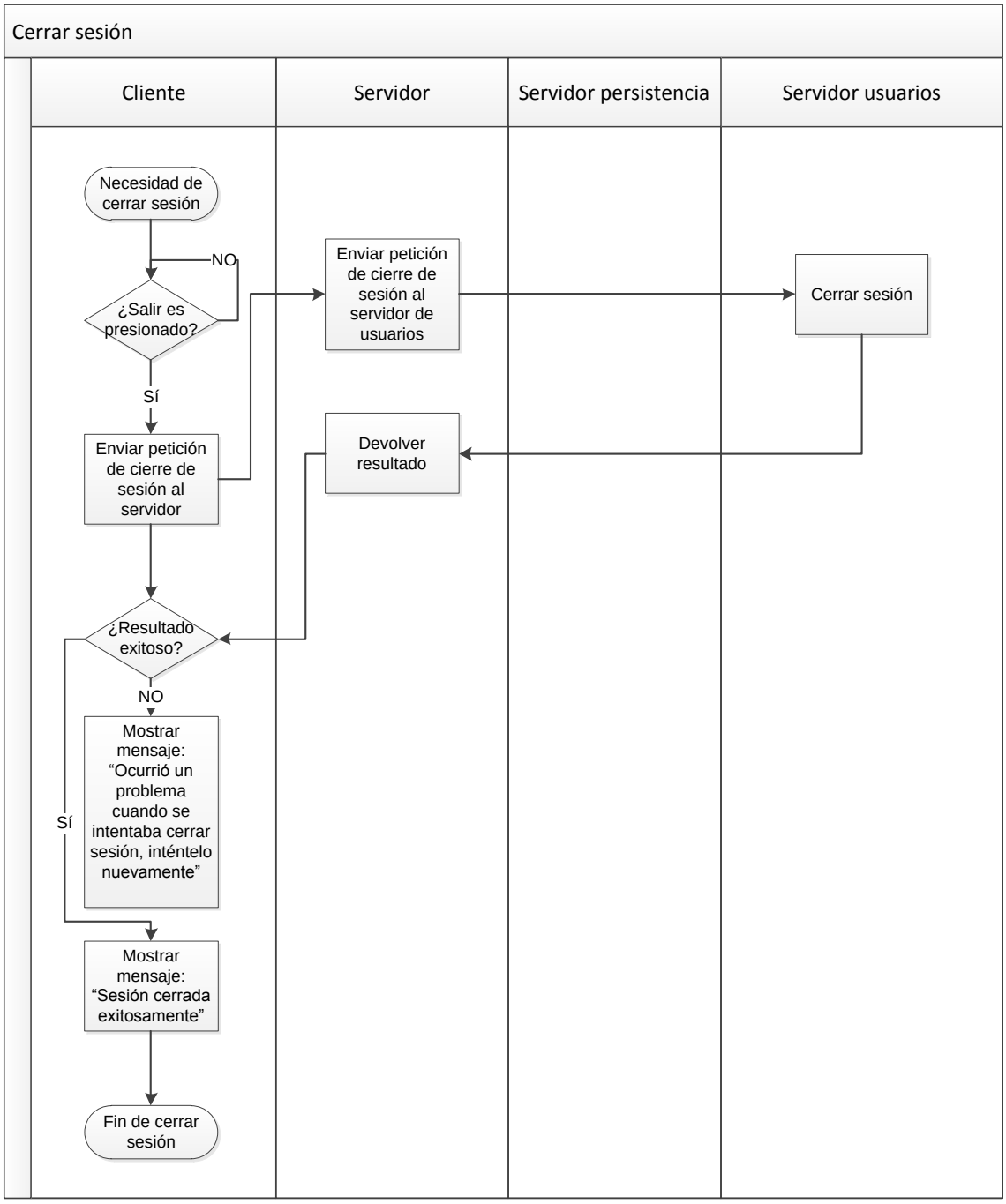
Fuente: elaboración propia.

Figura 34. **Diagrama de flujo, modificar tarea**



Fuente: elaboración propia.

Figura 35. **Diagrama de flujo, cerrar sesión**

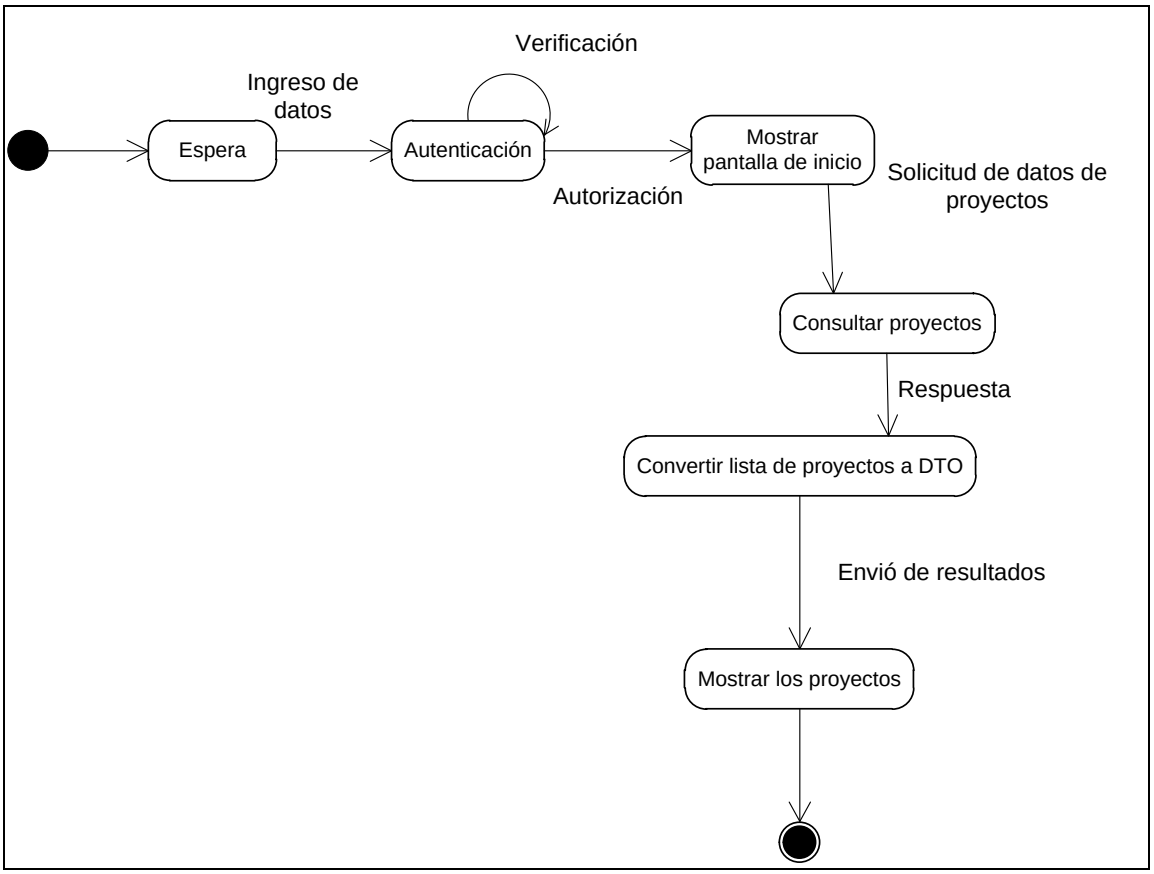


Fuente: elaboración propia.

3.5. Diagramas de estado

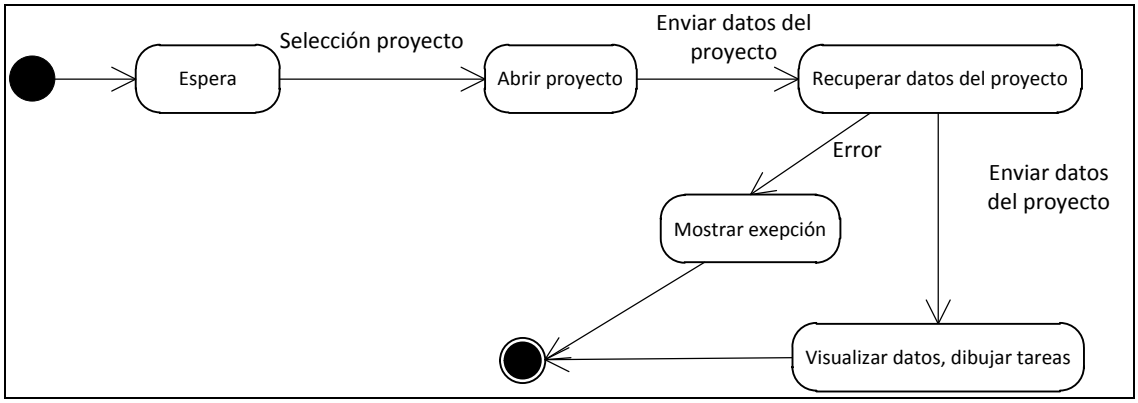
Los diagramas de estado que se presentan a continuación están orientados en la descripción de los cambios de estado en la interfaz gráfica de la aplicación Web.

Figura 36. **Diagrama estado, autenticarse**



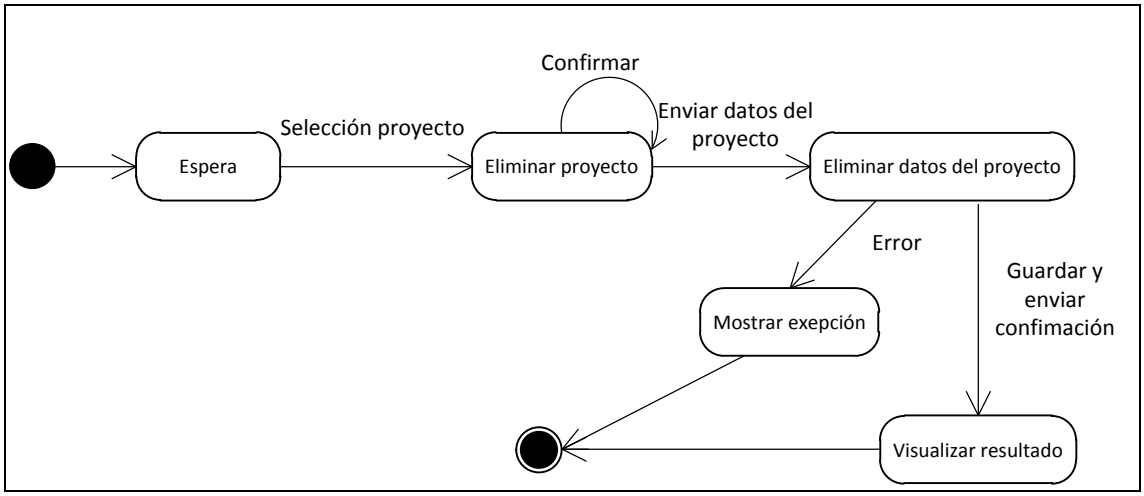
Fuente: elaboración propia.

Figura 37. **Diagrama estado, abrir proyecto**



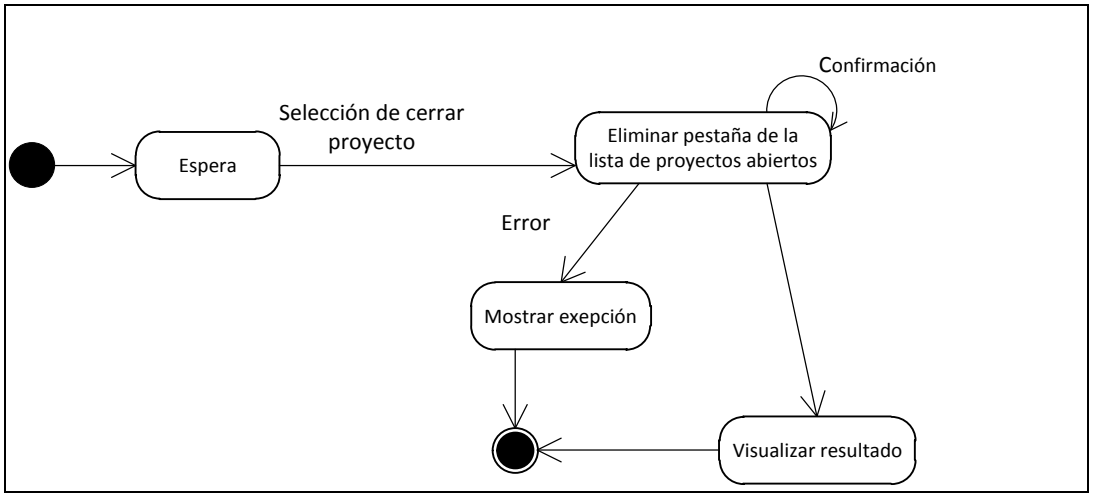
Fuente: elaboración propia

Figura 38. **Diagrama estado, eliminar proyecto**



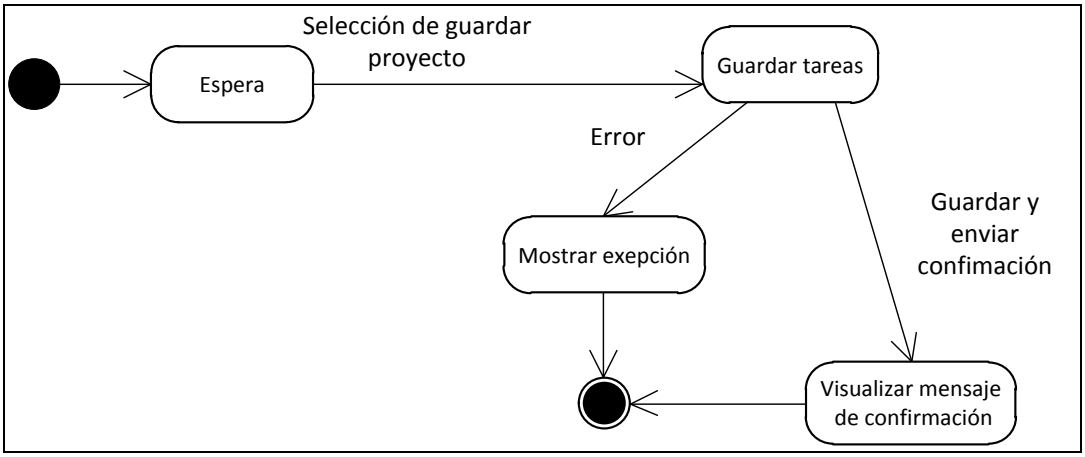
Fuente: elaboración propia.

Figura 39. **Diagrama estado, cerrar proyecto**



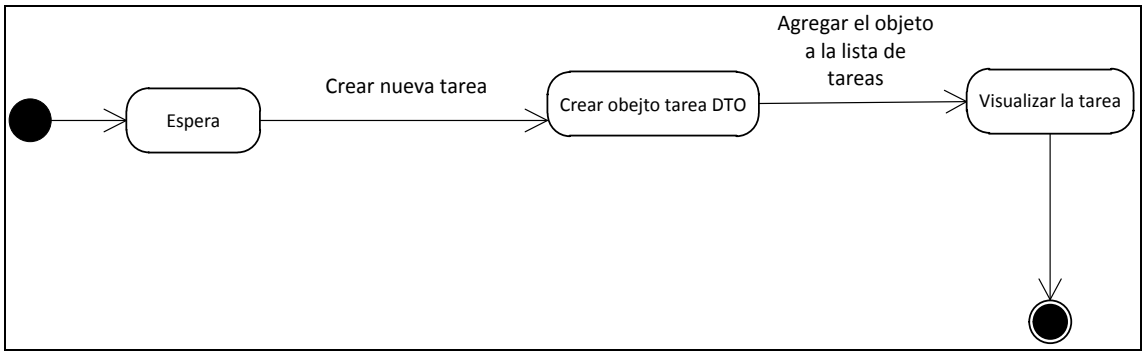
Fuente: elaboración propia.

Figura 40. **Diagrama estado, guardar proyecto**



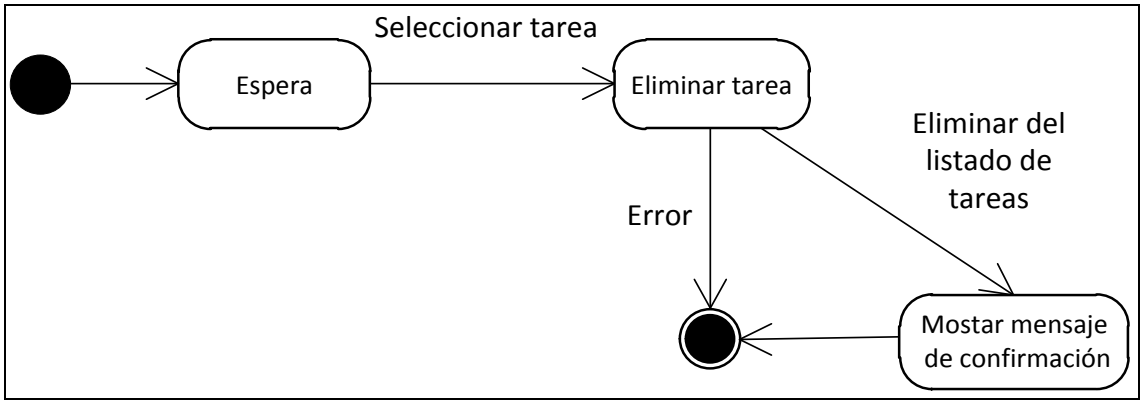
Fuente: elaboración propia.

Figura 41. **Diagrama estado, crear tarea**



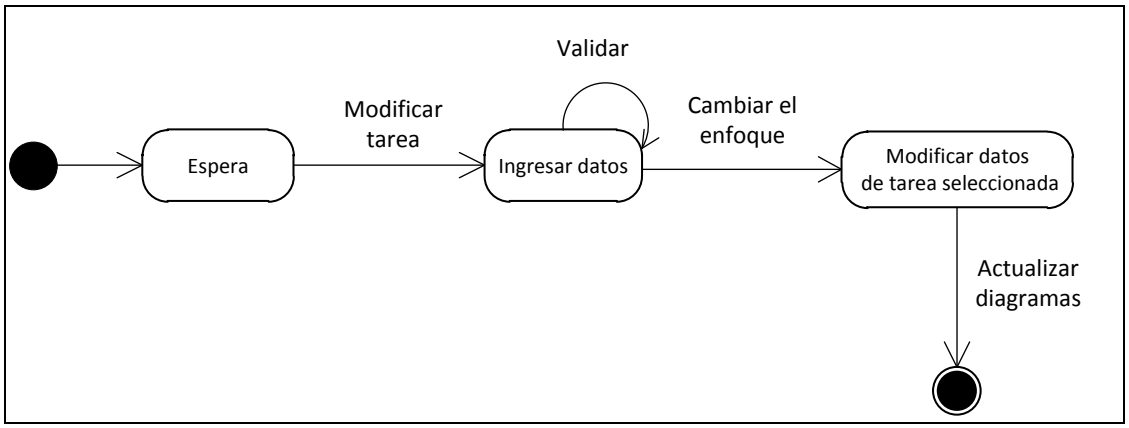
Fuente: elaboración propia.

Figura 42. **Diagrama estado, eliminar tarea**



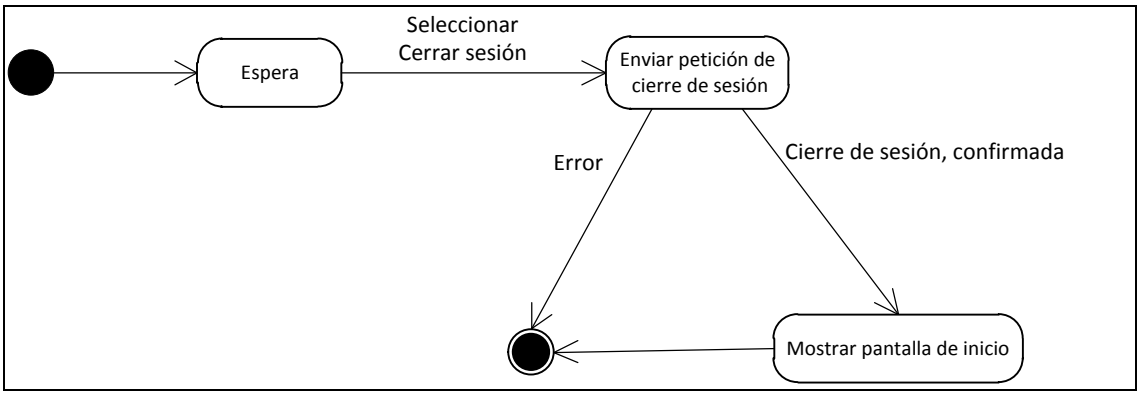
Fuente: elaboración propia.

Figura 43. **Diagrama estado, modificar tarea**



Fuente: elaboración propia.

Figura 44. **Diagrama estado, cerrar sesión**

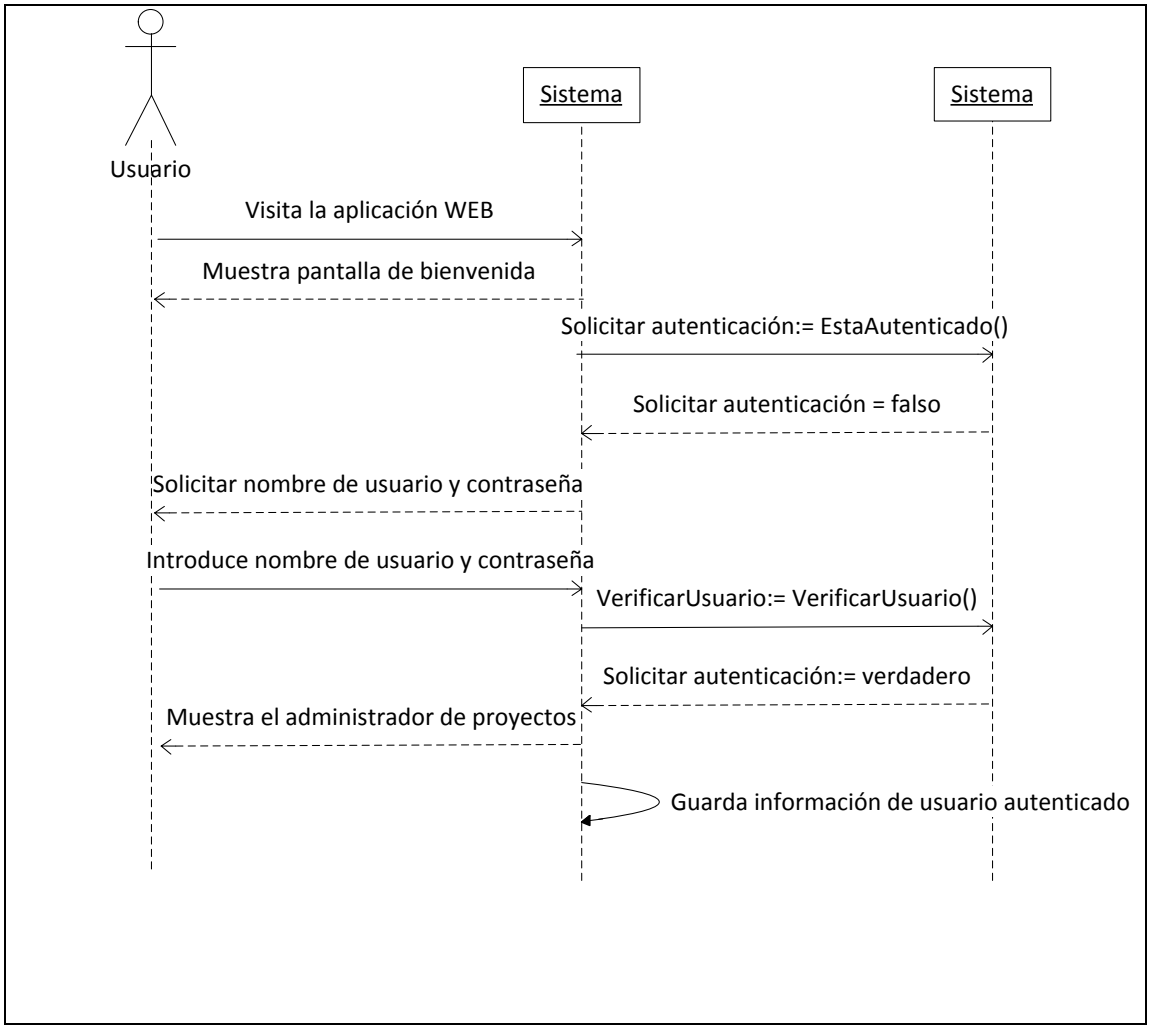


Fuente: elaboración propia.

3.6. Diagrama de secuencias

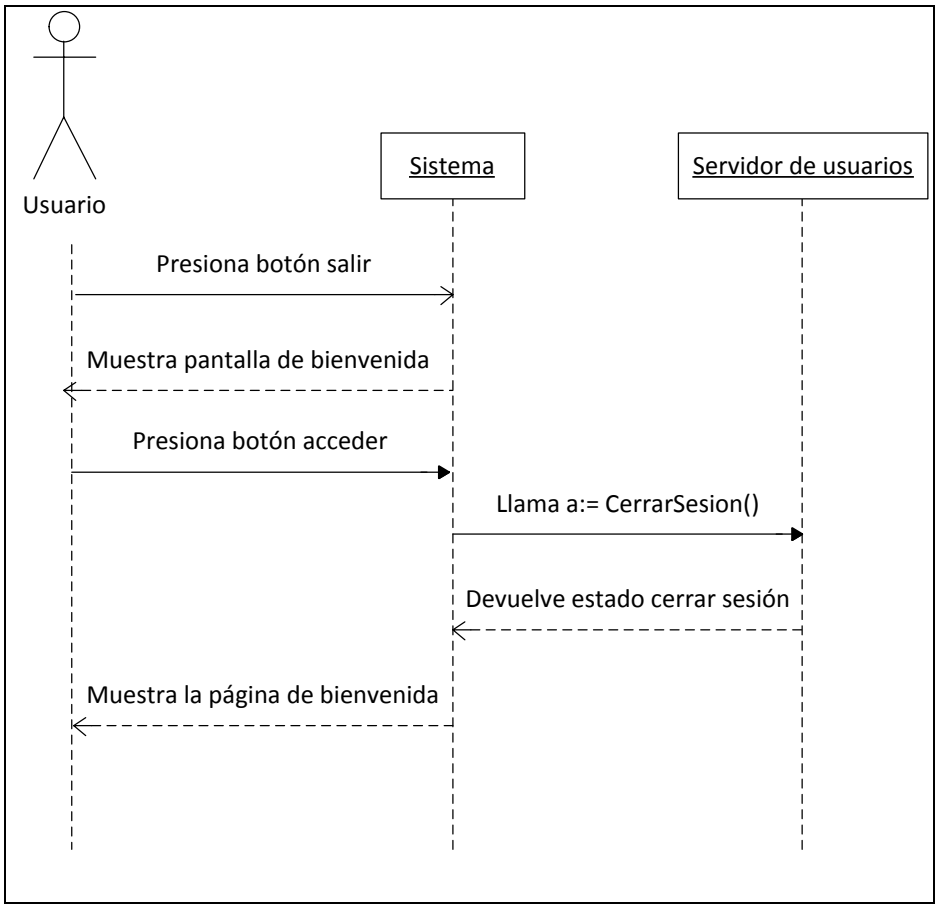
Estos diagramas ayudarán a la visualización de la interacción de los objetos involucrados en la aplicación a través del tiempo.

Figura 45. **Diagrama de secuencia, de autenticación**



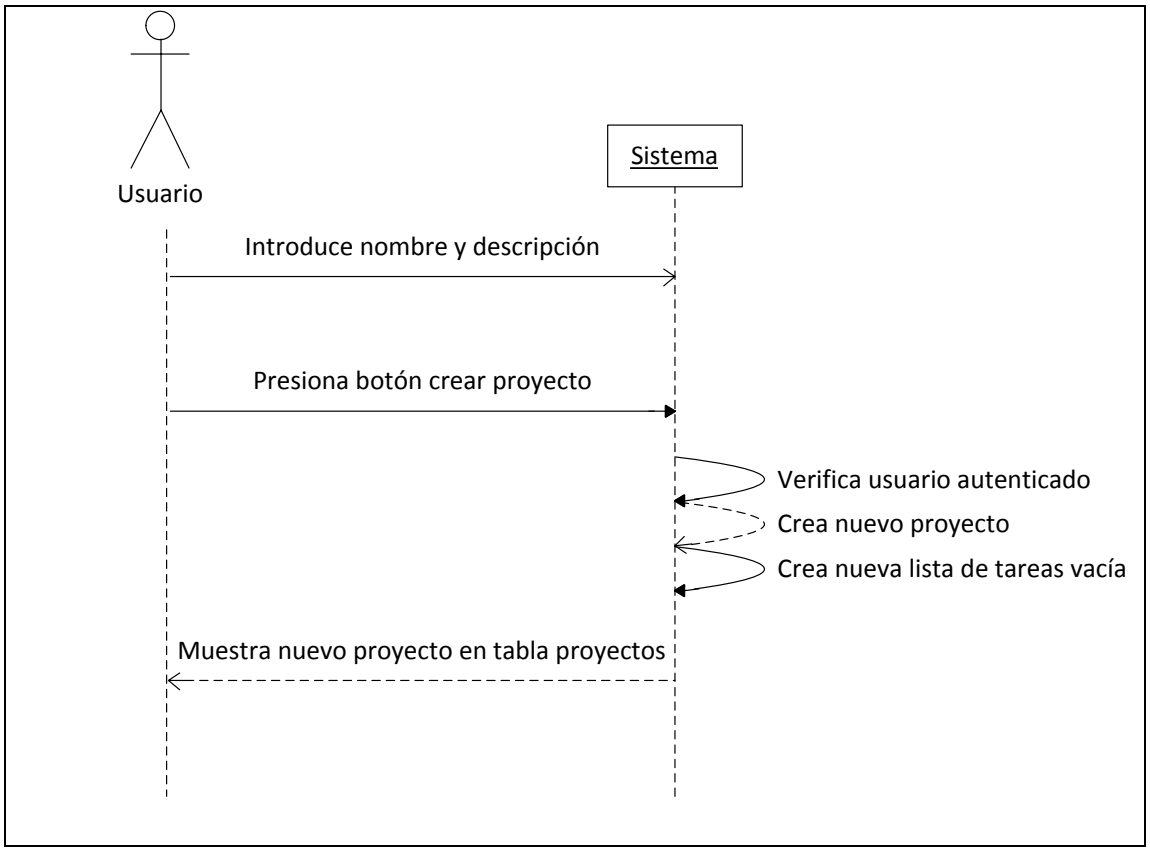
Fuente: elaboración propia.

Figura 46. **Diagrama de secuencias, de cerrar sesión**



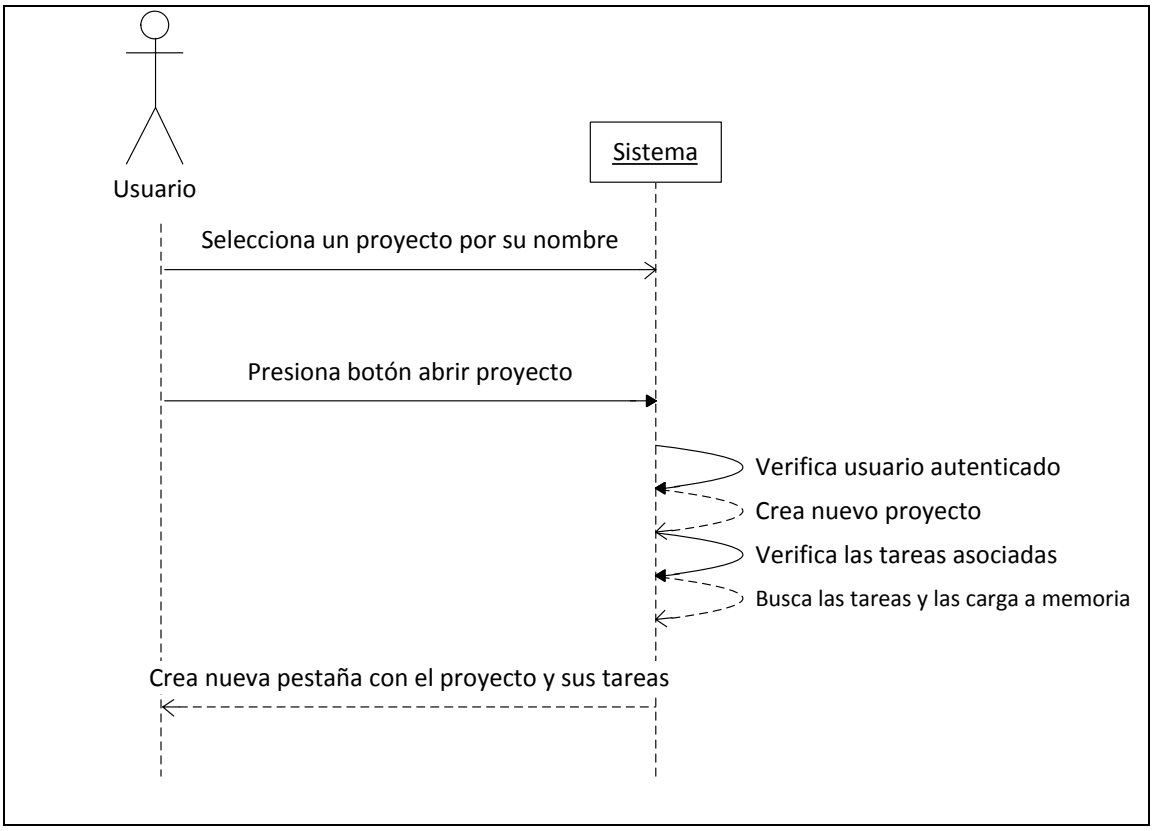
Fuente: elaboración propia.

Figura 47. **Diagrama de secuencias, crear proyecto**



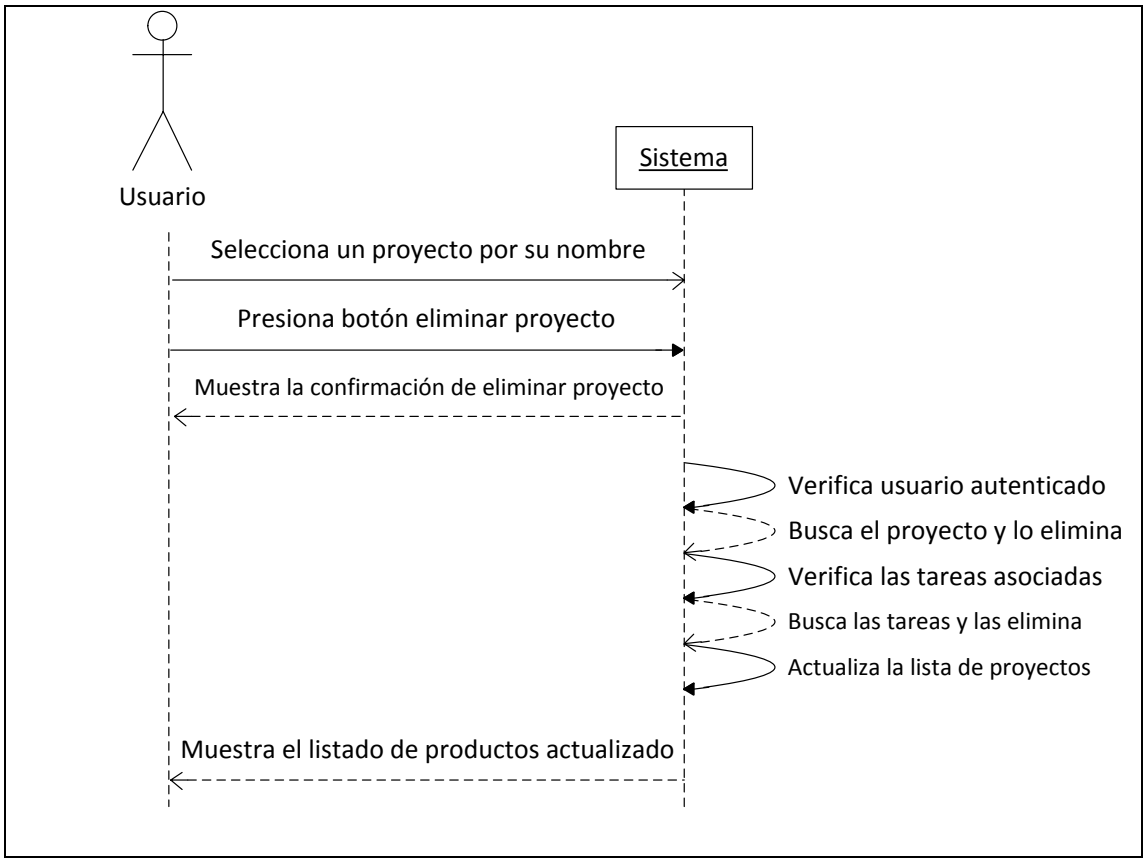
Fuente: elaboración propia.

Figura 48. **Diagrama de secuencias, abrir proyecto**



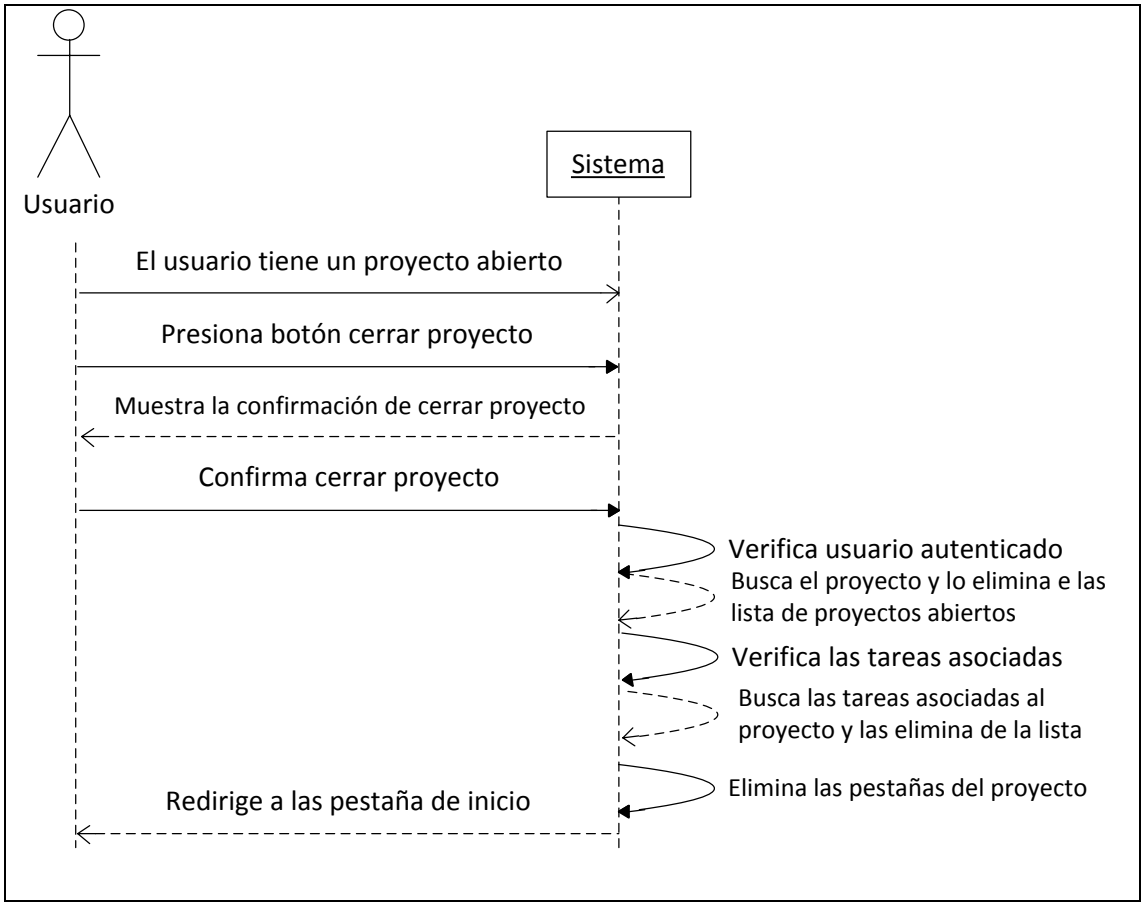
Fuente: elaboración propia.

Figura 49. **Diagrama de secuencias, eliminar proyecto**



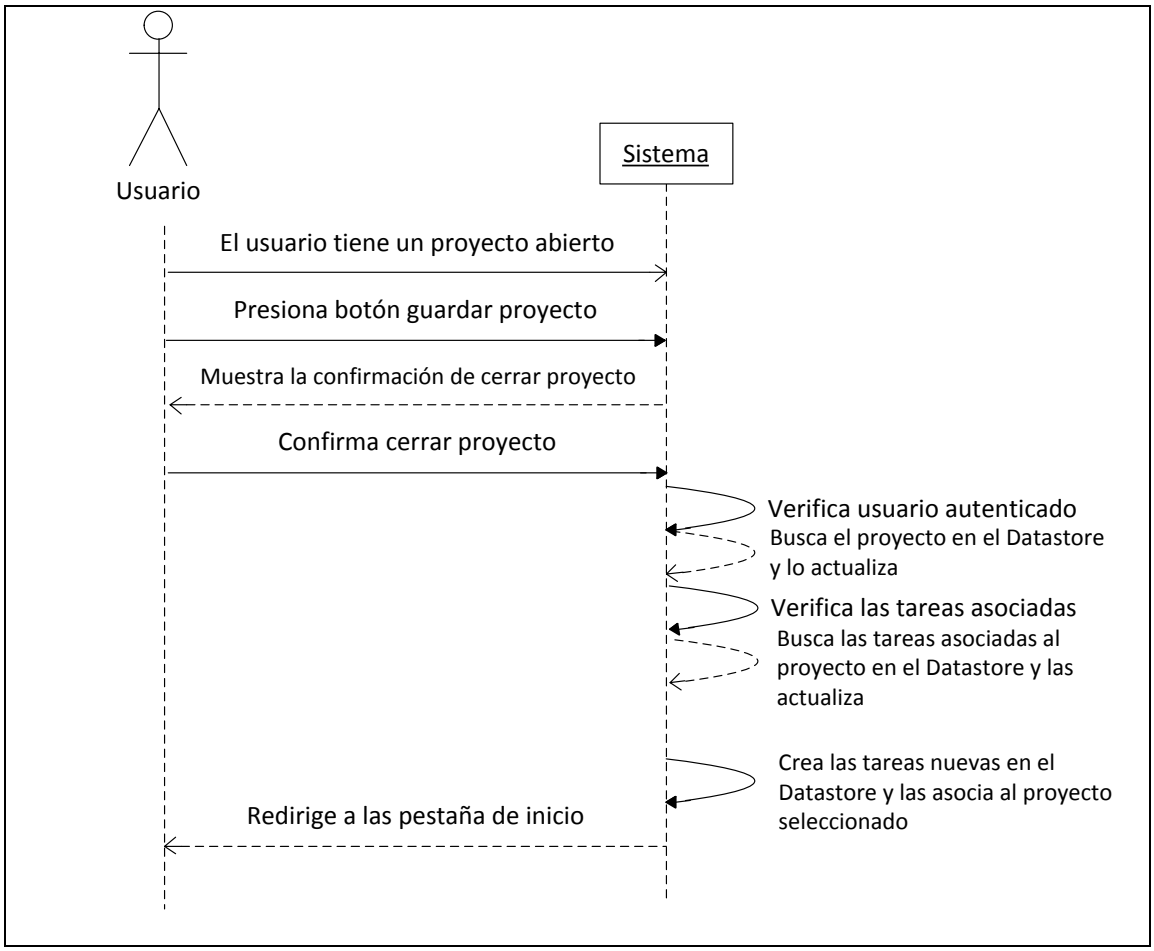
Fuente: elaboración propia.

Figura 50. **Diagrama de secuencias, cerrar proyecto**



Fuente: elaboración propia.

Figura 51. **Diagrama de secuencias, guardar proyecto**



Fuente: elaboración propia.

3.7. Diagrama de clases

En el siguiente diagrama, se muestran las clases divididas en sus respectivos paquetes tal como fue el resultado final de la implementación con GWT; el paquete *client* contiene las clases que se muestran en la capa de presentación y los *servlets* que se ejecutan en el navegador; en *shared* y *server* se encuentran las clases que se ejecutan en la capa de la lógica del negocio y la capa de datos.

A continuación se hace una breve descripción de las clases importantes utilizadas en el desarrollo de la aplicación:

- Plannigt: contiene toda la lógica de la aplicación, acá están las tareas descritas en los diagramas de casos de uso.

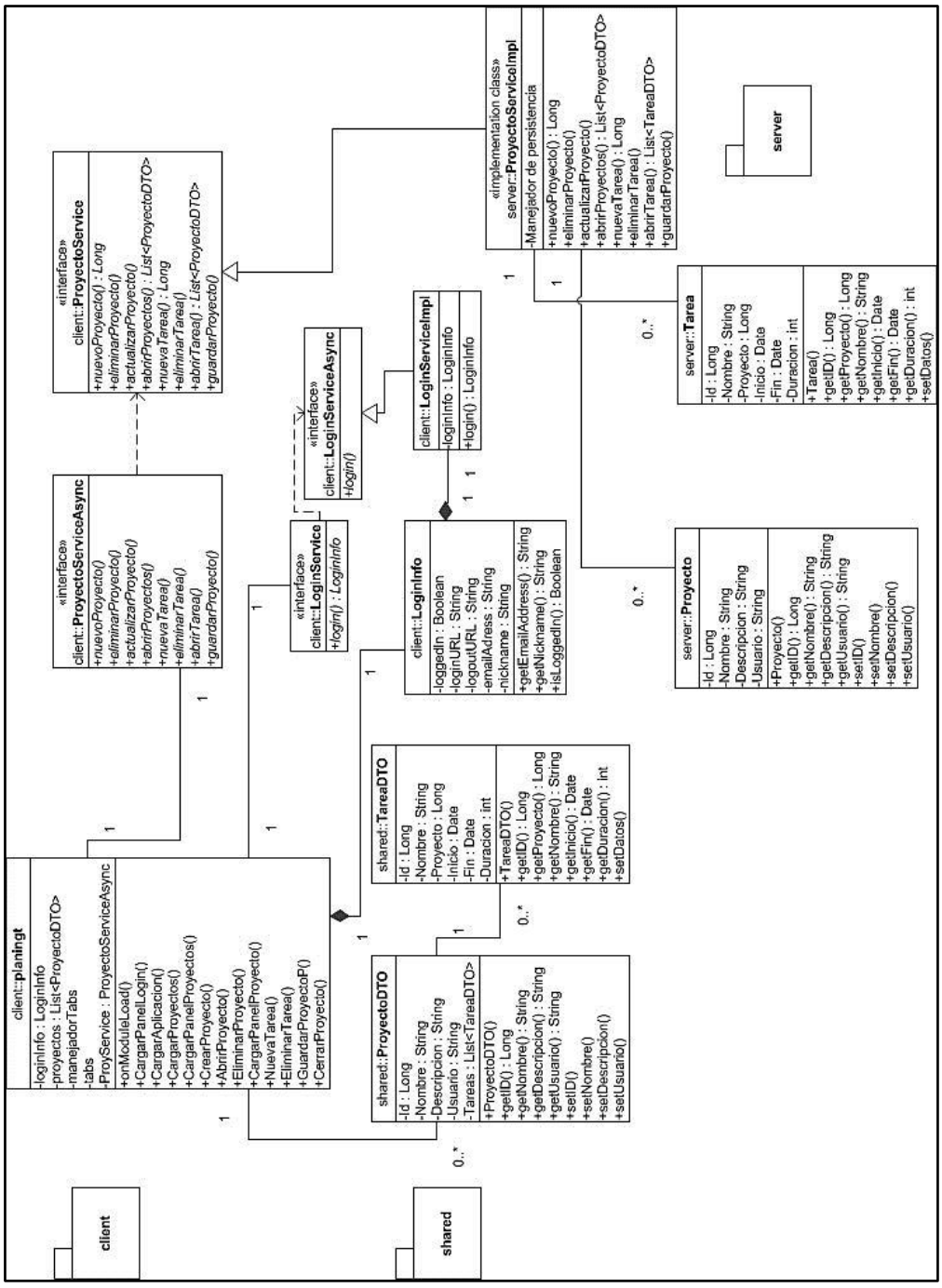
- ProyectoDTO: contiene toda la información de un proyecto, estos son: id, nombre, descripción, usuario, tareas y el conjunto de *getters* y *setters.*

- TareaDTO: contiene los atributos de una tarea los estos son: id, nombre, proyecto, inicio, fin, duración de la tarea y el conjunto de *getters* y *setters*.

- Paquete *client*: contiene las clases utilizadas para la autenticación de usuario, almacena la información y es estado del mismo.

- Paquete *server*: contiene las clases; proyecto, tarea y *ProyectoServiceImp,* las encargadas del manejo de la persistencia de los datos en el lado del servidor.

Figura 52. **Diagrama de clases**

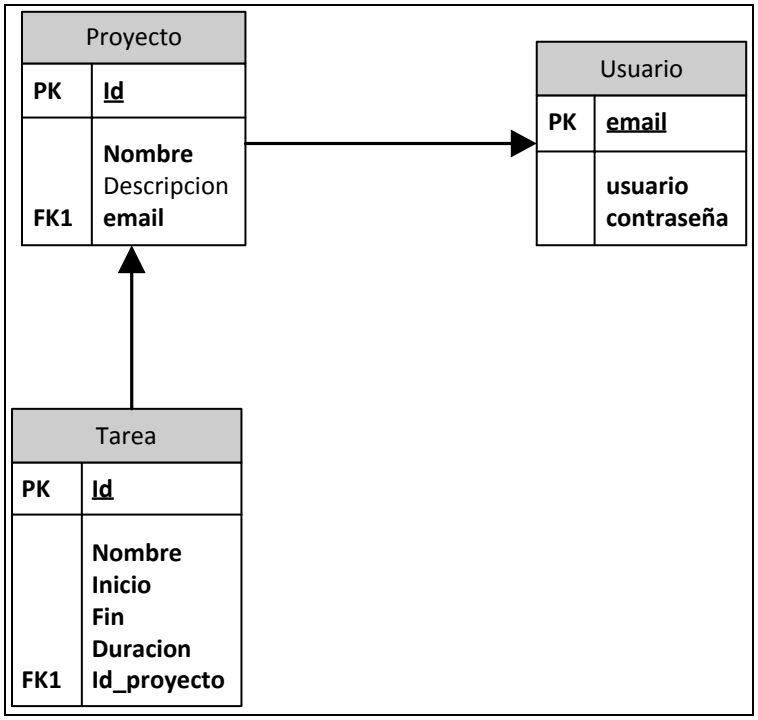


Fuente: elaboración propia.

3.8. Modelo entidad relación

Aunque se ha mencionado que JDO y la Datastore no trabajan un modelo relacional como tal, pero dada su flexibilidad permite adaptarse a las necesidades, acá se representa un diagrama con tres entidades; en la práctica cada una de estas entidades se representará con POJO (clase de Java con atributos, estos son accedidos mediante *getters* y *setters*), estos son programados para que los datos se almacenen como objetos persistentes.

Figura 53. **Diagrama entidad relación**



Fuente: elaboración propia.

3.8.1. Diccionario de la base de datos

En el diccionario de base de datos se describen los campos utilizados en el modelo de almacenamiento de objetos, ajustado al modelo relacional.

Tabla XXIII. **Descripción de la entidad usuario**

Usuario		
Campos	**Descripción**	**Tipo de dato**
Email (PK)	Índice de tabla	String
Nombre	Nombre del usuario	String

Fuente: elaboración propia.

La contraseña y el nombre de usuario no se almacenan debido a que no se declaran como objetos persistentes; dichos datos son necesarios únicamente para el acceso con cuentas de Google.

Tabla XXIV. **Descripción de la entidad proyecto**

Proyecto		
Campos	**Descripción**	**Tipo de dato**
ID (PK)	Llave primaría	Integer
Email (FK)	Correo electrónico del usuario	String
Nombre	Nombre del proyecto	String
Descripción	Descripción del documento	String

Fuente: elaboración propia.

Tabla XXV. **Descripción de la entidad tarea**

Tarea		
Campos	**Descripción**	**Tipo de dato**
ID (PK)	Llave primaria	Integer
Proyecto (FK)	Identificador del proyecto relacionado con la tarea	Integer
Nombre	Nombre que se la da a la tarea	String
Inicio	Inicio programado de la tarea	Date/time
Fin	Final programado de la tarea	Data/time

Fuente: elaboración propia.

Las llaves primarias y foráneas, una vez creadas, no se pueden modificar ni reutilizar, de acuerdo con las restricciones del Datastore.

4. RESULTADO DE LA IMPLEMENTACIÓN

A continuación se presentan los resultados obtenidos de la implementación del sistema Web, donde se mostrarán los requerimientos, cómo obtener el sistema y el resultado de las pruebas de rendimiento realizadas.

4.1. Requerimientos

Se mostrarán entonces los requerimientos de *software* y *hardware*, para obtener, modificar e ingresar al sistema Web.

4.1.1. Requerimientos de *software*

- Sistema operativo independiente con soporte de Java.
- Eclipse Helios 3.6 o superior, obtener desde el sitio Web de Eclipse en: [http://www.eclipse.org/downloads/].
- Java Development Kit JDK 6 o superior. [http://www.oracle.com/technetwork/java/javase/downloads/index.html].
- *Plugin* para IDE Eclipse, que permite el soporte para GWT y GAE incluye el SDK.

- *Plugin*SVN[15] para Eclipse, por ejemplo Subclipse.

- Manual de integración con Eclipse se puede encontrar en: [http://code.google.com/intl/esES/Webtoolkit/usingeclipse.html].

- *PluginSpeed Tracer*(solo para Chrome para desarrolladores), para medir el rendimiento y los problemas de la aplicación en: [http://code.google.com/intl/es-ES/Webtoolkit/speedtracer/get-started.html#downloading].

- Navegador Google Chrome 12 o superior en: [http://www.google.com/chrome?hl=es].

- Otros navegadores de Internet con soporte de JavaScript.

4.1.2. Requerimientos de *hardware*

- Procesadores de la familia Dual Core o superior
- Espacio en el disco duro de por lo menos 2 GB
- Memoria de 1 GB
- Tarjeta de red para conexión a Internet.

[15] Acerca de subversion: http://subversion.tigris.org/. Consulta: 7 de junio de 2011.

4.1.3. Navegadores soportados

Los navegadores de Internet soportados por las aplicaciones desarrolladas con el *framework* de GWT son:

- Google Chrome 12 o superior
- Mozilla Firefox 4 o superior
- Opera 11 o superior
- Safari 5 o superior
- Internet Explorer 9 o superior

Los navegadores de Internet utilizados para el desarrollo, implementación y pruebas son:

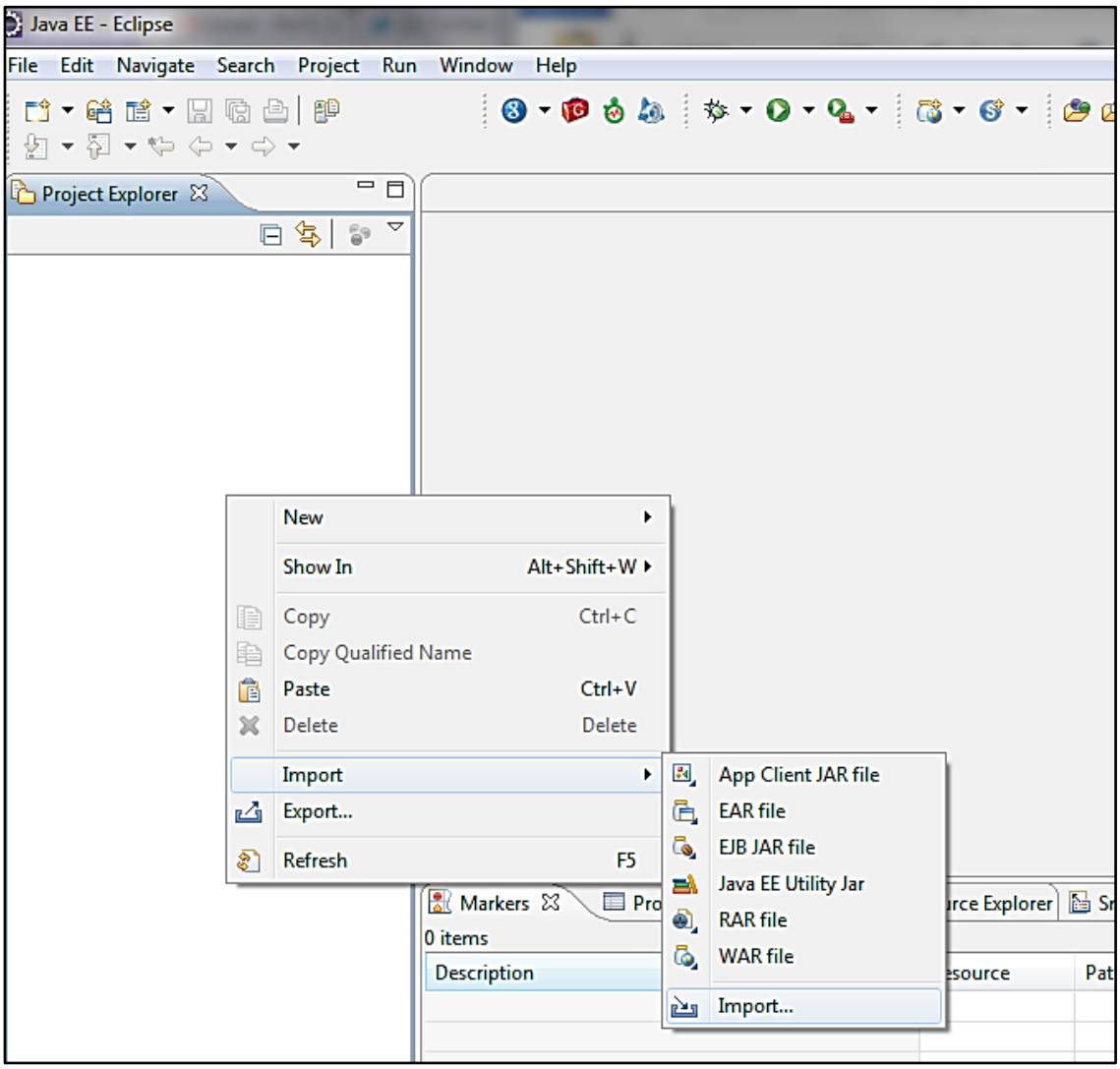
- Google Chrome 12, 13, 14.0.814.0
- Firefox 4, y 5.0
- Operar 11.10
- Safari 5.5
- Internet Explorer 9.0.810112
- Google Chrome 14.0814.0 dev-m (pruebas de rendimiento con Speed Tracer).

4.2. Obtención del proyecto

- Instalación de *plugin* subversión, a su elección por ejemplo, Subversive el mismo lo puede encontrar en la siguiente dirección: [http://www.polarion.com/products/svn/subversive.php].

- Se abre eclipse y se selecciona la sección de exploración, *Project explorer*, clic derecho y seleccionar *import*.

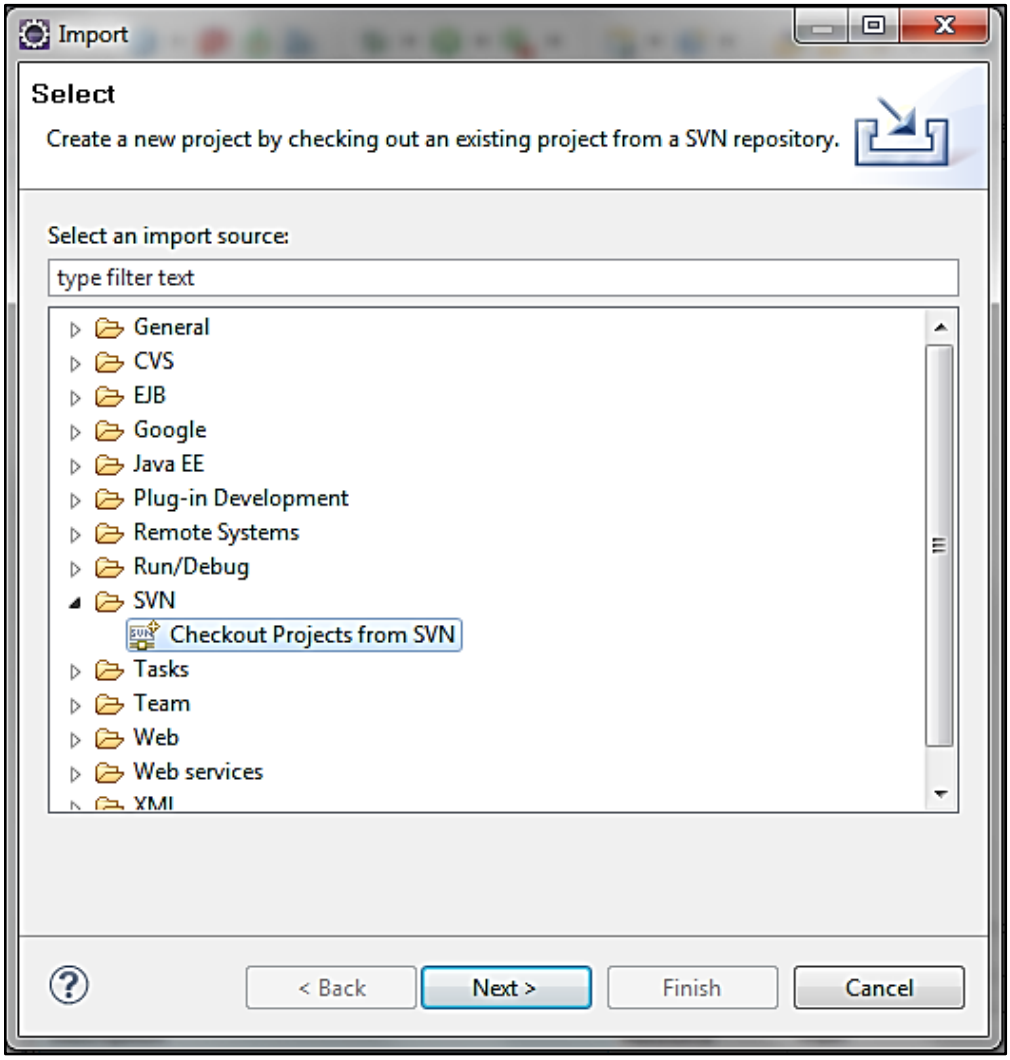
Figura 54. **Seleccionar menú de importar proyecto**



Fuente: elaboración propia. Editado de la captura del IDE Eclipse Helios.

- Seleccionar la rama de SVN, *Checkout Projects fromSVN*. Seleccionar en *Next*.

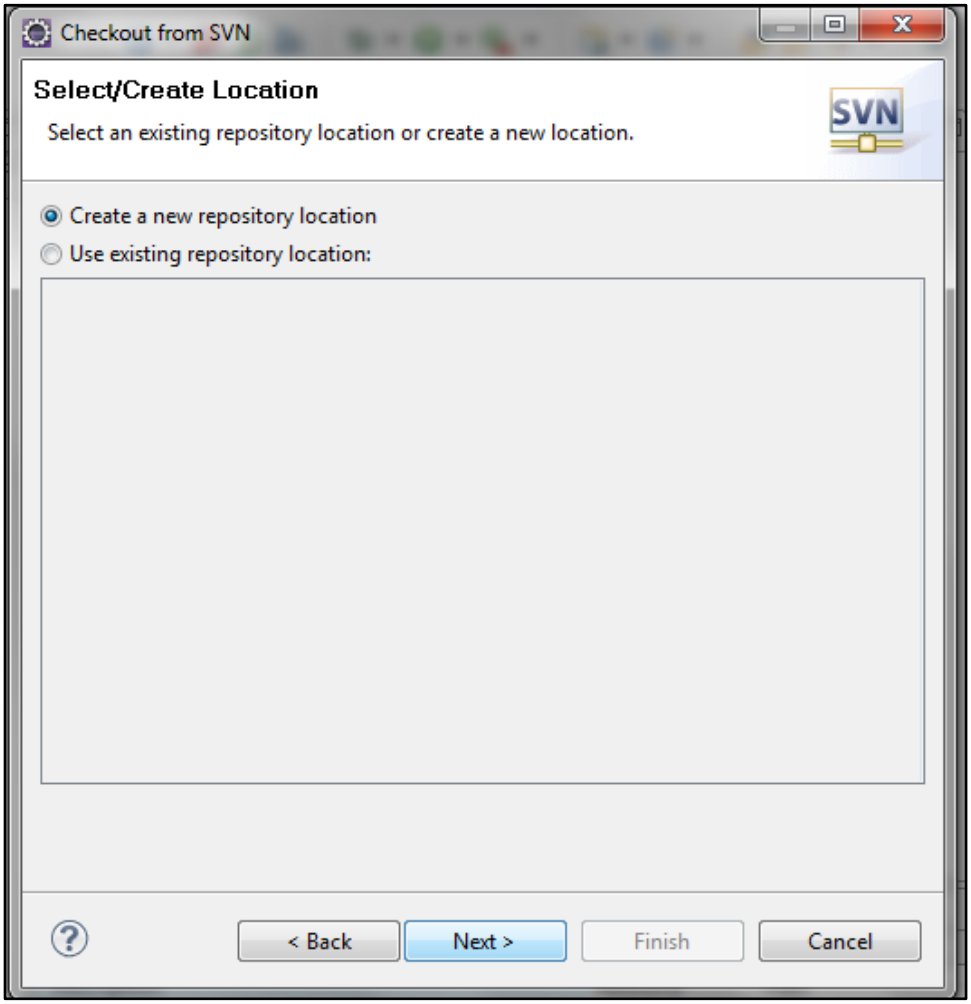
Figura 55. **Seleccionar fuente de importación SVN**



Fuente: elaboración propia. Editado de la captura del IDE Eclipse Helios.

- Seleccionar el crear una nueva fuente de repositorio, *Create a new repository location*, seleccionar *Next*.

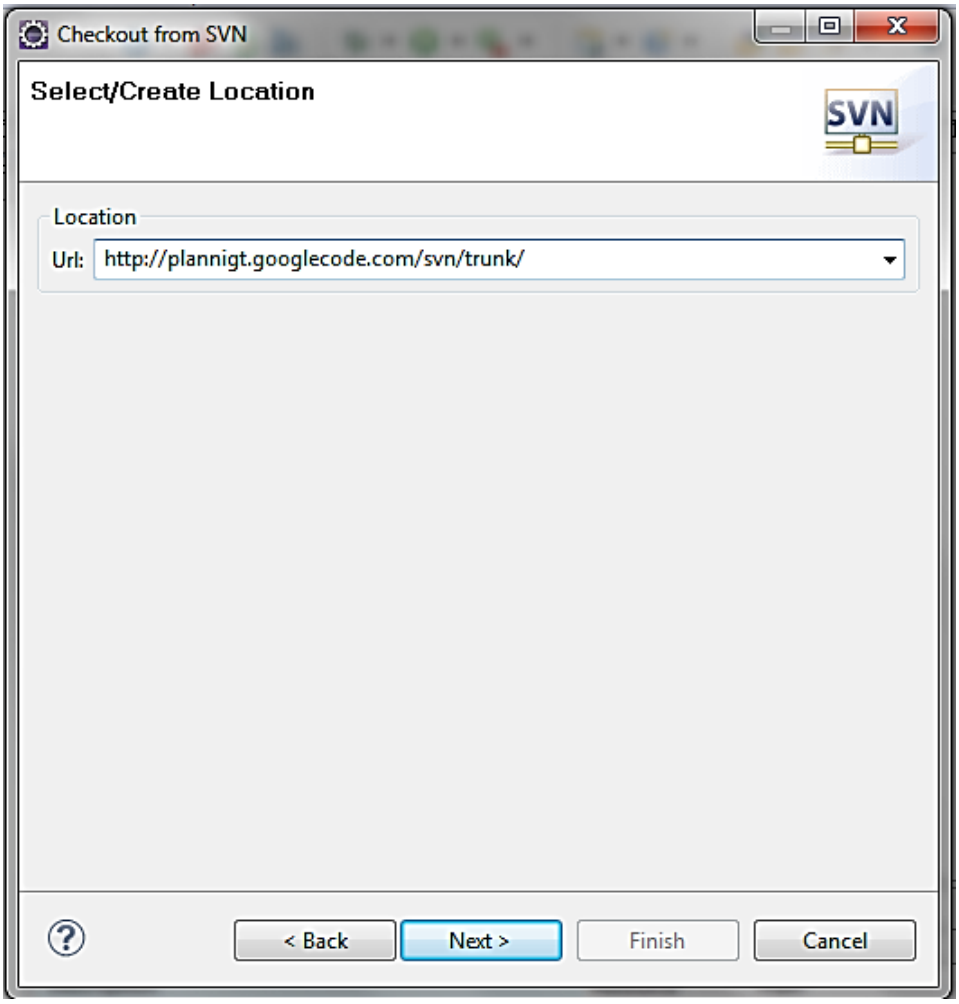
Figura 56. **Seleccionar crear ubicación de repositorio**



Fuente: elaboración propia. Editado de la captura del IDE Eclipse Helios.

- Obtener el código de la aplicación desde Google Code. Añadiendo la siguiente URL: [http://plannigt.googlecode.com/svn/trunk/]y seleccionar *Next*.

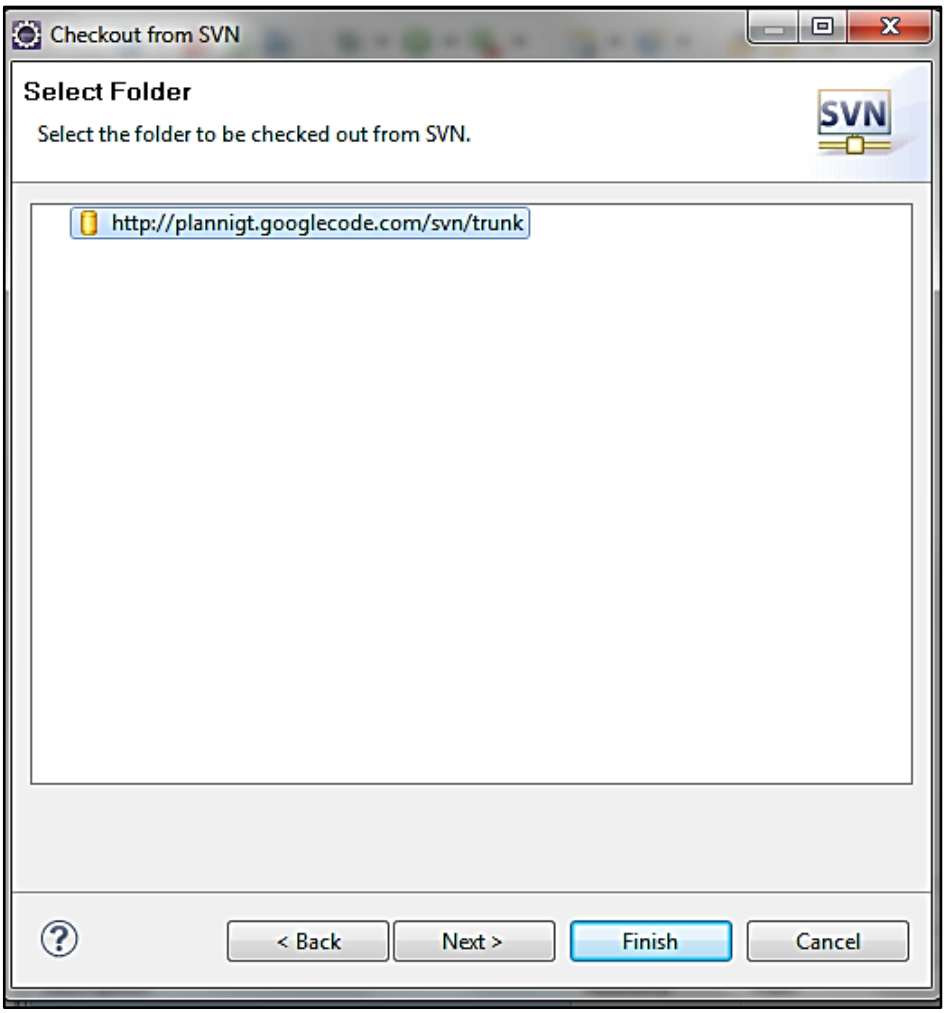
Figura 57. **Ingreso del *trunk* del repositorio**



Fuente: elaboración propia. Editado de la captura del IDE Eclipse Helios.

- Se selecciona el *trunk* y aceptar en finalizar.

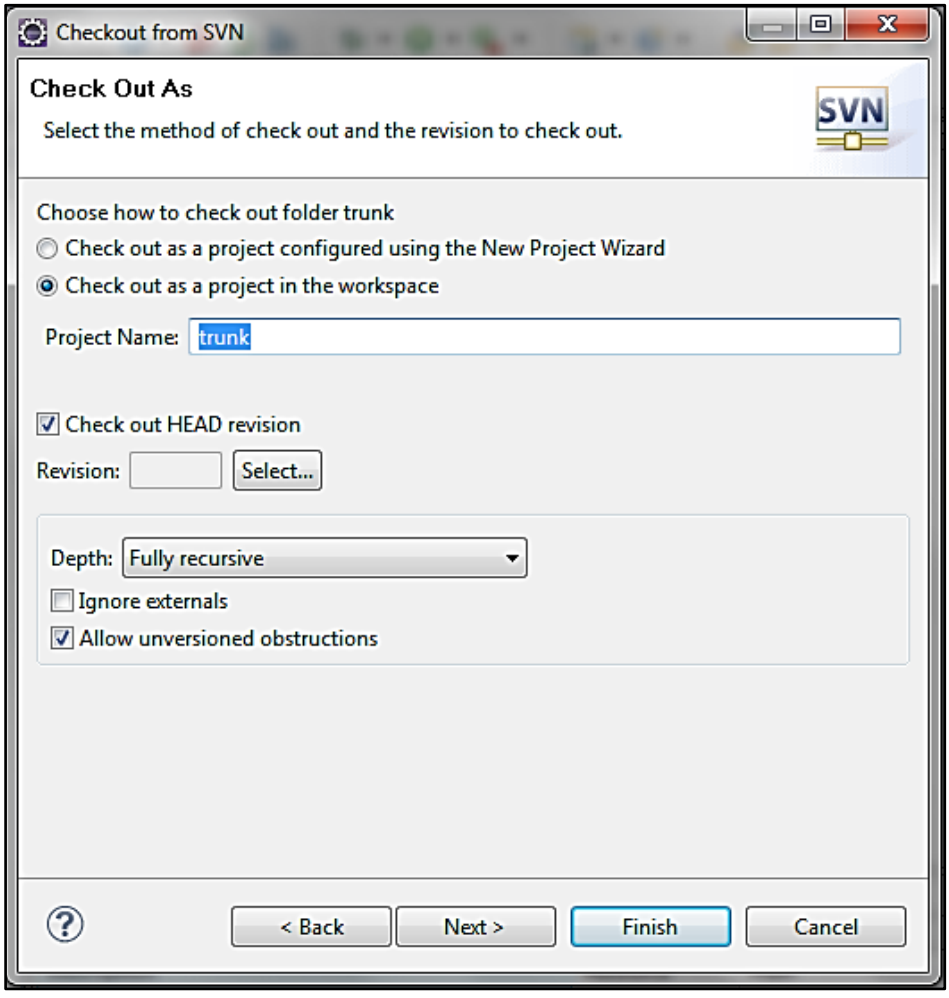
Figura 58. **Seleccionar el repositorio**



Fuente: elaboración propia. Editado de la captura del IDE Eclipse Helios.

- Seleccionar *checkout* como un proyecto en el área de trabajo, colocar el nombre y finalizar.

Figura 59. **Finalizar la exportación**



Fuente: elaboración propia. Editado de la captura del IDE Eclipse Helios.

Si todos los pasos se realizaron de forma correcta, ya se podrá compilar y ejecutar la aplicación, utilizando el *plugin* de GWT y GAE para el IDE Eclipse Helios o la versión utilizada.

4.3. Pruebas de rendimiento

En los requerimientos de *software* se ha hecho mención del *Plugin* de Google Chrome, Speed Tracer[16], que se ejecuta únicamente en la versión *development* de Chrome[17], para saber más sobre cómo instalar y utilizar dicha aplicación, el usuario debe dirigirse a la página del proyecto.

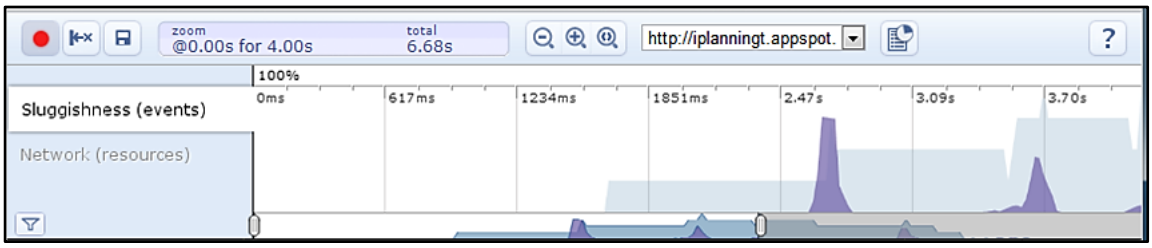
Speed Tracer es entonces un complemento que ayudará a identificar y reparar problemas de sus aplicaciones Web. Este complemento permite visualizar métricas, a través de las cuales se puede determinar cuánto tiempo emplean las aplicaciones Web en cargar y determinar la cantidad de recursos de red, que utilizan en su tiempo de carga.

A continuación se presenta un breve análisis, de los resultados obtenidos de la herramienta Speed Tracer, con la el sistema Web que se ha construido; cabe mencionar que las pruebas se realizaron con una velocidad de conexión e 512 Kbps.

[16] Speed Tracer: http://code.google.com/intl/es-ES/webtoolkit/speedtracer/. Consulta: 8 junio de 2011.
[17] Chrome Dev Channel: http://dev.chromium.org/getting-involved/dev-channel#TOC-Subscribing-to-a-channel. Consulta: 8 de junio de 2011.

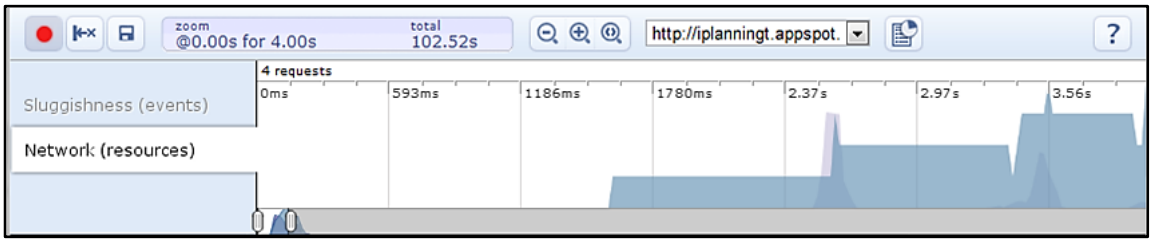
- Análisis de la autenticación de usuario (Login).

Figura 60. **Línea de tiempo de eventos**



Fuente: elaboración propia. Editado de la captura del *plugin Speed Tracer.*

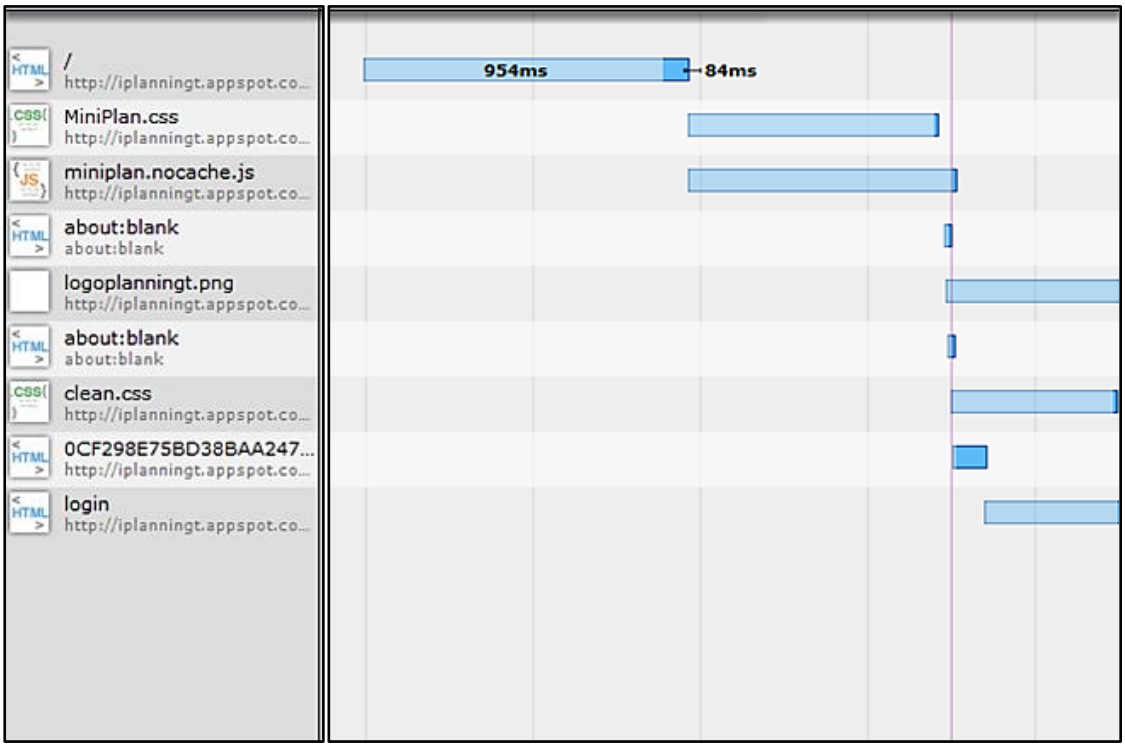
Figura 61. **Línea del tiempo de recursos de red utilizados**



Fuente: elaboración propia. Editado de la captura del *plugin Speed Tracer.*

En la figura 60, se muestran los eventos que pueden ralentizar la carga de la aplicación en el navegador, en lo que se conoce como *sluggishness*. En la figura 61, se puede visualizar cómo crece el recurso de red conforme avanza el tiempo de utilización de la aplicación. Puede verse que los eventos de *sluggishness* están relacionados proporcionalmente con el uso de la red.

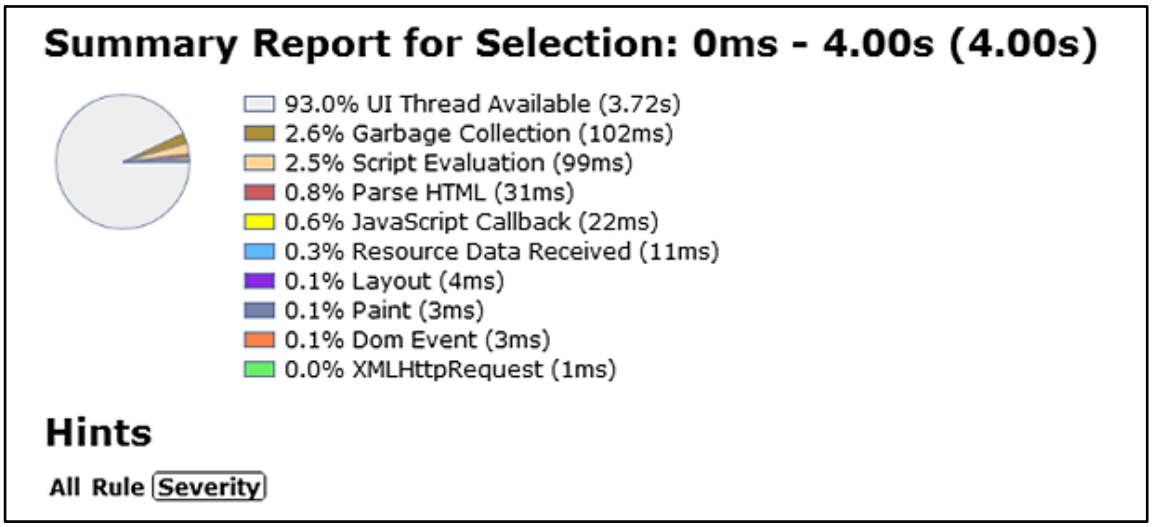
Figura 62. **Línea del tiempo de archivos cargados**



Fuente: elaboración propia. Editado de la captura del *plugin Speed Tracer.*

En la figura 62 se muestra la forma en la que se van cargando los archivos, de acuerdo a como son solicitados; se puede observar que el primer archivo, el HTML, es el que toma mayor tiempo en cargarse; 954 milisegundos. En el que 84 milisegundos representan al tiempo de saturación de los recursos de red y del sistema operativo empleado por el navegador Web; en este caso Google Chrome en su versión para desarrolladores, puede observarse que, 84 milisegundos es un tiempo relativamente minúsculo, por lo cual en este problema se pueden omitir esfuerzos por solucionarlo.

Figura 63. **Reporte del proceso *login***



Fuente: elaboración propia. Editado de la captura del *plugin Speed Tracer.*

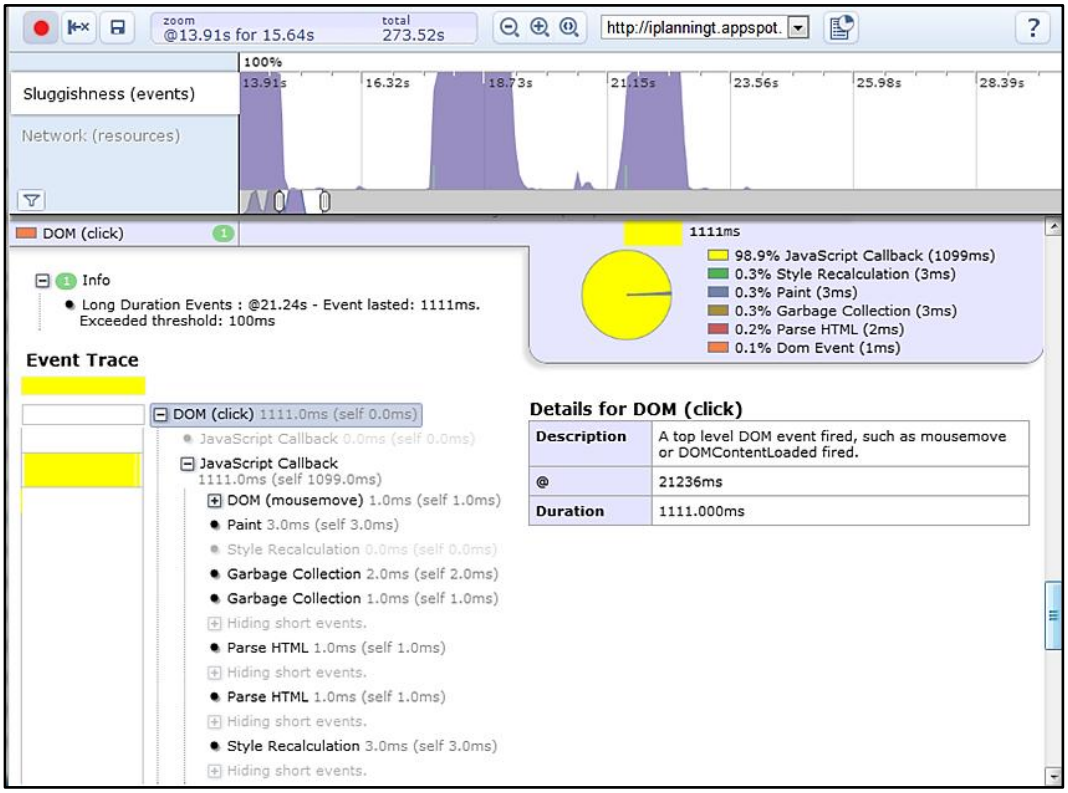
Speed Tracer le proporciona un reporte general que muestra cada una de las características que analizó y utiliza una nomenclatura por color, la cual informa sobre a cuáles características se le debe de poner mayor atención, debido a que en los elementos involucrados puede existir un problema de rendimiento potencial como se muestra en la figura 63.

- Verde (prioridad baja)
- Naranja (advertencia)
- Rojo (prioridad alta)

Los niveles de estos indicadores son bajos, por ejemplo los de DomEvent y XMLHttpRequest; además no se ha desplegado ninguna información extra en *Hints* (indicador), por lo cual esta prueba es satisfactoria.

- Análisis de aplicación, general: en la figura 64, se procede a revisar los resultados de una prueba utilizando la aplicación en un periodo mayor a 5 minutos.

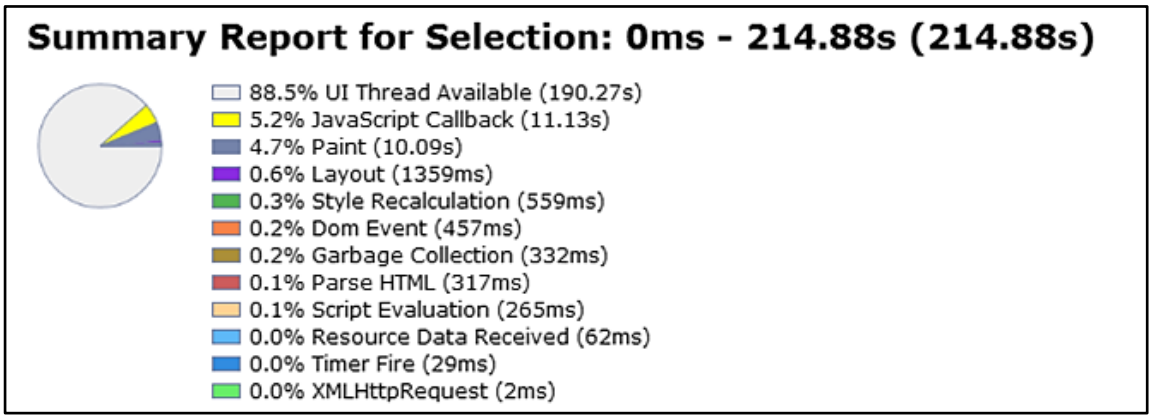
Figura 64. **Descripción de la velocidad de un evento**



Fuente: elaboración propia. Editado de la captura del *plugin Speed Tracer.*

Se pueden observar situaciones más interesantes, por ejemplo, se encuentran los primeros indicadores de alerta que están en la aplicación. Al desplegar la información en el evento Dom, puede observarse que la latencia ocurre en la solicitud de una instrucción JavaScript.

Figura 65. **Reporte de la utilización del servidor**



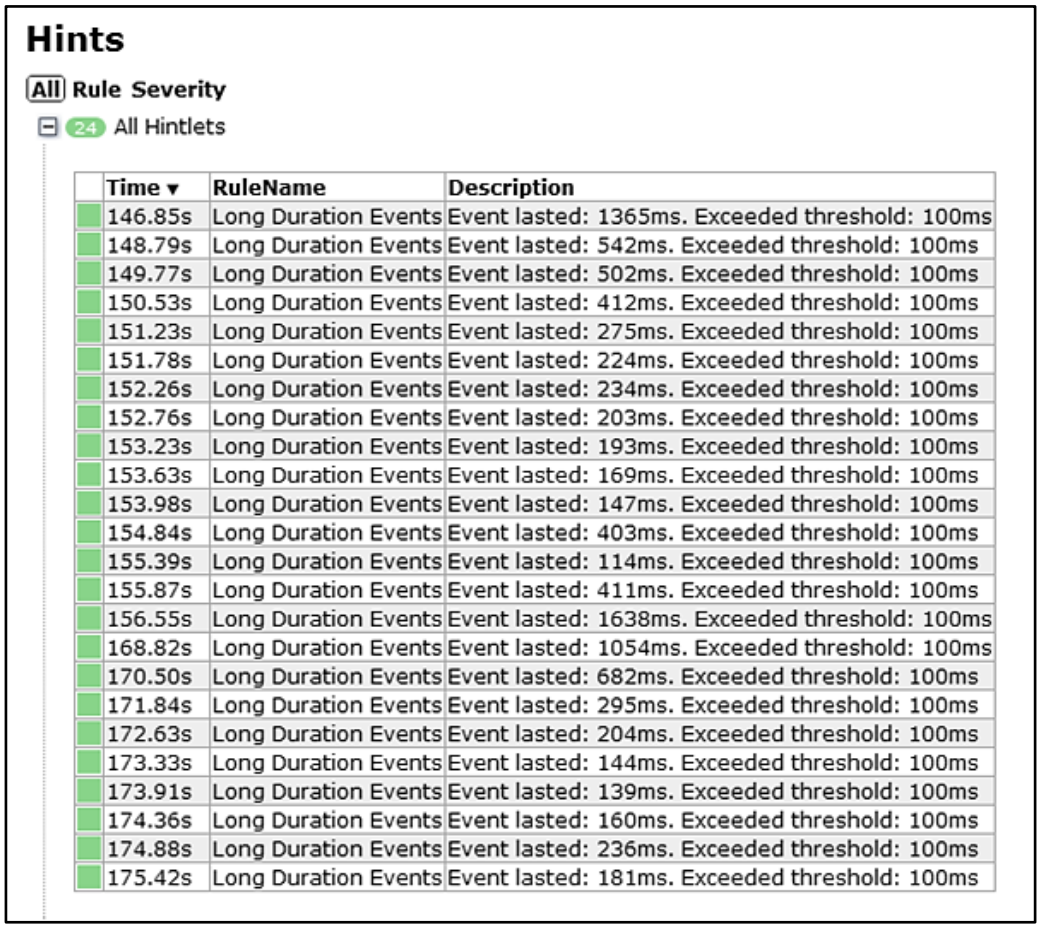
Fuente: elaboración propia. Editado de la captura del *plugin Speed Tracer.*

Se podrá observar en el sumario, en la figura 65, que la mayor cantidad de tiempo 5.2% se desperdicia en solicitudes de JavaScript, seguidos por las solicitudes de *paint* (dibujar) pero que no son significativas.

Por último, pueden observarse en la figura 66, los 24 indicadores que son posibles indicios de problemas en latencia; estos están catalogados por prioridad baja, por lo cual son errores que se encuentran en rangos que varían de acuerdo con la cantidad de datos procesados y la inmediatez de un proceso con otro; debido a esto no se pueden evitar del todo, pero en rubros generales la aplicación ha pasado satisfactoriamente la prueba de rendimiento.

Gracias al servicio de GAE, se despreocupa al usuario de las pruebas de estrés al servidor y otras situaciones similares.

Figura 66. **Reportes de indicadores con posibles errores**



Hints

All Rule Severity

24 All Hintlets

	Time ▾	RuleName	Description
	146.85s	Long Duration Events	Event lasted: 1365ms. Exceeded threshold: 100ms
	148.79s	Long Duration Events	Event lasted: 542ms. Exceeded threshold: 100ms
	149.77s	Long Duration Events	Event lasted: 502ms. Exceeded threshold: 100ms
	150.53s	Long Duration Events	Event lasted: 412ms. Exceeded threshold: 100ms
	151.23s	Long Duration Events	Event lasted: 275ms. Exceeded threshold: 100ms
	151.78s	Long Duration Events	Event lasted: 224ms. Exceeded threshold: 100ms
	152.26s	Long Duration Events	Event lasted: 234ms. Exceeded threshold: 100ms
	152.76s	Long Duration Events	Event lasted: 203ms. Exceeded threshold: 100ms
	153.23s	Long Duration Events	Event lasted: 193ms. Exceeded threshold: 100ms
	153.63s	Long Duration Events	Event lasted: 169ms. Exceeded threshold: 100ms
	153.98s	Long Duration Events	Event lasted: 147ms. Exceeded threshold: 100ms
	154.84s	Long Duration Events	Event lasted: 403ms. Exceeded threshold: 100ms
	155.39s	Long Duration Events	Event lasted: 114ms. Exceeded threshold: 100ms
	155.87s	Long Duration Events	Event lasted: 411ms. Exceeded threshold: 100ms
	156.55s	Long Duration Events	Event lasted: 1638ms. Exceeded threshold: 100ms
	168.82s	Long Duration Events	Event lasted: 1054ms. Exceeded threshold: 100ms
	170.50s	Long Duration Events	Event lasted: 682ms. Exceeded threshold: 100ms
	171.84s	Long Duration Events	Event lasted: 295ms. Exceeded threshold: 100ms
	172.63s	Long Duration Events	Event lasted: 204ms. Exceeded threshold: 100ms
	173.33s	Long Duration Events	Event lasted: 144ms. Exceeded threshold: 100ms
	173.91s	Long Duration Events	Event lasted: 139ms. Exceeded threshold: 100ms
	174.36s	Long Duration Events	Event lasted: 160ms. Exceeded threshold: 100ms
	174.88s	Long Duration Events	Event lasted: 236ms. Exceeded threshold: 100ms
	175.42s	Long Duration Events	Event lasted: 181ms. Exceeded threshold: 100ms

Fuente: elaboración propia. Editado de la captura del *plugin Speed Tracer.*

CONCLUSIONES

1. El desarrollo de un sistema Web minimalista para la planificación de proyectos individuales, permitió la utilización de tecnologías que se basan principalmente en RIA. Se eligió un framework de desarrollo que brinda facilidades para su implementación, el cual fue GWT, pues permite aprovechar el servicio de *cloud computing* GAE, para desarrollar aplicaciones AJAX a partir del uso del lenguaje Java.

1. El desarrollo de aplicaciones RIA con GWT y GAE, redujo significativamente la complejidad de la utilización de tecnologías AJAX, pero trajo en contraparte un gran número de problemas en la codificación, principalmente debido a que GWT como *framework* de desarrollo es relativamente nuevo y que GAE presenta algunas desventajas por su problemas de restricciones, convirtiendo con esto ciertas tareas sencillas en difíciles, pero siempre brindando ventajas en lo que se refiere a integración, administración y escalamiento.

2. El sistema Web no solo aprovechó las tecnologías que envuelven RIA, sino que además se valió de las ventajas que incluye utilizar tecnologías Google como GWT con GAE para el desarrollo e implementación, donde además se integró con servicios como Google Accounts para el manejo de cuentas de usuario y autenticación, Google Apps para tareas adicionales, el API de *hosting* y control de versiones de Google Code.

3. El sistema Web, llamado PlanninGt, se encuentra alojado sobre la infraestructura de GAE, la cual permite despreocuparse de la disponibilidad, espacio de almacenamiento, capacidad de procesamiento y ancho de banda, ya que de esto se encarga GAE como un sistema de *PaaS*.

4. El sistema PlanninGt es una herramienta para la administración de proyectos individuales, la cual se ha desarrollado con el objetivo de ser intuitiva, es decir de fácil utilización, pues fue diseñada como un sistema minimalista que le permite al usuario ejecutar acciones reduciendo la cantidad de movimientos en la interfaz; abarcando situaciones y casos de uso que un usuario necesitaría para planificar no solo proyectos, sino situaciones tan cotidianas como un día de campo, un cumpleaños o una agenda.

5. La implementación de PlanninGt tiene el valor agregado de haber utilizado las tecnologías de infraestructura Google que GAE provee y además se ha creado una aplicación de *software* libre bajo la licencia GNU/GPL, para que los desarrolladores de aplicaciones sobre GAE, puedan tomar como punto de partida y continuar con su desarrollo.

RECOMENDACIONES

1. A los desarrolladores, se les sugiere la revisión iterativa de sus requerimientos y casos de uso, para la elaboración de un sistema que logre cumplir con los requerimientos trazados. Esto es posible al utilizar una metodología de desarrollo como eXtream Programming, la cual se adapta de muy buena manera con el desarrollo de aplicaciones con Google App Engine, debido a que se puede aplicar un ciclo de vida circular, construir, probar, publicar, administrar, actualizar y construir.

2. Debido a que se están utilizando tecnologías que se encuentran en su fase de desarrollo *Beta*, se aconseja a utilizar las actualizaciones más recientes y mantenerse al día con el *FAQ, blog, wiki* y canal de noticias de cada proyecto de *software*, para darle seguimiento a los posibles cambios que las herramientas tendrán, para tenerlos en mente en el desarrollo de la aplicación.

3. Para el desarrollo de aplicaciones GAE y GWT, existen *plugins* para distintos IDE, pero que por excelencia para el desarrollo de la misma es Eclipse; aunque siempre se puede recurrir a buscar estructuras para la construcción de proyectos, haciendo uso de Maven.

4. Se debe de tener especial cuidado con la administración de la información en el Datastore, pues la información que se elimina es imposible de recuperarla.

5. El *plugin* Speed Tracer es una herramienta muy completa para el análisis de errores y rendimiento de las aplicaciones Web, pero existe la libertad de utilizar otra herramienta.

6. A los desarrolladores, se les recomienda leer la documentación oficial de cada una de las herramientas y tecnologías anteriormente descritas; este documento es para un fin ilustrativo y no una guía paso a paso para el desarrollo de aplicaciones sobre GAE.

7. A los usuarios se les aconseja utilizar el navegador de Internet Google Chrome, debido a que está optimizado para la ejecución de aplicaciones que se ejecutan sobre la infraestructura de GAE.

BIBLIOGRAFÍA

1. DWYER, Jeff. Pro *Web 2.0 application development with GWT*. New York: Apress, 2008. 480 p. ISBN: 978-1-4302-0638-5.

2. EGUÍLUZ, Javier. *Introducción a AJAX.* [en línea] España: librosweb.es, [ref. 8 junio de 2008] Disponible en Web: <http://www.librosweb.es/ajax/>.

3. EXPÓSITO, Raúl. *¿Qué son el cloud computing y google app engine?* [en línea] España: adictosaltrabajo.com. [ref. 7 de diciembre de 2009] Disponible en Web: <http://adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=cloud computing>.

4. EXPÓSITO, Raúl. *¿Cómo utilizar el datastore de Google App Engine con su API de nivel inferior?* [en línea] España: adictosaltrabajo.com. [ref. 17 de febrero de 2010] Disponible en Web:<http://adictosaltrabajo.com/tutoriales/tutoriales.php?pagina= datastoreAPIBajoNivel>.

5. FAY CHANG, J. D. *Bigtable: a distributed storage system for structured.* [en línea] EEUU: Google. [ref. 10 de noviembre de 2006] Disponible en Web: <http://labs.google.com/papers/bigtable.html>.

6. FUMEAUX, B. *Task-technology fit*. [en línea] Canadá: York University. [ref. 7 de noviembre de 2010] Disponible en Web: <http://www.fsc.yorku.ca/york/istheory/wiki/index.php/Task-technology_fit>.

7. GUERMEUR, Daniel. *Google App Engine Java and GWT Application Development*. Olton, Birmingham: Packt Publishing, 2010. 480 p. ISBN: 978-1-849690-44-7.

8. GOOGLE. *esgoogleWebtoolkit*. [en línea] EEUU: Google. [ref 15 de septiembre de 2009] Disponible en Web: <http://esgoogleWebtoolkit.blogspot.com/>.

9. _____. *Google App Engine documentación*. [en línea] EEUU: Google. [ref: 31 de octube de 2010] Disponible en Web: <http://code.google.com/intl/es/appengine/docs/>.

10. GUPTA, Vipul. *Accelerated GWT: building enterprise Google Web toolkit applications.* New York: Apress, 2008. 313 p. ISBN: 978-1-4302-0616-3.

11. LUBBERS, Peter. *Pro HTML5 Programming: powerful APIs for richer Internet application development*. New York: Apress, 2010. 305 p. ISBN: 978-1-4302-2791-5.

12. LIEBERMAN, Gerld. *Investigación de operaciones.* 8a ed. México: McGraw-Hill, 2003. 1256 p. ISBN: 978-9701034866.

13. MOORE, Dana. *Professional Rich Internet Applications: AJAX and beyond*. Indianápolis, Indiana: Wiley Publishing, 2007. 603 p. ISBN-13: 978-0-470-08280-5.

14. ODELL, Den. *Pro JavaScript RIA techniques: best practices, performance, and presentation*. New York: Apress, 2010. 442 p. ISBN-13 978-1-4302-1935-4.

15. Project Management Institute. *Guía de los fundamentos de dirección de proyectos.* 3a ed. Newtown Square, EEUU: PMI, 2004. 392 p. ISBN-13: 978-1930699731.

16. GHEMAWAT, Sanjay. *The Google File System.* [en línea] [ref. 17 de octubre de 2003] Disponible en Web: <http://labs.google.com/papers/gfs.html>.

17. TORO, Francisco. Planeación y control de proyectos usando herramientas computacionales. Colombia: Compulearing, 2008. 225 p. ISBN-13: 978-9583393235.

18. WELLS, Don, *Extreme programming: a gentle introduction.* [en línea] [ref. 28 de noviembre de 2011] Disponible en Web: <http://www.extremeprogramming.org/>.

19. WYKE SMITH, C. *Scriptin' with JavaScript and AJAX: a designer's guide*. EEUU: Riders Press, 2009. 312 p. ISBN-13: 978-0321572608.

20. ZAKAS, Nicholas. *Professional AJAX*. 2a ed. Indianápolis, Indiana: Wiley Publishing 2007. 627 p. ISBN: 978-0-470-10949-6.