PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

FACULTAD DE CIENCIAS E INGENIERÍA

SISTEMA INTEGRADO CON SERVICIOS WEB QUE BRINDE SOPORTE A LOS PROCESOS DE GESTIÓN DE PROYECTOS DE LA EMPRESA DESARROLLADORA DE SOFTWARE TAU

Tesis para optar el Título de Ingeniera Informática, que presenta el bachiller:

Paul Francisco Diaz Dumont

ASESOR: Jorge Alberto Berrocal Perez Albela

Lima, Abril de 2016

Resumen

En la actualidad existen diversas empresas que utilizan servicios web para dar soporte a sus procesos, lo que genera una dependencia directa al uso de este tipo de herramientas. Si bien pueden existir problemas de integración entre ellas, causando diversas dificultades tales como la distribución en la productividad y la inconsistencia de datos, los beneficios que brindan suelen ser mucho mayores por lo que se justifica la decisión de usar los servicios web.

La empresa Tau, denominada así por temas de confidencialidad, utiliza servicios de diversos proveedores para realizar sus actividades comerciales y presenta problemas por la falta de estándares de comunicación entre dichos componentes. El presente proyecto contempla el análisis de las deficiencias de esta forma de trabajo en el contexto específico de dicha organización con el fin de identificar posibilidades de mejora y con ello construir una herramienta que facilite su trabajo con aplicaciones basadas en tecnologías web.

De este modo, el resultado final del proyecto es un sistema de información que apoyará a la ejecución de ciertos procesos de la empresa y que cuenta con la particularidad de integrarse con múltiples proveedores de servicios web, actuando como componente unificador entre ellos. El proceso de construcción realizado se encuentra detallado a lo largo de los siete capítulos que conforman este documento. Los capítulos se encuentran distribuidos de la siguiente forma:

Los capítulos uno y dos describen la problemática que se busca resolver, así como el marco conceptual que permite entender de una manera más completa la situación en la cual se presenta.

El capítulo número tres contempla el análisis de las funcionalidades requeridas por el proyecto, y los componentes que definen su comportamiento, tales como los usuarios y los casos de uso.

El cuarto capítulo está enfocado a la especificación de los servicios web con los cuales se integrará el sistema propuesto y los mecanismos de comunicación y autenticación empleados.

Por otro lado, los capítulos cinco y seis detallan el aspecto técnico del desarrollo e implementación del producto. Se describen las herramientas y conceptos utilizados, así como la arquitectura del sistema y las consideraciones tomadas en cuenta para su construcción.

Finalmente, el capítulo siete presenta las conclusiones, recomendaciones y observaciones resultantes del desarrollo del presente trabajo, las cuales son aplicables para futuros proyectos con una temática similar.

Indice de contenidos

Indice de figuras

Indice de tablas

1.1 Problemática

En la actualidad se tiene un número creciente de empresas que hacen uso de servicios alojados en la nube para controlar sus actividades comerciales. En la Figura 1.1 se muestra el resultado de un estudio realizado por PwC, una prestigiosa firma transnacional de servicios profesionales [PWC1], en el que se encuestó a empresas a lo largo del mundo sobre el uso de tecnologías en la nube. Los resultados indicaron que un alto porcentaje de encuestados hace uso de dichos servicios y que, entre los años 2012 y 2013 el porcentaje de uso muestra una tendencia ascendente.

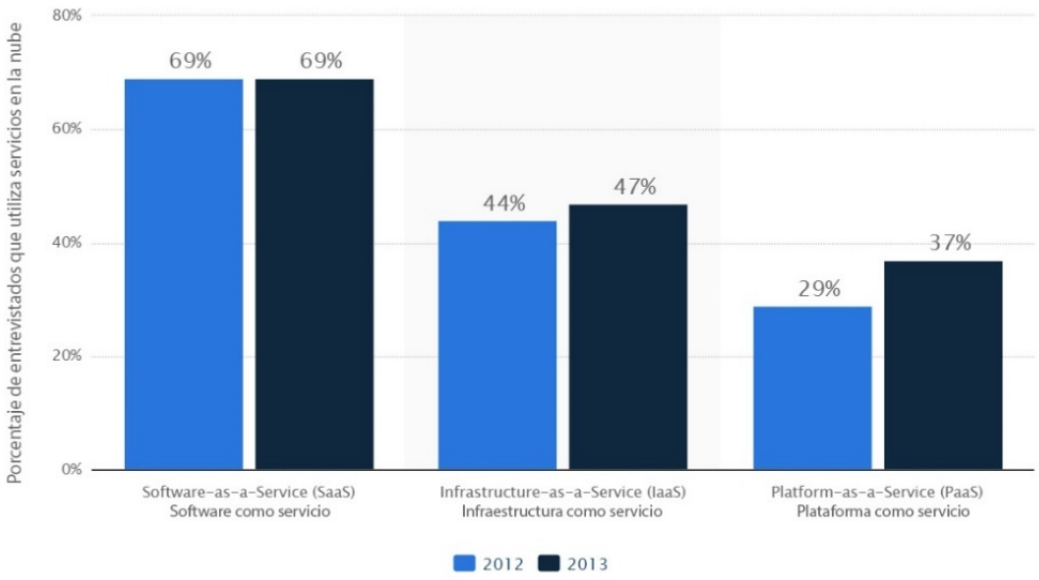

Figura 1.1: Porcentaje de uso de servicios en la nube [PWC2].

De este modo, el software en la nube juega un papel importante para las empresas, quienes optan por cambiar las aplicaciones de escritorio por otras accesibles a través de un entorno web. Para conocer el detalle del tipo de servicios que se busca reemplazar, se muestra en la Figura 1.2 el resultado de un estudio realizado por CISCO en el 2012 [CISCO] y que tuvo como objetivo evaluar el impacto de los servicios en la nube dentro de las empresas. La pregunta realizada a los expertos en la toma de decisiones de tecnologías de información fue "si estuviera limitado a migrar un único tipo de servicio a la nube, ¿cuál sería?". Los resultados de dicha pregunta se encuentran en la Figura 1.2 y se puede observar que servicios básicos como almacenamiento o correo electrónico son los que destacan en la cola de migración.

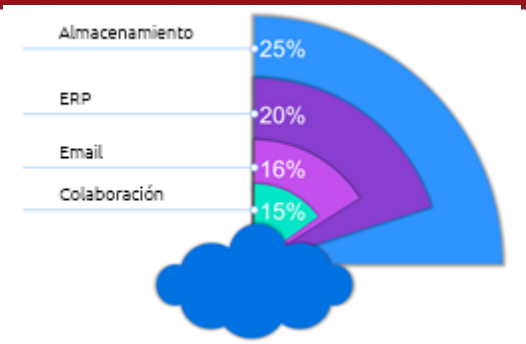

Figura 1.2: Ranking de tipo de servicios migrados a la nube [CISCO].

Esta tendencia creciente a reemplazar las aplicaciones de productividad tradicionales por otras alojadas en la nube trae consigo muchos beneficios, como el ahorro de tiempo y dinero [PAPER1], pero también implica algunas consecuencias negativas que se detallan a continuación [TAU]:

- Altas curvas de aprendizaje: cada servicio cuenta con un modo diferente de ser usado, provocando que la transición entre diferentes aplicaciones resulte difícil para el usuario final.
- Altas tasas de error: el uso simultáneo de múltiples herramientas conlleva a que sea más complicado agregar y actualizar la información en ellas, causando que se cometan errores u omisiones.
- Pérdida de la integridad de la información: al repetir el ingreso de información en múltiples ocasiones se incrementa la posibilidad de alterar el formato original de los datos causando que no reflejen el valor deseado.
- Disminución de la productividad: la interacción con diferentes plataformas fuerza al usuario a realizar tareas repetitivas y tediosas de ingreso de datos con lo que su rendimiento general disminuye.
- Rechazo a nuevas tecnologías: dados los problemas mencionados anteriormente, es posible que los usuarios empiecen a desarrollar un cierto rechazo a soluciones de este tipo contribuyendo a que se genere un entorno desfavorable para el desarrollo de las actividades tecnológicas de la empresa.
- Pérdida de objetividad: ya que la información se encuentra distribuida en diferentes lugares, es decir, se encuentra almacenada en servicios de múltiples proveedores, no existe un modo de analizar los datos en conjunto e interpretar el estado actual de los procesos de la empresa.

Tomando en cuenta las consideraciones mencionadas en los párrafos anteriores, se vuelve evidente que existe un grave problema en la administración de los múltiples servicios de los

que hacen uso las empresas, sin que la razón radique en que las herramientas provengan de diferentes proveedores, sino que estas no pueden comunicarse entre sí y compartir la información con la que trabajan.

El problema descrito anteriormente también se presenta en la empresa que se denominará TAU por políticas de privacidad, la cual hace uso de servicios en la nube para llevar el control de sus actividades comerciales. TAU es una empresa privada dedicada al desarrollo de software y en todas las etapas de sus procesos utiliza como herramientas de gestión de la información software en la nube.

El principal problema que se identifica en esta administración [TAU] consiste en la falta de sincronización entre los datos ingresados en los diferentes servicios, además del proceso de registro manual de mucha de la información, lo que causa retrasos en el análisis y la imposibilidad de tener una visión general de todo el proceso, dado que la información se encuentra repartida de manera aislada.

Finalmente, para remediar la problemática mencionada se propone como una alternativa de solución el desarrollar un sistema que actúe de interfaz intermediaria entre los principales sistemas basados en la nube usados por TAU, permitiendo la automatización del proceso de registro, análisis y control de la información contenida en las múltiples plataformas usadas.

1.2 Objetivo general

Analizar, diseñar e implementar un sistema de información integrado con servicios basados en tecnologías web que permita la automatización de los principales procesos de gestión de proyectos de una empresa desarrolladora de software.

1.3 Objetivos específicos

Los objetivos específicos (OE) del proyecto son:

- OE1: Modelar los procesos del negocio que serán soportados por la solución.
- OE2: Seleccionar adecuadamente las aplicaciones basadas en tecnologías web que satisfagan los requerimientos de los procesos seleccionados.
- OE3: Definir una arquitectura orientada a servicios que permita la comunicación entre las diferentes aplicaciones web elegidas y que trabajan de manera independiente.
- OE4: Diseñar e implementar la solución propuesta de modo que cumpla con los objetivos descritos anteriormente.

1.4 Resultados esperados

Los resultados esperados (RE) correspondientes a los objetivos específicos mencionados anteriormente son:

- Para OE1: Diagrama de procesos de negocio que contemplen los flujos automatizados de las tareas de gestión de la empresa.
- Para OE2: Listado de las aplicaciones basadas en tecnologías web con las cuales se integrará la solución propuesta, indicando también el motivo de la elección.
- Para OE3: Catálogo de los servicios que la solución propuesta consumirá de las aplicaciones seleccionadas en el objetivo 2, así como los servicios que ofrecerá para que otras aplicaciones se integren con ella.
- Para OE4: Producto de software final, desarrollado y funcional.

1.5 Herramientas, métodos y procedimientos

Para poder lograr los objetivos mencionados anteriormente se utilizarán diversas herramientas, en la Tabla 1.1 se muestra la relación entre los resultados esperados y las herramientas consideradas para su realización.

Herramientas a utilizar por resultado esperado	
Resultado esperado	Herramienta
RE1: Diagrama de procesos de negocio que contemplen los flujos automatizados de las tareas de gestión de la empresa.	BPMN, Bizagi Process Modeler
RE2: Lista de las aplicaciones basadas en tecnologías web con las cuales se integrará la solución propuesta, justificando de manera adecuada la elección	Estado del arte
RE3: Catálogo de los servicios que la solución propuesta consumirá de las aplicaciones seleccionadas en el objetivo 2, así como los servicios que ofrecerá para que otras aplicaciones se integren con ella.	Matriz de análisis, Postman
RE4: Producto de software final, desarrollado y funcional.	MySql, PHP, Laravel, Mysql Workbench

Tabla 1.1: Resultados esperados y herramientas a usar. Elaboración propia.

1.5.1 Herramientas

En esta sección se listan las herramientas que serán utilizadas para el desarrollo del proyecto.

1.5.1.1 *BPMN*

Business Process Model and Notation es un estándar abierto de notación gráfica para la definición de diagramas de flujo usados para mostrar los procesos de una empresa [METD01]. Mediante las reglas establecidas por el BPMN se pueden definir los procesos de negocio de una manera clara y sólida, que además permita su correcta interpretación y reutilización.

1.5.1.2 *Bizagi Process Modeler*

Herramienta gráfica gratuita que permite la creación de diagramas de procesos utilizando el estándar de notación BPMN [METD02].

1.5.1.3 *PHP*

Hypertext Preprocessor, es un lenguaje de código abierto especialmente adecuado para el desarrollo web, el cual es ejecutado en un servidor [METD03].

1.5.1.4 *Laravel*

Framework PHP de código abierto que utiliza el patrón MVC y permite el desarrollo de aplicaciones web. Entre sus características destacan una sintaxis sencilla, sistema de paquetes y dependencias, acceso simplificado a bases de datos y otras herramientas que agilizan el mantenimiento e instalación de aplicaciones [METD04].

1.5.1.5 *Mysql*

Sistema de administración de base de datos de tipo de relacional que soporta el estándar SQL para consulta y mantenimiento de la información almacenada [METD05].

1.5.1.6 *Mysql Workbench*

Herramienta visual que permite el trabajo con el motor de base de datos Mysql incluye funcionalidades de modelado de datos, desarrollo SQL y administración de la configuración de servidores [METD06].

1.5.1.7 *Postman*

Herramienta de pruebas de servicios tipo REST que funciona como extensión del navegador Google Chrome. Permite la definición de llamadas a servicios web y registro de las respuestas recibidas, agilizando el proceso de creación y consumo de servicios basados en este estándar [METD07].

1.5.2 Métodos y procedimientos

Este proyecto se realizará utilizando una metodología basada en RUP (Rational Unified Process) siendo el principal motivo de esta elección su carácter iterativo, el cual resultará beneficioso dado que es necesario tener avances concretos que el representante de TAU pueda revisar y validar.

Bajo los lineamientos de RUP se realizó la elaboración de diversos documentos que definen de manera clara los aspectos y consideraciones del proyecto. Con la recopilación de los requisitos se pudieron establecer las características del sistema que satisfagan las necesidades del usuario, además se elaboraron los documentos que definen la arquitectura de la solución, así como los lineamientos de desarrollo, tanto del código como de la interfaz gráfica.

1.6 Alcance

Dado que la solución planteada supone dar soporte a los diferentes procesos de gestión de proyectos de TAU a través de la integración de múltiples aplicaciones basadas en tecnologías web, es necesario que se implementen las siguientes funcionalidades:

- Registro y administración de proyectos de desarrollo.
- Registro y versionamiento de cotizaciones enviadas a los clientes.
- Administración del cronograma de entregables, informes, recursos y responsables que conforman un proyecto.
- Generar recordatorios de los hitos de un proyecto a los responsables asignados.

1.7 Justificación

El presente proyecto permitirá mostrar el beneficio de utilizar diversas aplicaciones web de manera conjunta, creando un entorno en el que dichas herramientas puedan comunicarse y trabajar de manera integrada, contribuyendo al objetivo de dar soporte a los procesos relevantes que posee una empresa.

La elaboración del proyecto beneficiará directamente a TAU dado que le proporcionará una herramienta con la cual automatizar ciertas tareas que realiza de manera recurrente y manual, además, el producto brindado servirá de base para la implementación de nuevas funcionalidades que den soporte a otros procesos y áreas de la empresa.

Finalmente, la solución propuesta permitirá al tesista aplicar los conocimientos adquiridos a lo largo de su carrera universitaria con un enfoque orientado a resolver un problema real.

1.8 Viabilidad

El principal factor que determina el éxito de este proyecto es el tiempo disponible para su realización. Durante el curso de Proyecto de Tesis 2 se dispondrá de un tiempo limitado, por lo que el resultado final se encontrará condicionado a una correcta planificación.

Adicionalmente, todas las herramientas y tecnologías necesarias para la implementación del sistema son de libre uso por lo que no se requerirá comprar licencias, además, se tendrá a disposición una gran cantidad de recursos de ayuda y consulta en línea.

Finalmente, dado que el tesista cuenta con experiencia en el uso de la mayoría de herramientas y tecnologías a emplearse, el tiempo de la curva de aprendizaje es mínimo.

1.9 Limitaciones

Existen dos limitaciones críticas para el desarrollo del proyecto:

- Tiempo: dado que se trata de un trabajo de fin de carrera que se realiza en el último ciclo de estudios, el tiempo con el que dispone el tesista es limitado y puede comprometer la integridad del proyecto.
- Coordinación de horarios: ya que es necesario contar con el apoyo de un representante de TAU que valide los avances realizados, se requiere coordinar reuniones entre esta persona y el tesista, lo cual puede ser complicado dada la carga de trabajo presente en una empresa en marcha.

1.10 Riesgos

Dada la naturaleza del proyecto, se han identificado los siguientes riesgos:

Riesgo identificado	Impacto	Medidas para mitigar el impacto
Uno o más de los servicios elegidos cambia sus políticas de uso.	Medio	Elaborar una lista de servicios sustitutos las opciones elegidas.
El tesista renuncia, es despedido o cambia de puesto dentro de la empresa	Grave	Programar el desarrollo del proyecto dentro del tiempo mínimo de estadía del tesista en la empresa definido por el acuerdo de contratación.

Capítulo 2: Marco conceptual y estado del arte

En este capítulo se explican algunos conceptos necesarios para entender la problemática planteada, además, se listan algunos productos existentes que han buscado solucionar de manera parcial o total dicho problema.

2.1 Marco conceptual

2.1.1 Definiciones

La solución propuesta involucra el conocimiento de algunos términos relacionados a la tecnología, los cuales deben ser asimilados para el correcto entendimiento del proyecto. A continuación se detallan dichos términos.

2.1.1.1 *Aplicación web*

Aplicación desarrollada para hacer uso total o parcial de recursos alojados en la web. En general, su uso se encuentra limitado a un navegador web, pero pueden darse casos en los que se accede por medio de una aplicación cliente instalada en el ordenador del usuario [DEF1].

2.1.1.2 *Servicio web*

Componente de software que puede ser accedido desde otra aplicación (como un cliente, un servidor u otro servicio web) a través de protocolos de transporte de alta disponibilidad [DEF2].

2.1.1.3 *Software como servicio (Saas)*

Paradigma de distribución de software en el que las aplicaciones se encuentran instaladas en servidores remotos y se accede a ellas a través de un entorno web, permitiendo que se elimine la complejidad de la instalación y mantenimiento de aplicaciones locales además de la reducción de costos de hardware. Por otro lado, permite un esquema de trabajo remoto ya que el único requisito para utilizar el software es el de contar con una conexión a internet [DEF3]. Entre los principales beneficios de este paradigma se tiene [PAPER2]:

- Ahorro de dinero: al no ser necesario implementar una infraestructura para alojar el software los costos de adquisición y uso se reducen considerablemente.
- Ahorro de tiempo: dado que no se necesita realizar procesos de instalación y configuración.
- Acceso inmediato a actualizaciones: el software tradicional conlleva a un tedioso proceso de actualización que no siempre es factible dado el gran volumen de información que se administra. Por otro lado, con el software en la nube el proceso de actualización es transparente para el usuario, quien no necesita realizar ninguna tarea.

- Formar parte de una comunidad: con el software tradicional se tiene una compra aislada de una licencia, mientras que con el software en la nube se forma parte de una comunidad de usuarios.

2.1.1.4 *Arquitectura orientada a servicios (SOA)*

Conjunto de principios y metodologías que permiten el desarrollo de software en la forma de servicios (componentes que pueden realizar una tarea específica). Estos servicios son construidos de manera que puedan ser reutilizados para diferentes objetivos y por este motivo poseen interfaces de comunicación que permiten el consumo por diversas aplicaciones y plataformas [DEF4].

2.1.1.5 *Middleware*

Software que conecta dos o más aplicaciones que trabajan de manera independiente [DEF]. El término se utiliza para describir productos separados que actúan a modo de integrador o de intermediario entre otras aplicaciones, permitiendo que se deleguen tareas y reduciendo la complejidad de implementación [DEF5].

2.1.1.6 *API*

Application Programming Interface, es un conjunto de rutinas, protocolos y herramientas que permiten construir aplicaciones de software [DEF6]. Una API especifica cómo los deben de interactuar los componentes de software y ofrece los bloques de construcción para desarrollar un programa informático.

2.1.1.7 *HTTP*

Protocolo de comunicación usado en la web, define cómo los mensajes distribuidos y qué acciones deben tomar los servidores y navegadores web ante ciertos comandos [DEF7]. El protocolo define los siguientes métodos para identificar las acciones a realizar:

- GET: solicita la representación de un recurso web, sólo debe recuperar información sin alterarla ni causar otro efecto.
- POST: indica al servidor que acepte la entidad proporcionada en la petición como una nueva asociada al recurso web identificado por la url de la petición.
- PATCH: solicita que la entidad proporcionada en la petición sea almacenada por el servidor, ya sea como una nueva o como la modificación de una ya existente.
- DELETE: indica que el recurso web debe de ser eliminado.

2.1.1.8 *REST*

Representational State Transfer, es un tipo de arquitectura utilizada para diseñar aplicaciones conectadas a través de una red [DEF8]. Se utiliza por protocolo HTTP como mecanismo de transporte y comunicación de datos entre las computadoras.

2.1.1.9 *OAuth2*

Protocolo abierto que permite la autenticación segura de una manera simple y estándar a través de aplicaciones web, móviles y de escritorio [DEF9]. Este mecanismo de autorización permite a aplicaciones de terceros obtener acceso limitado a un servicio HTTP, ya sea en representación del dueño del recurso o de la aplicación misma, a través de un procedimiento de autenticación entre las partes [DEF10].

2.1.2 Descripción de la empresa

TAU es una empresa privada que se dedica al desarrollo, mantenimiento e integración de software, también ofrece servicios de consultoría de sistema de información y capacitaciones sobre diversos temas relacionados a la tecnología [TAU]. Posee una cartera de clientes variada, desde proyectos de emprendimiento hasta empresas con años en el mercado; sin embargo, las solicitudes de servicios se gestionan de la misma manera siguiendo un conjunto de pasos establecidos por la organización.

Cada servicio ofrecido a una empresa se considera un proyecto individual y agrupa todas las actividades involucradas desde la fase de planificación hasta las fases de desarrollo y cierre del proyecto. Durante todas las etapas se realiza un control y gestión de los recursos usados por medio de herramientas basadas en la nube con las cuales se puede realizar el planeamiento del proyecto y almacenar toda la información relevante.

Para la gestión de un nuevo proyecto se tiene un proceso bien definido que permite organizar y preparar el entorno de trabajo y la documentación correspondiente. Este proceso de administración de un nuevo proyecto se divide en diferentes etapas en las cuales el uso de servicios web es indispensable, además, dado que marca el inicio formal del trabajo de TAU con un cliente, se debe tener especial cuidado en recopilar de manera correcta los datos necesarios para la correcta implementación de la solución.

Finalmente, este proceso es fundamental para la empresa y se realiza de manera manual, involucrando el mantenimiento de muchos archivos e información distribuida en diferentes servicios por lo que es necesario plantear una solución que permita optimizar el proceso sin obligar al usuario a cambiar sus aplicaciones en la nube, es decir, generar un entorno que unifique la información distribuida.

2.1.3 Procesos principales

Esta sección presenta los principales procesos de gestión de la empresa, los cuales serán contemplados en el desarrollo de la solución propuesta.

2.1.3.1 *Registro de cotización*

Durante el proceso de negociación con los clientes se preparan diversos documentos con las propuestas elaboradas por la empresa. Estas propuestas contienen información variada sobre los siguientes puntos:

- Alcance del proyecto: define claramente qué es lo que se realizará y cuando se considerará concluido el proyecto.
- Especificaciones técnicas para el desarrollo: en caso se realice una integración o el cliente solicite el uso de una tecnología en particular, en esta sección se incluyen los detalles.
- Costo asociado: indica el precio por el servicio ofrecido, así como detalles del cronograma de pagos.
- Tiempo estimado: indica el plazo en el que se tendrá que desarrollar la solución. También lista la fecha de los entregables parciales a presentarse.
- Otros documentos: en ocasiones se adjuntan otros documentos con el fin de justificar los costos o plazos propuestos.

De este modo, cada cotización lleva consigo múltiples archivos anexos que deben ser correctamente guardados y administrados, además, dada la naturaleza de estos documentos, existe un proceso de negociación que involucra la reelaboración de las cotizaciones, por lo que se debe manejar un historial de cada una de ellas junto con sus anexos. El procedimiento realizado de manera manual es el siguiente:

1. Recopilación: Como primer paso se procede a recolectar y organizar toda la información que será asociada a la cotización, como se mencionó anteriormente puede contener diversos documentos técnicos y comerciales.
2. Almacenamiento: Para guardar los archivos de la cotización se utiliza un servicio de almacenamiento en la nube, en el cual se tiene una estructura de carpetas para almacenar los contenidos. Los nombres de las carpetas ya se encuentran definidos y la integridad de los mismos, así como la organización de los ficheros en su interior son responsabilidad de la persona encargada del registro.
3. Control: Se utiliza un Excel para realizar el seguimiento de las cotizaciones, anotando los archivos que fueron agregados junto con las fechas, ubicaciones en Dropbox y comentarios requeridos para poder realizar un seguimiento posterior.

En la Figura 2.1 se muestra el diagrama del proceso de registro de cotización, contemplando los pasos descritos anteriormente:

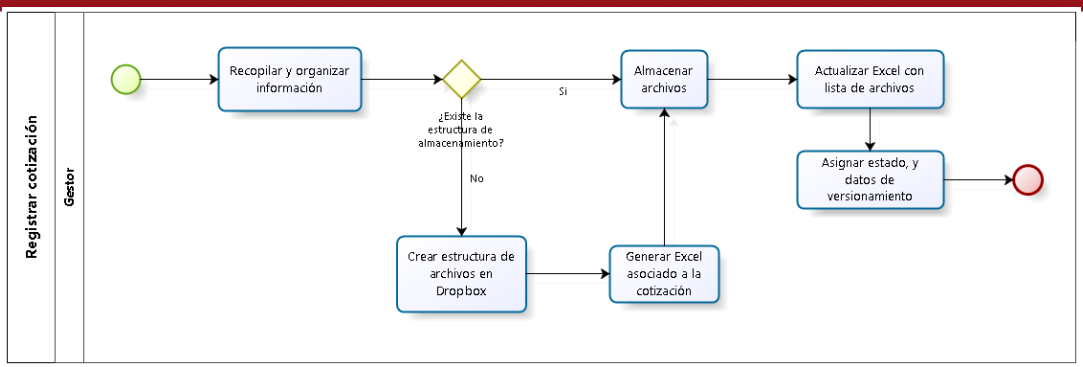

Figura 2.1: Flujo del proceso de registro de cotización. Elaboración propia.

2.1.3.2 *Registro de proyecto*

Luego de evaluar la solicitud del cliente, se establece un nombre para el proyecto y se crea una estructura de carpetas que permita almacenar la información correspondiente. Para realizar un análisis y evaluación del requerimiento y así poder elaborar la cotización se solicita al cliente que proporcione la información que considere relevante. Posterior a este paso de recopilación de datos, se almacenan los diversos medios recibidos en la estructura de carpetas mencionada anteriormente. En la Figura 2.2 se muestra el diagrama con las etapas involucradas en este proceso:

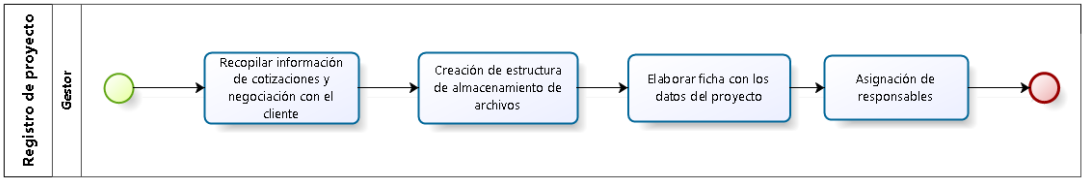

Figura 2.2: Flujo del proceso de registro de proyecto. Elaboración propia.

2.1.3.3 *Creación del proyecto*

Luego de completarse la negociación entre la empresa y el cliente, se procede a registrar formalmente el proyecto. Para esta fase es necesario recopilar toda la información acumulada en pasos anteriores y generar un nuevo grupo de archivos en el cual organizarla. Este procedimiento de migración y creación de archivos se realiza de manera manual y conlleva a una revisión exhaustiva por parte del usuario con el fin de almacenar los datos con un orden correcto. Por otro lado, es necesario definir y almacenar la información de los entregables que conforman la presentación del proyecto final. Los datos relevantes para estos documentos son:

- Fecha de entrega: Indica el día en que se debe presentar un avance del proyecto. No existe un recordatorio automático por lo que el usuario encargado debe realizar un seguimiento de estas fechas.

- Características del entregable: lista de los elementos que deben presentarse, estos pueden ser documentos, código fuente, entre otro tipo de archivos.

En la Figura 2.3 se muestra el flujo realizado para esta etapa de gestión:

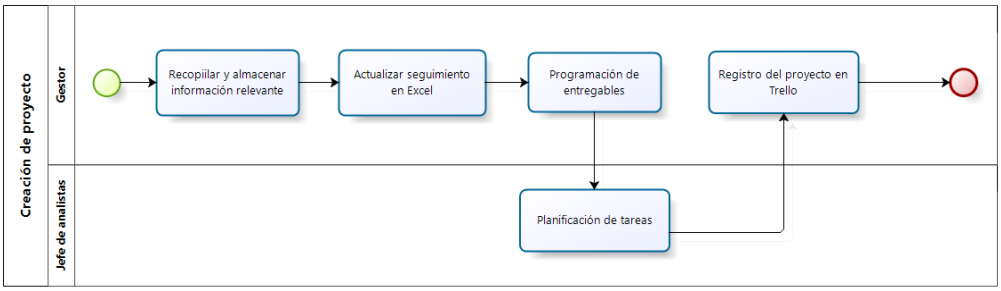

Figura 2.3: Flujo del proceso de creación de un proyecto. Elaboración propia.

2.1.3.4 *Asignación de recursos*

Esta etapa contiene el proceso de organización y asignación de personal a los nuevos proyectos, también contempla la notificación a los responsables por medio de correo electrónico. Este registro se lleva a cabo en hojas de cálculo y se almacena en servicios en la nube. Todos los pasos incluidos son realizados de manera manual por el usuario. El procedimiento actual contempla las siguientes actividades:

1. Selección: luego de determinar qué parte del personal se encargará del desarrollo del proyecto se procede a evaluar la carga de trabajo actual de éstas personas. Para este fin se revisan los archivos de Excel en los cuales se ha registrado la información de tareas y el servicio en la nube en el cual se mantiene el control de tareas y actividades de los proyectos.
2. Asignación: con la lista de personas que participarán en el proyecto se procede a registrarlas en un archivo en Excel. Adicionalmente, se registran los usuarios en el servicio en la nube en el que se administran las tareas y actividades.

En la Figura 2.4 se muestran los pasos comprendidos en esta fase:

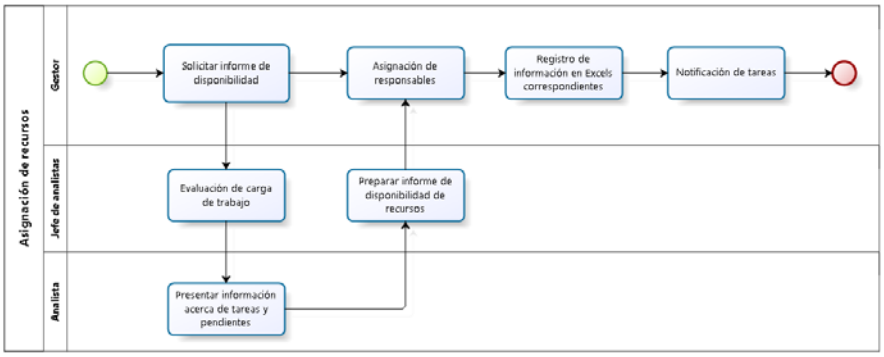

Figura 2.4: Flujo del proceso de asignación de recursos. Elaboración propia.

2.1.3.5 *Desarrollo*

Durante el desarrollo del proyecto se realiza un control de las actividades por medio de una solución basada en la nube. Esta herramienta permite la creación de tareas y asignación de responsables, también provee un medio para vincular recursos y generar listas de actividades a modo de checklist.

2.2 Estado del arte

A continuación se listan y se detallan proyectos existentes que de manera parcial contemplan soluciones para la problemática planteada.

2.2.1 Soluciones tipo suite

Este tipo de soluciones se distribuye como un paquete que incluye múltiples herramientas de productividad, las cuales cubren diferentes aspectos y se encuentran integradas entre sí, creando un entorno en el que la información puede procesarse de manera conjunta.

2.2.1.1 *Google Apps*

Conjunto de herramientas en línea para mensajería y colaboración que satisfacen las necesidades fundamentales de la empresa, incrementan la productividad y reduce costos, todas estas herramientas están hospedadas en la infraestructura de alta seguridad y disponibilidad de Google, no se requiere hardware o software y solo requiere una administración mínima [GAPP1]. Entre los diversos productos en esta suite se encuentran [GAPP2]:

2.2.1.1.1 Gmail

Servicio de correo electrónico con funcionalidades avanzadas de búsqueda, organización y almacenamiento de mensajes. Ofrece un amplio espacio de almacenamiento y una tolerancia alta a largos tiempos de inactividad de la cuenta, además posee funcionalidades inteligentes como la de reconocer palabras claves y comportarse de maneras diferentes, como es el caso de reconocer la palabra “adjunto” o similares y sugerir al usuario que adjunte un archivo al mensaje escrito o el de reconocer la palabra “reunión” y sugerir la creación de un evento en el calendario.

2.2.1.1.2 Calendar

Aplicación de tipo agenda que permite programar eventos y actividades. Cuenta con opciones avanzadas como compartir calendarios con otros usuarios y programar recordatorios para las fechas registradas, además, se encuentra integrado con los demás servicios de modo que es posible sincronizar los contactos de Gmail para incluirlos en los eventos.

2.2.1.1.3 Drive

Servicio en línea que permite el almacenamiento de archivos en la nube, utilizando la infraestructura de Google y ofreciendo acceso desde cualquier dispositivo con conexión a

internet. Permite organizar los archivos en carpetas y agregar usuarios con diferentes permisos sobre dicho esquema de organización además de proporcionar una manera simple de compartir y publicar archivos.

2.2.1.1.4 Docs

Aplicaciones de ofimática ofrecidas como servicios en línea y que cuenta con un editor de texto enriquecido, un procesador de hojas de cálculo y un editor de presentaciones. Además permite la edición de forma colaborativa entre múltiples usuarios con un enfoque en tiempo real, proporcionando actualizaciones sincronizadas.

2.2.1.1.5 Hangouts

Plataforma de mensajería instantánea y llamadas que soporta chats y videoconferencias grupales además de encontrarse integrada con otros servicios como Gmail y Calendar. Por otro lado, almacena de manera automática los historiales de las conversaciones, incluyendo las imágenes o emoticones que hayan sido enviados.

2.2.1.2 *Microsoft OneDrive*

Esta solución permite acceder a versiones web de las principales aplicaciones de tipo Office elaboradas por Microsoft, las cuales son: Word, Excel, PowerPoint y OneNote. Además, cuenta con almacenamiento gratuito en la nube tanto para los documentos generados como para los archivos de imágenes o música que el usuario desee almacenar. Por otro lado, los archivos se mantienen sincronizados y son accesibles desde múltiples aplicaciones desarrolladas para las plataformas móviles más usadas [ODRIVE].

2.2.1.3 *Zimbra Collaboration*

Solución Open Source de mensajería y colaboración construida en la nube [ZIMBRA1]. Los principales elementos que conforman esta suite son [ZIMBRA2]:

- Email: Plataforma de correo electrónico con capacidades de agrupación de mensajes, etiquetado, filtros, envío de mensajes con texto enriquecido, etc.
- Calendario: Ofrece robustas capacidades de organización y repetición de eventos, invitaciones, recordatorios y visibilidad de calendarios.
- Comunicaciones unificadas: Zimbra cuenta con medios de comunicación en tiempo real tales como: Click-2-Call, sistema que reconoce números telefónicos e Instant Messaging, cliente de mensajería accesible desde cualquier sección.

2.2.1.4 *Zoho*

Completa suite de herramientas en línea de carácter empresarial, que permiten controlar los procesos de negocio, el uso de la información y aumentar la productividad de diferentes áreas del negocio. La gran variedad de productos ofrecidos ha sido dividida en los siguientes grupos:

2.2.1.4.1 Para negocios [ZOHO1]

Aplicaciones incluídas:

- Assist: Conexión remota entre diferentes equipos con total seguridad.
- Books: Administración de finanzas.
- BugTracker: Seguimiento del descubrimiento y solución de errores.
- Campaigns: Creador de campañas de correo electrónico.
- ContactManager: Administración de información de contactos.
- Invoice: Administración de cuentas
- Sites: Herramienta de creación de páginas web.
- Survey: Elaboración de encuestas con estadísticas de análisis.
- Vault: Administración de contraseñas.

2.2.1.4.2 Para colaboración [ZOHO2]

Aplicaciones incluídas:

- Chat: Servicio de mensajería instantánea.
- Mail: Cliente de correo electrónico seguro y sin publicidad.
- Meeting: Herramienta para realizar conferencias y reuniones en línea.
- Projects: Administración de análisis y desarrollo de proyectos.
- Wiki: Creación de repositorios de conocimiento.

2.2.1.4.3 Para productividad [ZOHO3]

Se incluye una suite de office completa que además permite la edición colaborativa de documentos. Además, ofrece una herramienta de calendario y agenda integrada a la aplicación Mail.

2.2.2 Soluciones específicas

Existen otro tipo de soluciones que cumplen un único objetivo en particular. A continuación se detallan algunos de ellos:

2.2.2.1 *Dropbox*

Servicio de almacenamiento de archivos en la nube. Su principal uso es el de mantener una copia sincronizada de los archivos del usuario, permitiendo el acceso desde cualquier dispositivo con el cliente de Dropbox instalado, además, posee funcionalidades de compartición de archivos entre diferentes usuarios del servicio, dándoles privilegios de lectura y/o escritura. Por otro lado, provee un historial de versiones de los archivos subidos, lo cual sumado a los precios del servicio y a la facilidad de su uso lo convierten en una de las alternativas más usadas [DBOX].

2.2.2.2 *Trello*

Herramienta colaborativa de tipo Project Management Software que organiza los proyectos en tableros, mostrando las tareas en las que se está trabajando, los responsables junto con sus actividades asignadas y el estado actual de las etapas de desarrollo [TLLO1]. Cada proyecto se muestra como un tablero en el que se colocan listas (conjunto de tareas a realizar) y dentro de cada lista se colocan tarjetas (tareas) pudiendo asignar para cada una diversos responsables junto con plazos de entrega, además, es posible vincular recursos, y ordenar las tarjetas de acuerdo al criterio del usuario.

2.2.2.3 *Evernote*

Aplicación que permite guardar notas en diferentes formatos y mantenerlas sincronizadas en todos los dispositivos del usuario. El contenido de las notas digitales puede ser variado ya que soporta texto, imágenes, videos, entre otros. Cuenta con aplicaciones móviles así como integración con los principales navegadores web del mercado [ENOTE].

2.2.2.4 *Telegram*

Aplicación de mensajería instantánea gratuita y multiplataforma. Permite el envío de mensajes, imágenes, videos, sonidos y archivos de cualquier tipo [TGRM1].

2.2.3 Conclusiones

Luego de realizar una revisión de las soluciones mencionadas párrafos arriba, se puede observar que el único modo en que diversas aplicaciones comparten información y trabajen de manera conjunta es que se encuentren dentro de una misma solución tipo suite. Este comportamiento obliga al usuario a reemplazar sus herramientas actuales y mudarse a algún entorno ya existente, el cual ofrece muchas de las características que el usuario final necesita pero que en muchos casos presentan una curva de aprendizaje.

En la Tabla 2.1 se muestra un cuadro comparativo entre las soluciones de tipo suite mencionadas anteriormente, tomando como criterios las funcionalidades con las que cuentan. Se muestra una “X” en los casos en los que un producto posee la funcionalidad.

Característica / Servicio	Google Apps	Microsoft OneDrive	Zimbra	Zoho
Suite de Office	X	X		X
Versión gratuita		X	X	X
Aplicación móvil	X	X		X

Tabla 2.1: Comparativa de soluciones tipo suite. Elaboración propia.

Almacenamiento de archivos	X	X		X
Herramientas de administración de proyectos				X
Aplicación de calendario	X		X	X
Administración de contactos	X	X	X	X
Interfaz de comunicación externa	X			X

Tabla 2.2: Comparativa de soluciones tipo suite. Elaboración propia.

Finalmente, si bien las suites de aplicaciones cuentan con muchas funcionalidades, el obligar al usuario a migrar su información y cambiar la forma de realizar sus actividades provoca consecuencias negativas. De este modo, la solución propuesta busca integrar las aplicaciones que TAU utiliza actualmente, optimizando su proceso sin obligar a los usuarios finales a migrar completamente a una nueva plataforma.

Capítulo 3: Análisis

Este capítulo abarca los diferentes temas necesarios para analizar los requerimientos del proyecto y así definir el modo en el que se desarrollará la solución.

3.1 Actores del sistema

Tomando en cuenta el análisis de los procesos de la organización se definieron los actores que utilizarán el sistema, los cuales servirán de punto de partida para conocer los detalles requeridos del proyecto. Cada uno de ellos posee una lista específica de características de uso y de acciones a realizar dentro de la aplicación. En la Figura 3.1 se muestran dichos actores:

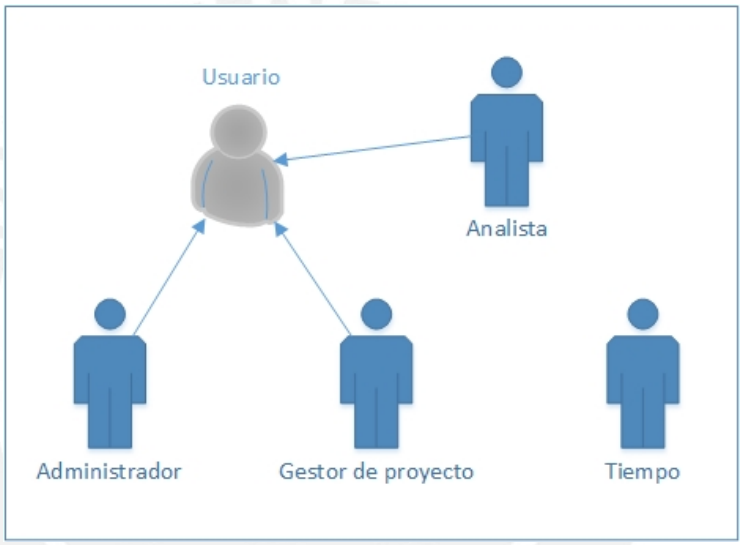

Figura 3.1: Diagrama de actores del sistema. Elaboración propia.

A continuación se describen las características de cada uno de ellos:

3.1.1 Usuario

Toda persona que cuente con credenciales válidas de acceso al sistema.

3.1.2 Administrador

Tipo de usuario que cuenta con todos los privilegios de acceso y administración del sistema.

3.1.3 Gestor de Proyecto

Tipo de usuario que gestiona los proyectos y los recursos asociados a ellos, así como las actividades y plazos de ejecución.

3.1.4 Analista

Persona que participa en el desarrollo de un proyecto y es el encargado de realizar las múltiples tareas requeridas.

3.1.5 Tiempo

Encargado de realizar tareas automáticas de notificación y cambio de estados de las diferentes entidades que participan en el sistema.

3.2 Paquetes del sistema

En esta sección se detallan los paquetes que ayudarán a la mejor organización y análisis de la solución propuesta. Cada paquete agrupa funcionalidades similares y asociadas a una misma entidad.

3.2.1 Paquete de proyectos

Comprende las operaciones dedicadas a la administración, control y mantenimiento de los proyectos registrados en el sistema.

3.2.2 Paquete de cotizaciones

Comprende las operaciones que definen la creación, gestión y versionamiento de las cotizaciones generadas para los diferentes proyectos.

3.2.3 Paquete de usuarios

Comprende las operaciones relacionadas a la administración de los usuarios que acceden al sistema y a la consulta de la información asociada a las tareas realizadas por ellos.

3.2.4 Paquete de actividades

Comprende las operaciones enfocadas en la gestión y visualización de las actividades, eventos y recordatorios de las diferentes entidades del sistema.

En la Figura 3.2 se muestran los paquetes y actores de la solución de modo que sea clara la relación que existe entre ellos y las tareas que le corresponden realizar a cada rol.

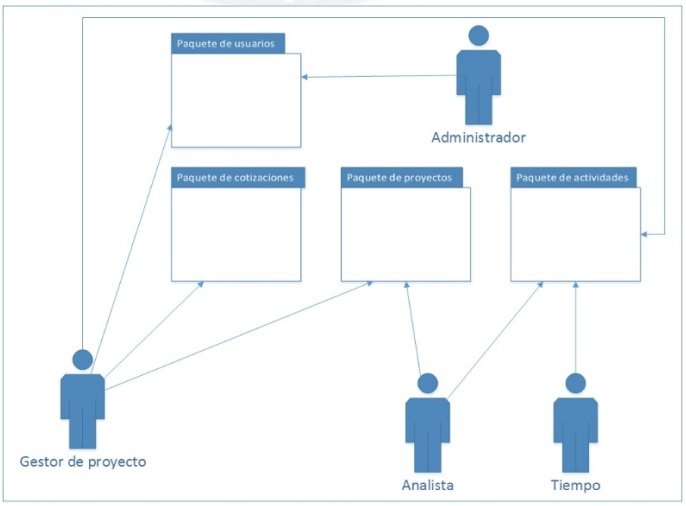

Figura 3.2: Diagrama de actores y paquetes. Elaboración propia.

3.3 Identificación de requerimientos

Luego de levantar la información de los procesos, así como de las funcionalidades que se necesitan en el sistema, se procedió a definir los requerimientos funcionales y no funcionales para la aplicación, agrupados en los paquetes que se lograron definir en la sección anterior.

Para cada requisito se asignó un código en el formato "COD-X", siendo "COD" una cadena de tres caracteres que define el módulo al que pertenece el requisito y "X" el número correlativo asignado. Además, se determinó una prioridad numérica para cada elemento, siendo "1" prioridad alta, "2" prioridad media y "3" prioridad baja. Por otro lado, se definió también un criterio de exigencia, siendo las opciones "Exigible" o "Deseable". En las Tablas 3.1, 3.2, 3.3 y 3.4 se muestran los requisitos funcionales clasificados por los paquetes de proyectos, cotizaciones, usuarios y actividades respectivamente. Por otro lado, en la Tabla 3.5 se encuentran los requisitos no funcionales del sistema.

Paquete de proyectos			
Código	Descripción	Prioridad	Exigencia
PRY-01	El sistema permitirá registrar, modificar y administrar proyectos.	1	Exigible
PRY-02	El sistema permitirá registrar, modificar y administrar los entregables de un proyecto.	1	Exigible
PRY-03	El sistema permitirá asignar un grupo de trabajo al proyecto.	1	Exigible
PRY-04	El sistema permitirá que los miembros del grupo de trabajo asignado puedan realizar comentarios.	2	Exigible
PRY-05	El sistema permitirá registrar, visualizar y responder dudas de los usuarios.	2	Exigible
PRY-06	El sistema permitirá visualizar las tareas asignadas a cada uno de los usuarios que trabajan en el proyecto.	1	Exigible
PRY-07	El sistema permitirá registrar, modificar y visualizar eventos relacionados al proyecto.	2	Exigible
PRY-08	El sistema permitirá asignar una prioridad al proyecto. Las prioridades disponibles serán: Alta, media y baja.	3	Deseable

Tabla 3.1: Requisitos funcionales del paquete de proyectos. Elaboración propia.

Paquete de cotizaciones			
Código	Descripción	Prioridad	Exigencia
COT-01	El sistema permitirá registrar, modificar y anular cotizaciones.	1	Exigible
COT-02	El sistema permitirá asignar un estado a la cotización. Los estados disponibles son: Creada, enviada y anulada.	2	Exigible
COT-03	El sistema permitirá asignar una prioridad a la cotización. Las prioridades disponibles son: Alta, media y baja.	3	Deseable
COT-04	El sistema permitirá registrar, modificar y eliminar comentarios asociados a una cotización.	1	Exigible
COT-05	El sistema permitirá registrar, modificar y eliminar archivos asociados a una cotización.	1	Exigible
COT-06	El sistema mostrará una vista previa de los archivos asociados a una cotización.	3	Deseable
COT-07	El sistema permitirá visualizar las cotizaciones registradas.	1	Exigible
COT-08	El sistema permitirá filtrar y/o ordenar las cotizaciones registradas según la fecha de registro, estado, prioridad y/o proyecto asociado.	1	Exigible
COT-09	El sistema registrará de manera automática un historial de las acciones realizadas sobre una cotización. Las acciones disponibles son: creación, inserción y modificación de comentarios, inserción y modificación de archivos, y edición de información.	3	Deseable
COT-10	El sistema permitirá ingresar el monto asociado a la cotización.	1	Exigible
COT-11	El sistema permitirá ingresar una fecha de expiración para la cotización.	2	Exigible
COT-12	El sistema asignará un número correlativo a cada cotización.	2	Exigible

Tabla 3.2: Requisitos funcionales del paquete de cotizaciones. Elaboración propia.

Paquete de usuarios			
Código	Descripción	Prioridad	Exigencia
USR-01	El sistema permitirá registrar, modificar y deshabilitar usuarios.	1	Exigible
USR-02	El sistema permitirá asignar a cada usuario un nombre de usuario y una contraseña.	1	Exigible
USR-03	El sistema permitirá asignar a cada usuario un rol.	1	Exigible
USR-04	El sistema permitirá registrar para cada usuario su dirección de correo electrónico, su usuario de Trello y su usuario de Telegram.	1	Exigible
USR-05	El sistema permitirá la búsqueda de usuarios.	1	Exigible
USR-06	El sistema permitirá visualizar los proyectos en los que se encuentra vinculado un usuario.	2	Exigible
USR-07	El sistema permitirá visualizar las actividades asignadas a cada uno de los usuarios.	3	Deseable
USR-08	El sistema permitirá visualizar la lista de actividades recientes realizadas por el usuario.	1	Exigible
USR-09	El sistema registrará de manera automática la fecha y hora de último acceso de cada usuario.	3	Deseable
USR-10	El sistema permitirá que sólo los usuarios con el rol “Administrador” editen la información de los usuarios.	2	Exigible

Tabla 3.3: Requisitos funcionales del módulo de usuarios. Elaboración propia.

Paquete de actividades			
Código	Descripción	Prioridad	Exigencia
ACT-01	El sistema permitirá visualizar todos los eventos registrados.	1	Exigible
ACT-02	El sistema permitirá registrar notificaciones para los eventos almacenados.	1	Exigible
ACT-03	El sistema permitirá visualizar las actividades asociadas a un usuario.	2	Exigible

Tabla 3.4: Requisitos funcionales del paquete de actividades. Elaboración propia.

Requisitos no funcionales			
Código	Descripción	Prioridad	Exigencia
NF-01	El sistema debe funcionar en una plataforma web.	1	Exigible
NF-02	El sistema deberá funcionar correctamente en los siguientes navegadores: Chrome 22+, Firefox 24+, Opera 10+ e Internet Explorer 9+	2	Exigible
NF-03	El sistema contará con una versión responsive para dispositivos móviles.	3	Deseable
NF-04	El sistema implementará los servicios web utilizando la arquitectura REST.	1	Exigible
NF-05	El sistema permitirá la ejecución de comandos a través de atajos con el teclado.	3	Deseable
NF-06	El sistema deberá siguiendo patrones de desarrollo que permitan que sea escalable y mantenible.	2	Exigible
NF-07	El servidor donde se ejecute el sistema debe ser alguna distribución de Linux.	1	Exigible
NF-08	El sistema encriptará las contraseñas antes de almacenarlas en la base de datos.	2	Exigible
NF-09	El sistema deberá mostrar mensajes informativos cada vez que se realice una operación.	2	Exigible
NF-10	El sistema deberá utilizar el motor de base de datos MySql.	1	Exigible

Tabla 3.5: Requisitos no funcionales. Elaboración propia.

3.4 Casos de uso

Los casos de uso muestran las acciones que se podrán realizar en el sistema desde el punto de vista del usuario. A continuación se listan todos los casos de uso agrupados según los paquetes definidos en las secciones anteriores, la especificación completa y detallada de cada uno de ellos puede ser encontrada dentro del documento de anexos.

3.4.1 Paquete de proyectos

Se describen los casos de uso vinculados al paquete de proyectos.

3.4.1.1 *Administrar proyecto*

Comprende las funcionalidades de creación, modificación y consulta de un proyecto, incluyendo el ingreso de la información relacionada al cliente, la cotización aprobada y los usuarios que se encontrarán involucrados.

3.4.1.2 *Administrar entregable*

Comprende las funcionalidades de creación, modificación y consulta de los entregables asociados a un proyecto, incluyendo el ingreso de la información relacionada a los contenidos por entregar, la fecha de vencimiento y los responsables del cumplimiento.

3.4.1.3 *Administrar recursos*

Comprende las funcionalidades de creación, modificación y consulta de los recursos asociados a un proyecto, entre los cuales se encuentran los grupos de usuarios y los archivos que contienen información relevante para el proyecto.

3.4.1.4 *Administrar actividades*

Comprende las funcionalidades de creación, modificación y consulta de las actividades asociadas a un proyecto, las cuales organizan las etapas de desarrollo y distribuyen responsabilidades.

3.4.1.5 *Consultar actividades*

Comprende las funcionalidades de consulta de la información de las actividades de un proyecto, incluyendo el listado con sus respectivos estados, plazos de entrega y responsables asignados.

En la Figura 3.3 se muestra la relación entre los casos de uso de este paquete y los actores.

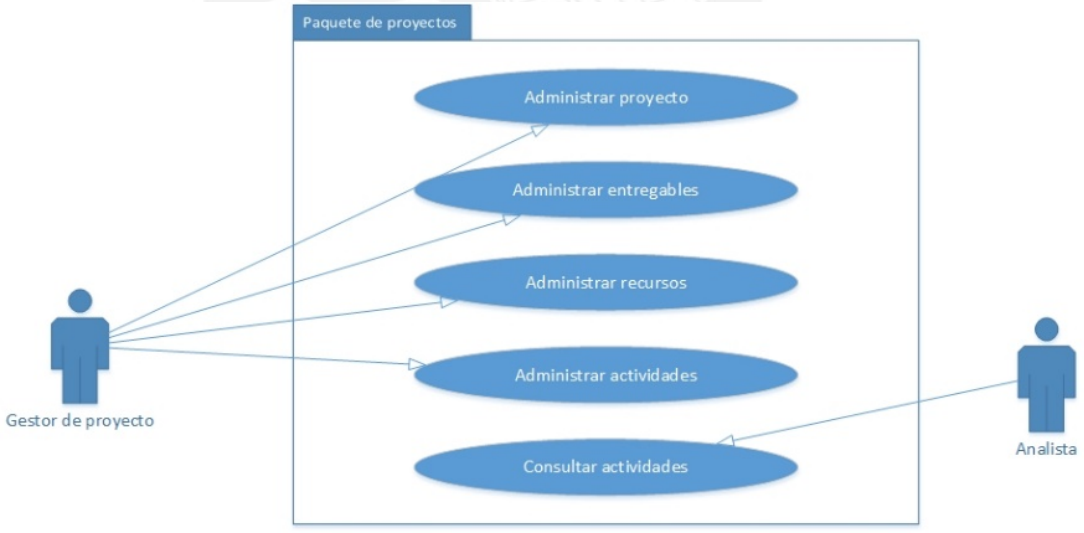

Figura 3.3: Diagrama de casos de uso del paquete de proyectos. Elaboración propia.

3.4.2 Paquete de cotizaciones

Se describen los casos de uso vinculados al paquete de cotizaciones.

3.4.2.1 *Administrar cotizaciones*

Comprende las funcionalidades de creación, modificación y consulta de una cotización, incluyendo la vinculación con un proyecto, el monto asignado, fecha de expiración, entre otros datos relevantes.

3.4.2.2 *Buscar cotizaciones*

Comprende las funcionalidades de búsqueda y ordenamiento de cotizaciones utilizando múltiples criterios definidos por el usuario.

3.4.2.3 *Consultar cotización*

Comprende las funcionalidades de consulta de una cotización específica, incluyendo información sobre el estado, los comentarios realizados y los archivos incluidos.

3.4.2.4 *Aprobar cotización*

Comprende la funcionalidad de aprobación de una cotización, lo cual inicia un procedimiento interno de organización de la información que finalmente permite el inicio de un proyecto.

3.4.2.5 *Administrar comentarios*

Comprende las funcionalidades relacionadas a la creación y modificación de los comentarios asociados a cada cotización, los cuales describen consideraciones tomadas en cuenta durante la negociación con el cliente.

3.4.2.6 *Administrar archivos*

Comprende las funcionalidades relacionadas a la creación y modificación de los archivos asociados a cada cotización, así como los comentarios que se encuentran vinculados a ellos y que sirven como descripción.

En la Figura 3.4 se muestra la relación entre los casos de uso de este paquete y los actores.

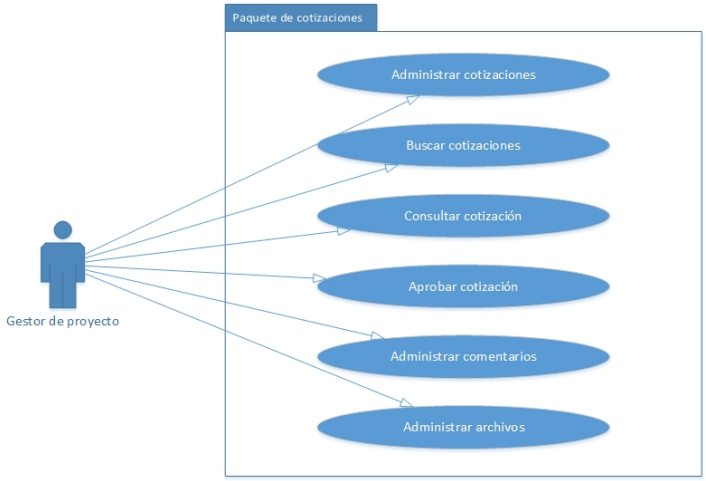

Figura 3.4: Diagrama de casos de uso del paquete de cotizaciones. Elaboración propia.

3.4.3 Paquete de usuarios

Se describen los casos de uso vinculados al paquete de usuarios.

3.4.3.1 *Administrar usuarios*

Comprende las funcionalidades de creación, modificación y consulta de los usuarios del sistema, incluyendo información como los nombres de usuario en los diversos servicios web utilizados.

3.4.3.2 *Administrar grupos de usuarios*

Comprende las funcionalidades de creación, modificación y consulta de los grupos de usuarios del sistema, concepto que permitirá la rápida asociación de múltiples personas a un proyecto.

En la Figura 3.5 se muestra la relación entre los casos de uso de este paquete y los actores.

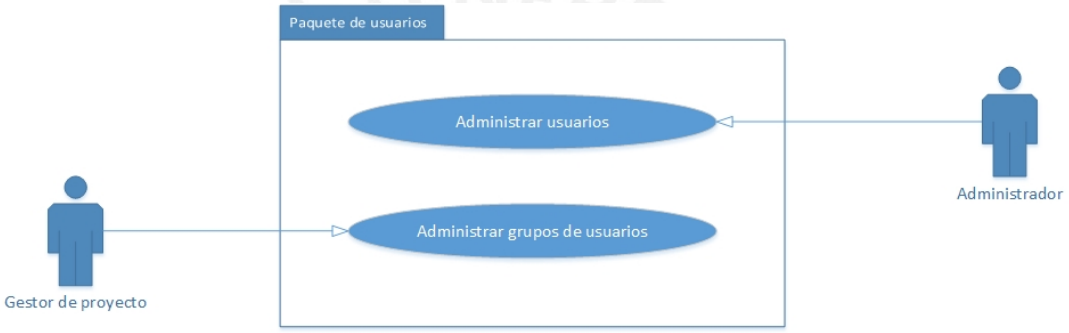

Figura 3.5: Diagrama de casos de uso del paquete de usuarios. Elaboración propia.

3.4.4 Paquete de actividades

Se describen los casos de uso vinculados al paquete de actividades.

3.4.4.1 *Administrar eventos*

Comprende las funcionalidades de creación y modificación de eventos en el sistema, los cuales pueden encontrarse vinculados a los proyectos y cotizaciones.

3.4.4.2 *Administrar recordatorios*

Comprende las funcionalidades de creación y modificación de recordatorios para los diferentes eventos registrados en el sistema.

3.4.4.3 *Visualizar eventos*

Comprende las funcionalidades de consulta de los eventos de un usuario, indicando las fechas de realización, otras personas involucradas e información relevante asociada.

3.4.4.4 *Enviar recordatorios*

Comprende las funcionalidades de notificación de eventos próximos a ocurrir a los usuarios vinculados, así como la configuración del mecanismo a través del cual se realizará la alerta.

En la Figura 3.6 se muestra la relación entre los casos de uso de este paquete y los actores.

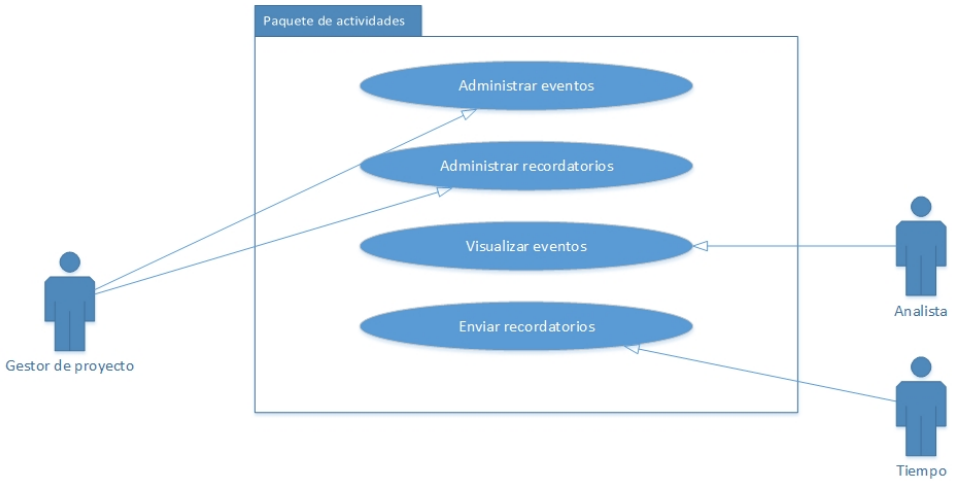

Figura 3.6: Diagrama de casos de uso del paquete de actividades. Elaboración propia.

Capítulo 4: Servicios web

En el presente capítulo se describen los servicios web con los que se integrará el sistema propuesto, además se mencionan las características de comunicación y forma de trabajo de cada uno de ellos, junto con la manera en la que serán utilizados para cumplir los requerimientos planteados.

4.1 Catálogo de servicios web empleados en la integración

A través de la elaboración del estado del arte así como de la recopilación de información proveniente de TAU, se realizó la evaluación y selección de las alternativas más adecuadas para el proyecto. Se consideró que las opciones que se evaluarán cumplan con las siguientes características:

- Actualmente son utilizadas por TAU en sus principales procesos.
- Proveen funcionalidades específicas para los requerimientos.
- Permiten que un sistema externo se comunique y administre la información que contienen.
- Cuentan con una licencia de uso que no limita la funcionalidad requerida.

A continuación se presenta la lista de servicios web a utilizar, incluyendo también la lista de métodos a consumir de cada uno. La especificación completa de los métodos de cada servicio se puede encontrar dentro del documento de anexos.

4.1.1 Trello

En la actualidad esta herramienta es utilizada por TAU en sus procesos de control de proyectos, además posee un gran número de funcionalidades que ayudan a la gestión y manejo de las múltiples actividades que se realizan en la empresa. Para su funcionamiento se describen los conceptos que conforman este servicio [TLLO2]:

- Organización: Define un grupo de personas que trabajarán de manera conjunta para la realización de un proyecto.
- Tablero: Se utiliza para representar un proyecto y permite la colaboración de diferentes usuarios con el fin de realizar un conjunto de tareas de manera organizada.
- Lista: Mantiene organizado un conjunto de tarjetas en las diferentes fases de progreso.
- Tarjeta: Unidad fundamental de un tablero que se utiliza para representar tareas o ideas.

En la Figura 4.1 se muestra la interfaz de Trello y se especifican los elementos básicos de administración mencionados anteriormente.

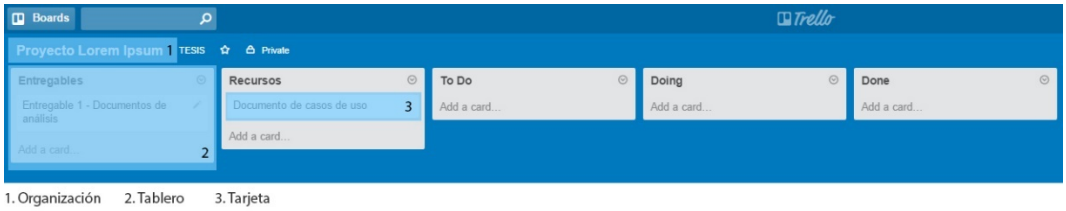

Figura 4.1: Principales elementos de Trello. Elaboración propia.

Este mecanismo de control de actividades es particularmente útil dada la naturaleza de los proyectos y el tipo de gestión que requieren, las diferentes listas permiten organizar de una manera eficiente las tareas y recursos necesarios para lograr un objetivo.

Además, cada organización cuenta con un grupo de usuarios definidos que pueden ser asignados de manera independiente a cada tarjeta, de modo que las responsabilidades puedan ser repartidas de forma que no haya interferencias entre los usuarios.

Por otro lado, cada tarjeta puede contener comentarios, archivos, etiquetas y fechas de expiración, permitiendo una gestión avanzada de cada ítem y un buen control del desarrollo dadas las funcionalidades de seguimiento de actividades. Estas características serán usadas por la solución propuesta a través de llamadas a la API REST y permitirán integrar su información con las tareas que se realizarán en el sistema.

En la Tabla 4.1 se muestran los métodos del servicio Trello que se utilizarán para la integración, así como los requerimientos funcionales a los cuales da soporte [TLLO3].

Métodos a utilizar de Trello			
Método (URL)	Verbo	Descripción	Requisitos funcionales
boards/{id}	GET	Obtiene la información de un tablero con un id en particular.	PRY-01, PRY-02, PRY-03, PRY-05, PRY-06
boards/{id}/cards	GET	Obtiene las tarjetas de un tablero con un id en particular.	
boards/{id}/members	GET	Obtiene los miembros de un tablero con un id en particular.	
boards/{id}/lists	GET	Obtiene las listas de un tablero con un id en particular.	

Tabla 4.1: Métodos a utilizar de Trello. Elaboración propia.

boards	POST	Crea un nuevo tablero.	
lists/{id}	GET	Obtiene la información de una lista con un id en particular.	
lists/{id}/cards	GET	Obtiene las tarjeta de una lista con un id en particular.	
lists	POST	Crea una nueva lista.	
cards/{id}	GET	Obtiene la información de una tarjeta con un id en particular.	
cards/{id}/attachments	GET	Obtiene los archivos adjuntos de una tarjeta con un id en particular.	
cards/{id}/members	GET	Obtiene los miembros asignados a una tarjeta con un id en particular.	
cards	POST	Crea una nueva tarjeta.	
cards/{id}	DELETE	Elimina una tarjeta con un id en particular.	

Tabla 4.2: Métodos a utilizar de Trello. Elaboración propia.

4.1.2 Drive

Como se mencionó en el capítulo 2, la alternativa de almacenamiento de archivos usada por la organización es Dropbox debido a que fue seleccionada en un inicio cuando las herramientas web empleadas aún no eran parte de la plataforma de Google Apps. Ya que en la actualidad se utilizan servicios de Google como Gmail y Calendar se propone el cambio de Dropbox a Drive de modo que la mayor parte de los servicios usados por la organización tengan el mismo proveedor, facilitando la integración entre ellos.

Utilizar Drive de Google creará un entorno más uniforme de trabajo que al mismo tiempo cumpla con los requisitos del proyecto, dadas sus características de versionamiento de archivos y opciones de colaboración. Por otro lado, desde el punto de vista técnico permitirá una implementación más ágil ya que los estándares de autenticación y comunicación serán los mismos que los usados por las otras opciones seleccionadas.

En la Tabla 4.2 se muestran los métodos del servicio Google Drive que se utilizarán para la integración, así como los requerimientos funcionales a los cuales da soporte [GAPP3].

Métodos a utilizar de Google Drive			
Método (URL)	Verbo	Descripción	Requisitos funcionales
files	GET	Obtiene la lista de archivos de un usuario.	COT-05, COT-06, PRY-01, PRY-02
files/{id}	GET	Obtiene la información de un archivo con un id en particular.	
files	POST	Inserta un nuevo archivo.	
files/{id}	PATCH	Actualiza la información de un archivo con un id en particular.	
files/{id}	DELETE	Elimina de manera permanente un archivo con un id en particular.	
files/{id}	POST	Crea una copia de un archivo con un id en particular.	

Tabla 4.3: Métodos a utilizar de Google Drive. Elaboración propia.

4.1.3 Calendar

Esta alternativa de Google permite administrar eventos en un calendario y es usada por la organización para programar diversas actividades como reuniones y presentaciones [TAU]. Actualmente el uso dado es limitado a las opciones más básicas pero con la solución propuesta se llegará a un nivel de integración que utilice de mejor manera las funcionalidades de recordatorios e invitaciones para dar soporte a los procesos que contemplará la solución.

En la Tabla 4.3 se muestran los métodos del servicio Google Calendar que se utilizarán para la integración, así como los requerimientos funcionales a los cuales da soporte [GAPP4].

Métodos a utilizar de Trello			
Método (URL)	Verbo	Descripción	Requisitos funcionales
calendars/{id}/events	POST	Crea un nuevo evento en un calendario con un id en particular.	ACT-02, PRY-07
calendars/{id}/events	GET	Obtiene la lista de eventos de un calendario con un id en particular.	
calendars/{id}/events/{idEvento}	PATCH	Actualiza la información de un evento con un id en particular.	
calendars/{id}/events/{idEvento}	DELETE	Elimina un evento con un id en particular.	

Tabla 4.4: Métodos a utilizar de Google Calendar. Elaboración propia.

4.1.4 Gmail

Usado como motor de correo electrónico y presente dentro de la comunicación de la empresa, se utilizará también como parte de la integración del proyecto con sistemas web. En particular, las funcionalidades de envío de correo, archivos adjuntos (que trabajará de manera conjunta con la herramienta Drive) y administración de contactos [GAPP5].

En la Tabla 4.4 se muestran los métodos del servicio Gmail que se utilizarán para la integración, así como los requerimientos funcionales a los cuales da soporte.

Métodos a utilizar de Gmail			
Método (URL)	Verbo	Descripción	Requisitos funcionales
users/{id}/messages/send	POST	Envía un correo electrónico desde la cuenta de un usuario con un id en particular.	ACT-02, USR-01, USR-04, PRY-03
users/{id}/drafts	POST	Crea un borrador de correo electrónico desde la cuenta de un usuario con un id en particular.	

Tabla 4.5: Métodos a utilizar de Gmail. Elaboración propia.

4.1.5 Telegram

Esta aplicación permite la mensajería instantánea entre dispositivos móviles y fue seleccionada para agilizar la comunicación entre los participantes de un proyecto. Como se mencionó en el capítulo 2, algunos recordatorios y mensajes intercambiados por los usuarios a modo de recordatorio o coordinación se realizan de manera variada, utilizando diferentes opciones en el mercado como correo electrónico, WhatsApp e incluso mensajes de texto. Es por esto que se plantea utilizar Telegram como estándar de comunicación en tiempo real utlizando las ventajas de Smartphones, siguiendo con el modelo de trabajo que la empresa utiliza.

Entre los beneficios de esta aplicación destacan [TGRM1]: la seguridad del software ya que utiliza una capa de encriptación personalizada, características comunes de clientes de mensajería, como chats grupales, envío de imágenes y archivos, entre otros; clientes para diversos sistemas operativos y principalmente un concepto llamado “Bot”, que será el mecanismo utilizado para integrar la aplicación con el sistema propuesto y hacer uso de la información relevante.

Un bot, dentro de la terminología de Telegram, es una cuenta operada por Software en lugar de una persona [TGRM2], y permite la interacción con usuarios reales a través de comandos específicos. Esta funcionalidad a través de comandos permitirá el envío de la información del

sistema de modo que los usuarios puedan revisar rápidamente ciertos aspectos de los proyectos en los cuales se encuentran trabajando.

El modo en que un bot puede ser utilizado es a través de una conversación, es decir, invitándolo a un chat que también puede ser también grupal. Al estar incluido en la conversación este componente interpreta los mensajes recibidos y al detectar ciertos comandos predefinidos procederá a enviar la información a una URL definida por el creador de dicho bot, una vez allí se pueden realizar todas las operaciones necesarias y devolver un mensaje al cliente de la aplicación [TGRM3]. En la figura 4.2 se muestran dos ventanas de Telegram en las que se tiene una conversación con un Bot. En el lado izquierdo, el chat muestra los comandos a los cuales puede responder el Bot mientras que al lado derecho se muestra la respuesta al ejecutar cada uno de dichos comandos disponibles.

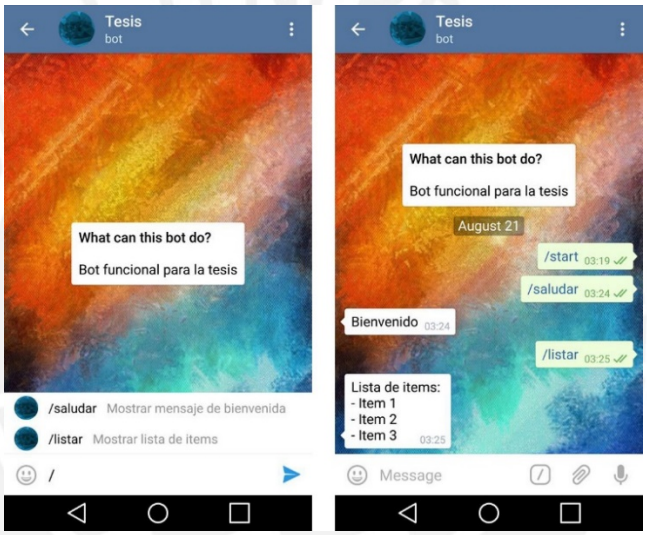

Figura 4.2: Ejemplo de interacción con un Bot de Telegram. Elaboración propia.

En la Tabla 4.5 se muestran los métodos del servicio Telegram que se utilizarán para la integración, así como los requerimientos funcionales a los cuales da soporte [TGRM4].

Métodos a utilizar de Trello			
Método (URL)	Verbo	Descripción	Requisitos funcionales
sendMessage	POST	Envía un mensaje de texto a un chat activo.	ACT-01, ACT-02, ACT-03
sendPhoto	POST	Envía una imagen a un chat aactivo.	
sendDocument	POST	Envía un archivo a un chat activo.	
sendLocation	POST	Envía una ubicación de mapa a un chat activo.	

Tabla 4.6: Métodos a utilizar de Telegram. Elaboración propia.

Finalmente, en la Tabla 4.6 se muestra la relación que existe entre los requerimientos funcionales del sistema y los servicios web seleccionados. Para fines de presentación solo se han incluido los requerimientos asociados a los servicios y no aquellos que son propios del proyecto.

Servicios a utilizar por requisito					
Requisito/Servicio	Trello	Gmail	Calendar	Drive	Telegram
PRY-01	X			X	
PRY-02	X				
PRY-03	X				
PRY-05					X
PRY-06	X				X
PRY-07	X		X		
COT-05				X	

Tabla 4.6: Matriz de servicios web VS requerimientos funcionales. Elaboración propia.

COT-11			X		X
USR-04	X	X	X	X	X
USR-07	X				
ACT-01	X	X	X		
ACT-02		X	X		X
ACT-03	X		X		

Tabla 4.7: Matriz de servicios web VS requerimientos funcionales. Elaboración propia.

4.2 Catálogo de servicios expuestos por la solución

Para que la solución propuesta funcione correctamente se requiere que exponga un servicio web con ciertos métodos que ayuden a satisfacer los requerimientos planteados en el capítulo anterior. En la Tabla 4.7 se muestra la lista de los métodos necesarios, la especificación técnica completa se puede encontrar en el documento de anexos.

Métodos que la solución propuesta debe exponer		
Método (URL)	Verbo	Descripción
cotizaciones/estados	GET	Obtiene la lista de estados disponibles para una cotización.
cotizaciones/estados/{id}	GET	Obtiene la información de un estado de cotización con un id en particular.
cotizaciones	GET	Obtiene la lista de cotizaciones.

Tabla 4.7: Matriz de servicios web ofrecidos por la solución. Elaboración propia.

cotizaciones	POST	Crea una cotización.
cotizaciones/{id}	GET	Obtiene la información de una cotización con un id en particular
cotizaciones/{id}	PATCH	Actualiza la información de una cotización con un id en particular.
cotizaciones/{id}	DELETE	Elimina una cotización con un id en particular.
cotizaciones/{id}/ comentarios	GET	Obtiene la lista de comentarios de una cotización con un id en particular.
cotizaciones/{id}/ comentarios	POST	Inserta un nuevo comentario para una cotización con un id en particular
cotizaciones/{id}/ comentarios/ {idComentario}	PATCH	Actualiza la información de un comentario con un id en particular.
cotizaciones/{id}/ comentarios/ {idComentario}	DELETE	Elimina un comentario con un id en particular.
cotizaciones/{id}/ archivos	GET	Obtiene la lista de archivos de una cotización con un id en particular.
cotizaciones/{id}/ archivos	POST	Inserta un nuevo archivo para una cotización con un id en particular
cotizaciones/{id}/ archivos / {idArchivo}	PATCH	Actualiza la información de un archivo con un id en particular.
cotizaciones/{id}/ archivos /{idArchiv}	DELETE	Elimina un archivo con un id en particular.
usuarios/estados	GET	Obtiene la lista de estados disponibles para un usuario.
usuarios/estados/{id}	GET	Obtiene la información de un estado de usuario con un id en particular.
usuarios/roles	GET	Obtiene la lista de roles disponibles para un usuario.
usuarios/roles/{id}	GET	Obtiene la información de un rol de usuario con un id en particular.
usuarios/grupos	GET	Obtiene la lista de grupos de usuarios.

Tabla 4.7: Matriz de servicios web ofrecidos por la solución. Elaboración propia.

usuarios/grupos	POST	Crea un nuevo grupo de usuarios.
usuarios/grupos/{id}	GET	Obtiene la información de un grupo de usuario con un id en particular.
usuarios/grupos/{id}	PATCH	Actualiza la información de un grupo de usuario con un id en particular.
usuarios/grupos/{id}	DELETE	Elimina un grupo de usuario con un id en particular.
usuarios	GET	Obtiene la lista de usuarios.
usuarios/{id}	GET	Obtiene la información de un usuario con un id en particular.
usuarios	POST	Crea un usuario.
usuarios/{id}	PATCH	Actualiza la información de un usuario con un id en particular.
usuarios/{id}	DELETE	Elimina un usuario con un id en particular.
usuarios/{id}/ proyectos	GET	Obtiene la lista de los proyectos en los que está registrado el usuario.
usuarios/{id}/tareas	GET	Obtiene la lista de tareas asignadas a un usuario con un id en particular.
actividades	GET	Obtiene la lista de actividades y eventos.
actividades	POST	Crea una actividad.
actividades/{id}	GET	Obtiene la información de una actividad con un id en particular.
actividades/{id}	PATCH	Actualiza la información de una actividad con un id en particular.
actividades/{id}	DELETE	Elimina una actividad con un id en particular.
actividades/{id}/ recordatorios	GET	Obtiene los recordatorios asignados a una actividad con un id en particular.
actividades/{id}/ recordatorios	POST	Crea un recordatorio para una actividad con un id en particular.
actividades/{id} recordatorios	PATCH	Actualiza la información del recordatorio de una actividad con un id en particular.
proyectos	GET	Obtiene la lista de proyectos.
proyectos	POST	Crea un proyecto.

Tabla 4.7: Matriz de servicios web ofrecidos por la solución. Elaboración propia.

proyectos/{id}	GET	Obtiene la información de un proyecto con un id en particular.
proyectos/{id}	PATCH	Actualiza la información de un proyecto con un id en particular.
proyectos/{id}/ entregables	GET	Obtiene la lista de entregables de un proyecto con un id en particular.
proyectos/{id}/ entregables	POST	Crea un entregable para un proyecto con un id en particular.
proyectos/{id}/tareas	GET	Obtiene la lista de tareas para un proyecto con un id en particular.
proyectos/{id}/tareas	POST	Inserta una nueva tarea en un proyecto con un id en particular.
proyectos/{id}/ consultas	GET	Obtiene la lista de consultas de un proyecto con un id en particular.
proyectos/{id}/ consultas/ {idConsulta}	POST	Responde a una consulta con un id en particular.
proyectos/{id}/ miembros	GET	Obtiene la lista de miembros asignados a un proyecto con un id en particular.
proyectos/{id}/ recursos	GET	Obtiene la lista de recursos asociados a un proyecto con un id en particular.
proyectos/{id}/ recursos	POST	Crea un nuevo recurso para un proyecto con un id en particular.
proyectos/{id}/ recursos/{idRecurso}	PATCH	Actualiza la información de un recurso con un id en particular
bot	POST	Encargado del procesamiento del Bot de Telegram

Tabla 4.8: Matriz de servicios web ofrecidos por la solución propuesta. Elaboración propia.

4.3 Estándares de integración

En esta sección se especifican los estándares que permitirán la correcta comunicación y trabajo entre los múltiples servicios independientes elegidos.

4.3.1 Comunicación

Todas las alternativas seleccionadas exponen servicios REST a través de los cuales se puede acceder a la información que contienen y poder realizar operaciones de administración y modificación. Para las llamadas a estos servicios se utiliza el protocolo HTTP y cada petición realizada contiene las siguientes partes:

- URL de la petición.
- Verbo HTTP (GET, POST, PUT, DELETE) que define el tipo de operación a realizar.
- Parámetros de autenticación y permisos.
- Información adicional asociada a la operación.

Por otro lado, el formato de respuesta de estas peticiones es un objeto JSON que puede ser interpretado por el cliente para conocer si la la operación se realizó correctamente o si ocurrió algún error, además de poder recibir información adicional en caso haya sido solicitada. En la Figura 4.3 se muestra como ejemplo una llamada a la API REST de Trello con el propósito de obtener la información de un tablero en específico.

Figura 4.3: Ejemplo de llamada a una API. Elaboración propia.

Por otro lado, en la Figura 4.4 se muestra la respuesta del servicio para la petición mencionada anteriormente y que contiene la información solicitada en formato JSON.


```
{
  "id": "5575154461a48ce27f42082c",
  "name": "Tesis",
  "desc": "",
  "descData": null,
  "closed": false,
  "idOrganization": "5575153cd86557d4b624e4d9",
  "pinned": false,
  "url": "https://trello.com/b/D8cO7LXq/tesis",
  "shortUrl": "https://trello.com/b/D8cO7LXq",
  "prefs": {
    "permissionLevel": "org",
    "voting": "disabled",
    "comments": "members",
    "invitations": "members",
    "selfJoin": true,
    "cardCovers": true,
    "cardAging": "regular",
    "calendarFeedEnabled": false,
    "background": "blue",
    "backgroundColor": "#0079BF",
    "backgroundImage": null,
    "backgroundImageScaled": null,
    "backgroundTile": false,
    "backgroundBrightness": "unknown",
    "canBePublic": true,
    "canBeOrg": true,
    "canBePrivate": true,
    "canInvite": true
  },
  "labelNames": {
    "green": "",
    "yellow": "",
    "orange": "",
    "red": "",
    "purple": "Prioridad alta!",
    "blue": "",
    "sky": "",
    "lime": "",
    "pink": "",
    "black": ""
  }
}
```


Figura 4.4: Ejemplo de respuesta de una API. Elaboración propia.

4.3.2 Autenticación

Para que las peticiones realizadas a los servicios REST sean aceptadas es necesario enviar junto a la información de la llamada un token de seguridad proporcionado por la misma aplicación. Este token valida la identidad del solicitante y los permisos que tiene asignados para trabajar sobre los datos almacenados.

A excepción de la API de Telegram, el estándar de seguridad para generar los token de seguridad es OAuth2, que como se menciona en el capítulo 1, permite a aplicaciones de terceros obtener acceso a la información de un usuario sin necesidad de que éste proporcione su contraseña. El flujo de información para la autorización a través de OAuth2 se muestra en la Figura 4.5.

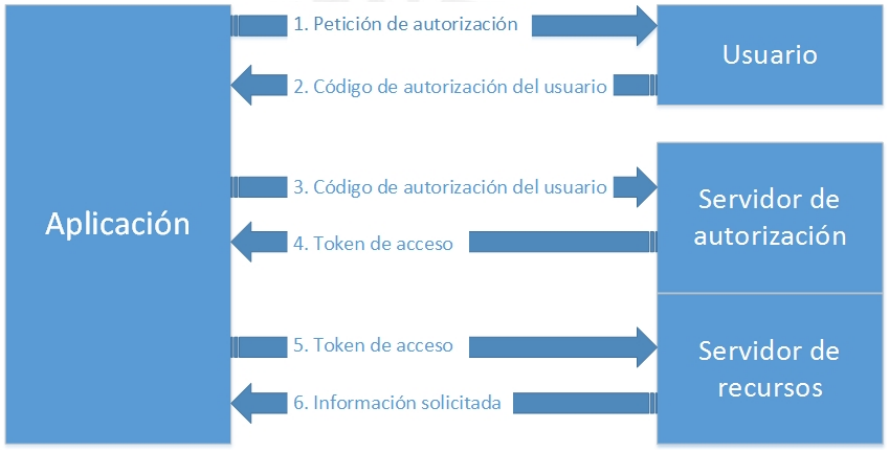

Figura 4.5: Flujo de información del estándar OAuth2. Elaboración propia.

El proceso exacto que ocurre durante la autenticación de la aplicación es el siguiente:

1. La aplicación solicita al usuario el acceso a sus recursos protegidos
2. Si el usuario autoriza la petición, la aplicación recibe un código que valida que su petición fue aceptada.
3. La aplicación se comunica con el servidor de autorización presentando el código de autorización recibido del usuario.
4. Si la identidad de la aplicación y el código proporcionado son correctos entonces el servidor de autorización construye un token de seguridad para la aplicación. En este punto la autorización se completa.
5. La aplicación se comunica con el servidor de recursos (que almacena la información del usuario) presentando el token de seguridad recibido.
6. Si el token es válido entonces el servidor de recursos devuelve la información solicitada por la aplicación.

Además, el token adquirido es almacenado de modo que pueda ser utilizado por el sistema en peticiones posteriores.

Por otro lado, en el caso de Telegram no se utiliza el estándar OAuth2 sino un procedimiento de autenticación propio. Primero es necesario crear un Bot, para lo cual se debe abrir un chat con el contacto llamado “BotFather”, el cual es un bot desarrollado por el mismo Telegram y que permite realizar todas las tareas relacionadas a la creación y mantenimiento de este componente [TGRM04]. En la Figura 4.6 se observa al lado izquierdo una ventana de chat con “BotFather”, en el cual da la bienvenida y ofrece una lista de comandos disponibles. Entre los comandos se encuentra “/newbot” el cual inicio el proceso guiado de creación de un bot. Al lado derecho se observa el resultado de ejecutar dicho comando.

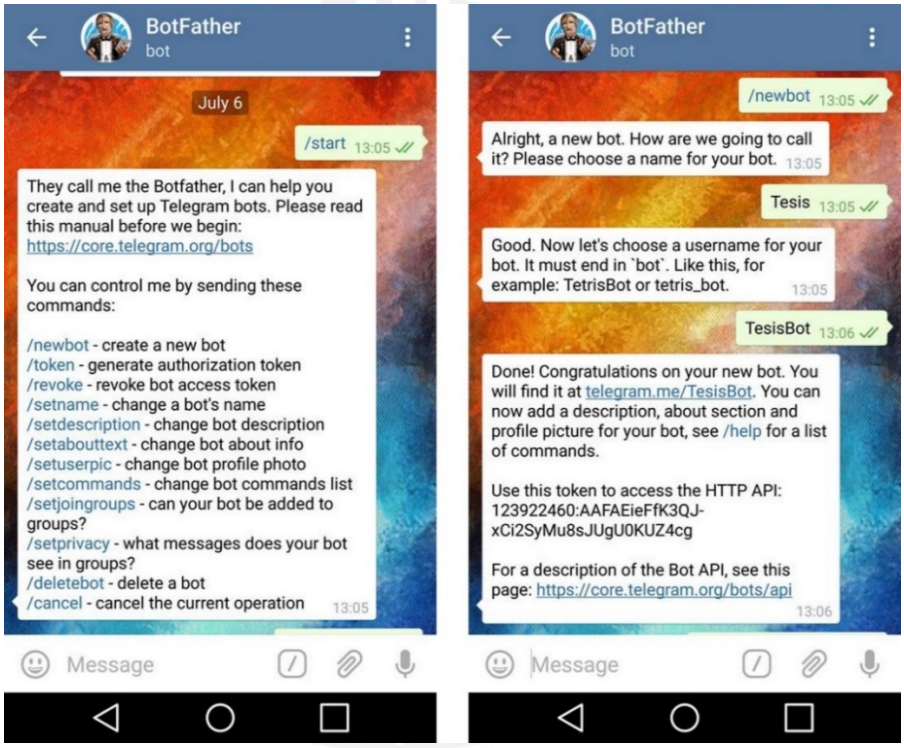

Figura 4.6: Creación y autenticación de un Bot de Telegram. Elaboración propia.

Como resultado de esta operación, el servicio de Telegram retorna un token que permite la comunicación con el servicio REST que controla las acciones que puede realizar el bot y que serán utilizadas para integrarlo con el sistema propuesto.

4.4 Procedimiento de integración

Para todos los casos se sigue un flujo similar para realizar la integración con las API, el proceso se resume en los siguientes pasos:

1. Creación de una aplicación: a través de la consola de desarrolladores de los servicios web se puede crear una aplicación y con ello recibir un código identificador que puede

ser utilizado para demostrar la identidad del desarrollador y así solicitar permisos al usuario.

2. Solicitud de permiso de lectura y escritura: utilizando el protocolo OAuth2 o alguno de los mecanismos de autenticación necesarios según el tipo de API se procede a solicitar al usuario los permisos para administrar su información. Este procedimiento de solicitud de permisos debe realizarse de manera independiente para cada uno de los servicios incluidos en el proyecto.
3. Ejecución de peticiones: con el token de seguridad apropiado se procede a realizar la comunicación con los servicios REST a través del protocolo HTTP, enviando también el verbo que define la operación (GET, POST, etc).

En la Figura 4.7 se muestra el flujo de consumo de un servicio web, el cual consiste en realizar una petición HTTP con un cierto verbo a la url que define el método a utilizar. Dentro del paquete de la petición se incluye el token de seguridad y otros datos que requiere el servicio para funcionar correctamente. Luego de que el servidor procese la orden, responderá con un objeto JSON que contiene información de la operación.

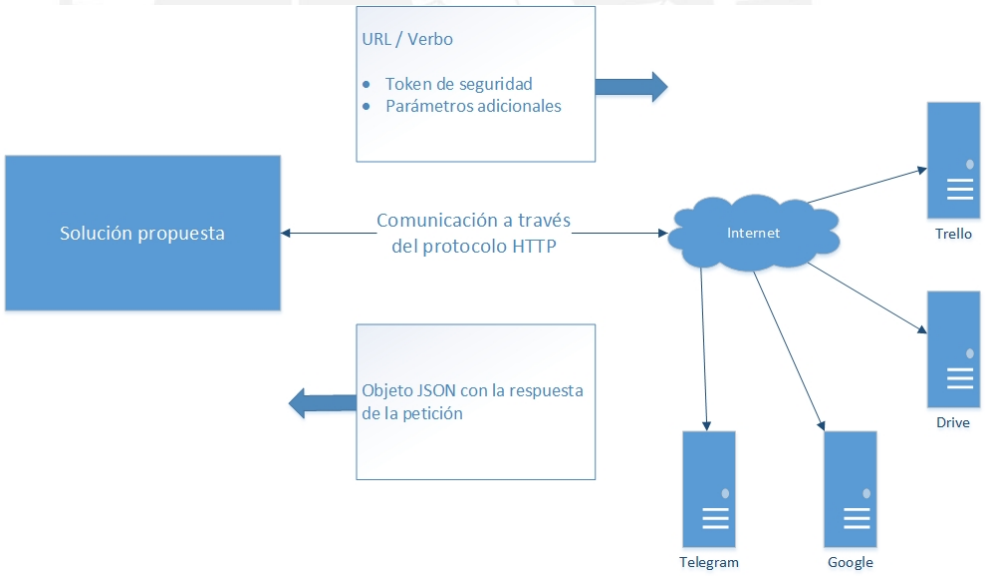

Figura 4.7: Diagrama de comunicación. Elaboración propia.

Capítulo 5: Diseño

El presente capítulo define las consideraciones tomadas para el diseño de la solución propuesta, abarcando desde la organización de los componentes que conforman el sistema hasta las especificaciones gráficas del mismo.

5.1 Arquitectura del sistema

El presente proyecto funcionará en una plataforma web cuya arquitectura interna se encontrará definida según las necesidades identificadas el capítulo de análisis. Dado que se necesita interactuar con múltiples servicios web y al mismo tiempo dar soporte a la lógica de negocio específica de la organización, se plantea manejar un componente que actúe como middleware entre los servicios web de los diferentes proveedores, integrando la funcionalidad entre ellos y estandarizando sus modos de trabajo, además, este componente encapsulará también parte de la lógica de negocio que cubre los requerimientos de la solución propuesta y que requiere las funcionalidades de los servicios de terceros. Siguiendo una arquitectura basada en servicios, este componente expondrá una variedad de funcionalidades a través de una API, la cual podrá ser consumida por otras aplicaciones que deseen hacer uso de esta plataforma favoreciendo la escalabilidad y flexibilidad de la solución.

La principal aplicación que consumirá la API mencionada anteriormente será el sistema desarrollado para TAU, que a través de los servicios expuestos por el middleware implementará el resto de la lógica de negocio incluyendo también la interfaz con la cual interactuarán los usuarios. En la Figura 5.1 se muestra la representación de la arquitectura final del sistema:

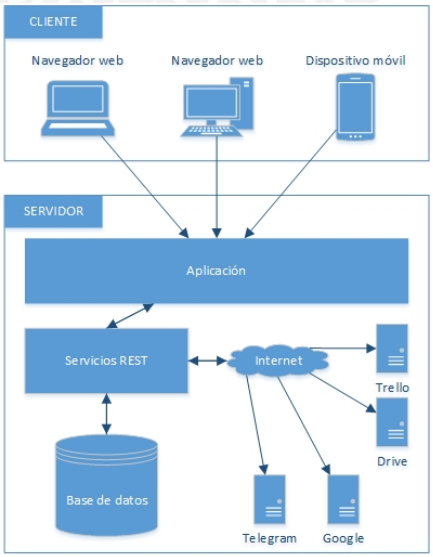

Figura 5.1: Diagrama de arquitectura del sistema. Elaboración propia.

Se pueden observar tanto los clientes desde los cuales se utilizará el sistema y el servidor encargado de procesar las acciones. Este segundo componente maneja diversas capas que ayudan a organizar las responsabilidades:

- Capa de lógica de negocio: contiene las funcionalidades propias de los procesos de la empresa y trabaja de manera conjunta con el middleware a fin de lograr satisfacer los requerimientos planteados.
- Capa de la API REST o middleware: encargada de interactuar con los múltiples servicios web con los cuales se integrará la solución propuesta y de exponer los servicios propios de modo que puedan ser utilizados por otros componentes.
- Capa de persistencia: cumple el rol de interactuar con la base de datos y realizar la consulta y persistencia de datos.

5.2 Diagrama de base de datos

El modelo de datos debe permitir que toda la información requerida por el sistema sea guardada de manera ordenada y que las relaciones entre las diversas entidades se representen correctamente. La estructura planteada debe favorecer las múltiples consultas que va a realizar la aplicación de forma recurrente y guardar también valores clave para la comunicación con los servicios web del catálogo.

Para la elaboración del modelo se empleó la herramienta MySql Workbench y se obtuvo el diagrama final que se muestra en la Figura 5.2, la descripción completa de cada una de las tablas se encuentra en el documento de Anexos. Dado el gran número de tablas que conforman el diagrama se procedió a agrupar dichas entidades según la función que cumplen, asignando a cada grupo un color diferente. Los grupos y sus respectivos colores son los siguientes:

- Proyectos (celeste): Contiene la información de los proyectos almacenados en el sistema, así como los campos que relacionan esta entidad con los componentes del servicio web de Trello.

- Entregables (marrón): Contiene la información relevante de los entregables asociados a un proyecto y la relación existente con los componentes del servicio web de Trello.

- Cotizaciones (naranja): Contiene la información de las cotizaciones almacenadas en el sistema incluyendo las referencias a los archivos y comentarios asociados, así como las características de su versionamiento.

- Usuarios (oro): Contiene la información de los usuarios registrados en el sistema y que formarán parte de las diversas actividades contempladas en los requisitos, además, almacena la información relevante de los nombres de usuario en las diferentes plataformas web que serán usadas en la integración

- Grupos de usuarios (verde): Contiene la información de los grupos de usuarios creados en el sistema y que serán asignados al proyecto dentro del servicio web de Trello.

- Archivos (rojo): Contiene la información relacionada a los archivos que administra el sistema, además, guarda la relación existente entre el archivo y el servicio de almacenamiento de Google Drive.

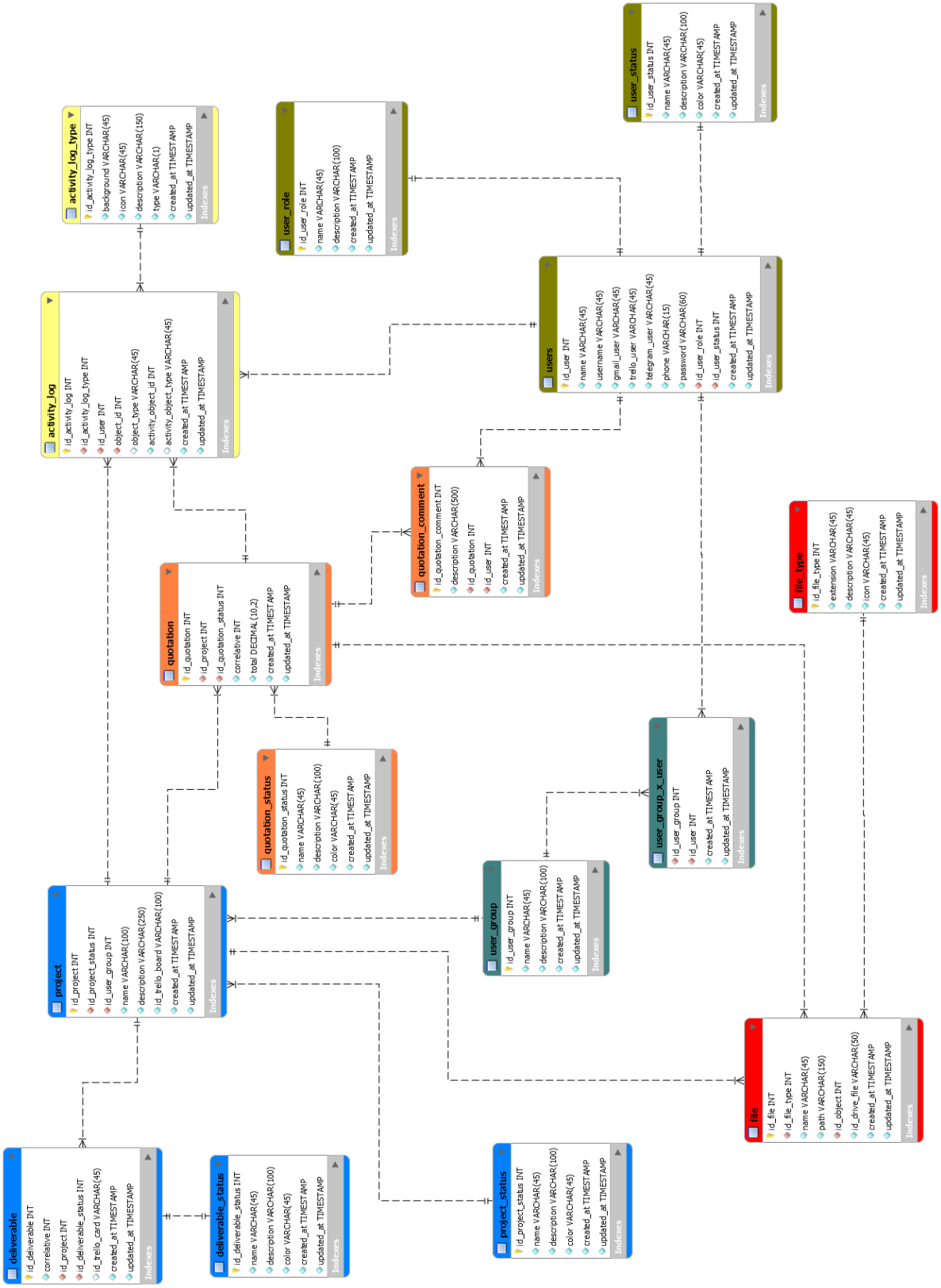

Figura 5.2: Modelo de base de datos. Elaboración propia.

5.3 Interfaz gráfica

En esta sección se definirán los criterios empleados para la elaboración de las interfaces del sistema de información, además se indicarán las consideraciones para un correcto desarrollo.

5.3.1 Estándares

A continuación se indican los lineamientos empleados:

- Todas las operaciones que realice el sistema deberán mostrar un mensaje informativo con el resultado, ya sea de éxito o error.
- Los formularios en los cuales el usuario deba ingresar información deberán validar los campos requeridos.
- Todos los estados de las diferentes entidades disponibles deberán contar con un color único que permita su rápida identificación.
- Las operaciones de edición de datos deberán realizarse dentro de la misma página de la entidad, a través de ventanas emergentes o popups.

5.3.2 Diseño de pantallas

En esta sección se muestran algunas capturas de pantalla de las páginas del sistema, las cuales presentan la línea gráfica y el orden de los elementos. Todas las pantallas del sistema, a excepción de la página de inicio de sesión, siguen la misma distribución de contenidos, la cual se puede apreciar en la Figura 5.3.

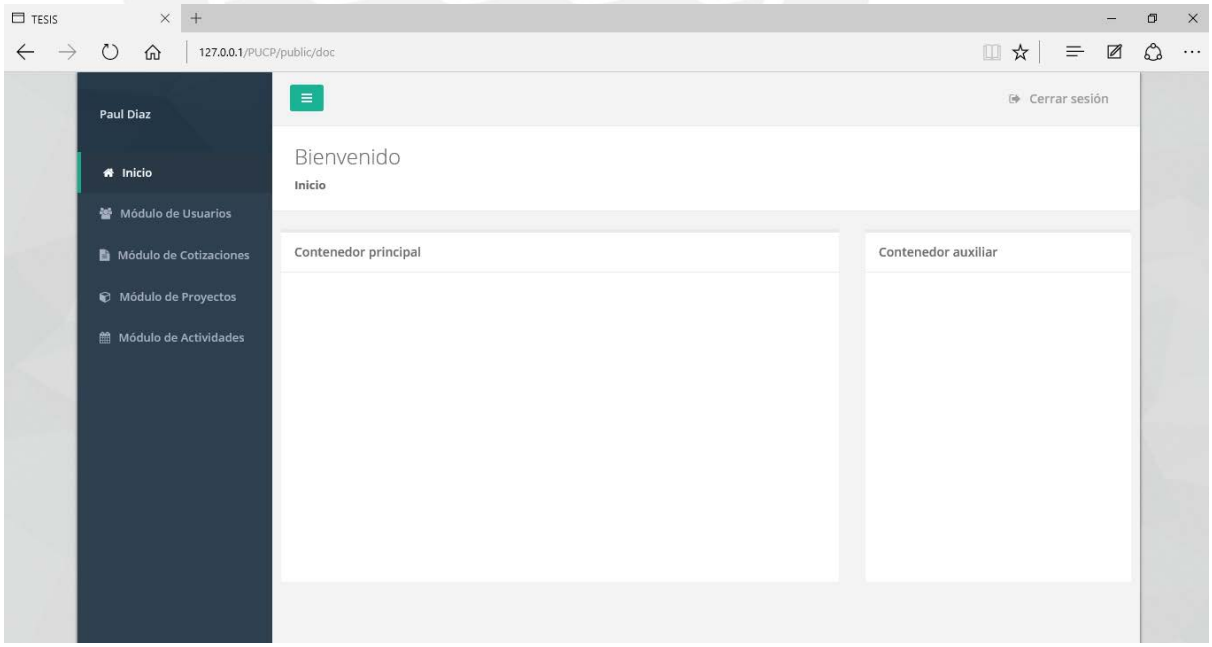

Figura 5.3: Distribución de elementos del sistema. Elaboración propia.

Al lado izquierdo se encuentra el menú principal del sistema, el cual muestra el nombre de usuario y la lista de accesos a los diferentes módulos. Al lado derecho se encuentra el botón

para cerrar la sesión actual y también el contenido principal de la página actual. Dicho contenido consta de los siguientes elementos:

- Título: Informa al usuario de manera clara en qué sección del sistema se encuentra.
- Navegador: En algunos módulos habrá subsecciones por lo que el navegador permite mantener el orden de las secciones a las cuales ha accedido el usuario.
- Contenedor principal: Muestra la información relevante de la página actual y todos los datos con los cuales interactuará el usuario.
- Contenedor auxiliar: Usado en algunas secciones en las que es necesario mostrar información adicional que complemente lo que se encuentre en el contenedor principal.

Por otro lado, siguiendo los estándares de diseño se cuenta con mensajes informativos para el usuario, los cuales aparecen por un periodo corto de tiempo o hasta que el usuario los cancele manualmente. En la Figura 5.4 se muestran los mensajes de error y de éxito planteados:

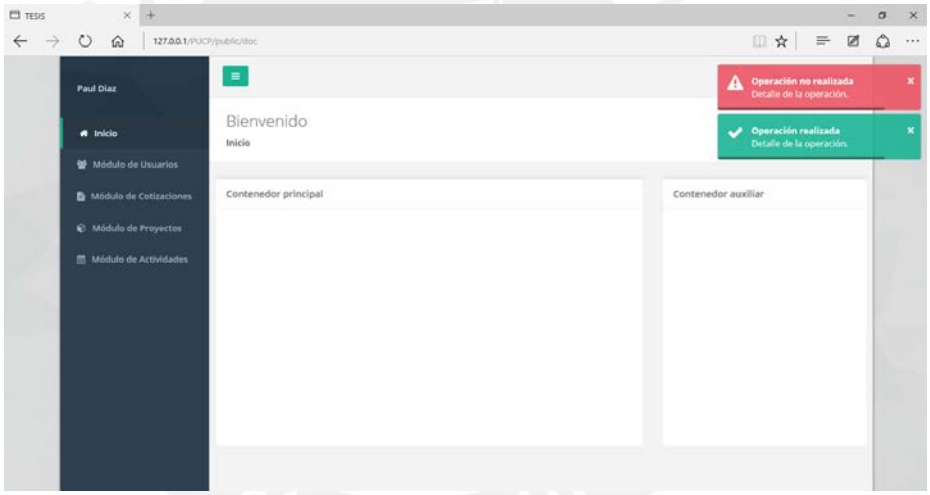

Figura 5.4: Mensajes informativos para el usuario. Elaboración propia.

Otro elemento recurrente dentro de las vistas es el de ventanas emergentes (popups), usados para realizar ciertas acciones adicionales. En la Figura 5.5 se muestra el aspecto de dichos popups, los cuales se posicionan sobre el resto de contenido usando una superposición de fondo oscuro semitransparente. Su contenido consta de tres partes: Título informativo, contenido y botones de acción, entre los cuales siempre estará presente el botón cancelar, encargado de cerrar la ventana y mostrar el contenido normal de la página.

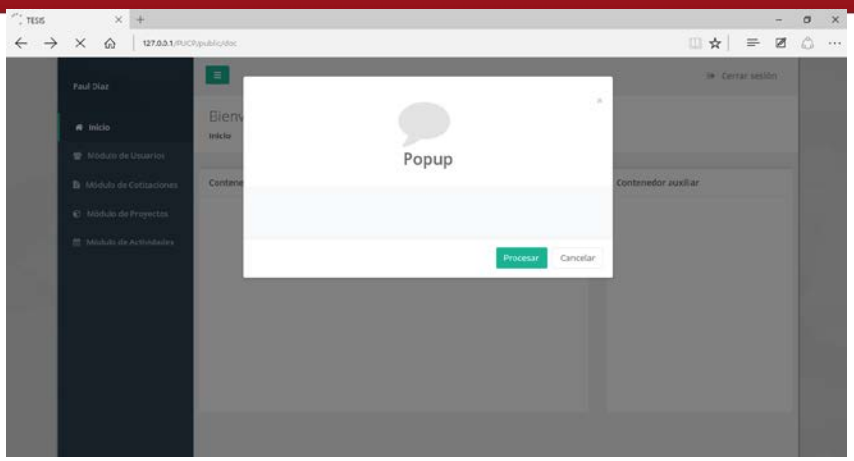

Figura 5.5: Popup adicional de operaciones. Elaboración propia.

5.3.3 Diseño final de las pantallas

Tomando en cuenta los estándares definidos anteriormente, así como los requisitos que debe cumplir el proyecto, se procedió a realizar el diseño final que permite al usuario interactuar con el sistema y realizar sus tareas de una manera organizada. A continuación, se muestran las pantallas que conforman el procedimiento de registro de una cotización, etapa que inicia la gestión de un proyecto.

5.3.3.1 *Registro de un proyecto*

La ventana mostrada en la Figura 5.6 permite la creación de un proyecto, en esta primera etapa sólo se requiere un nombre y una descripción.

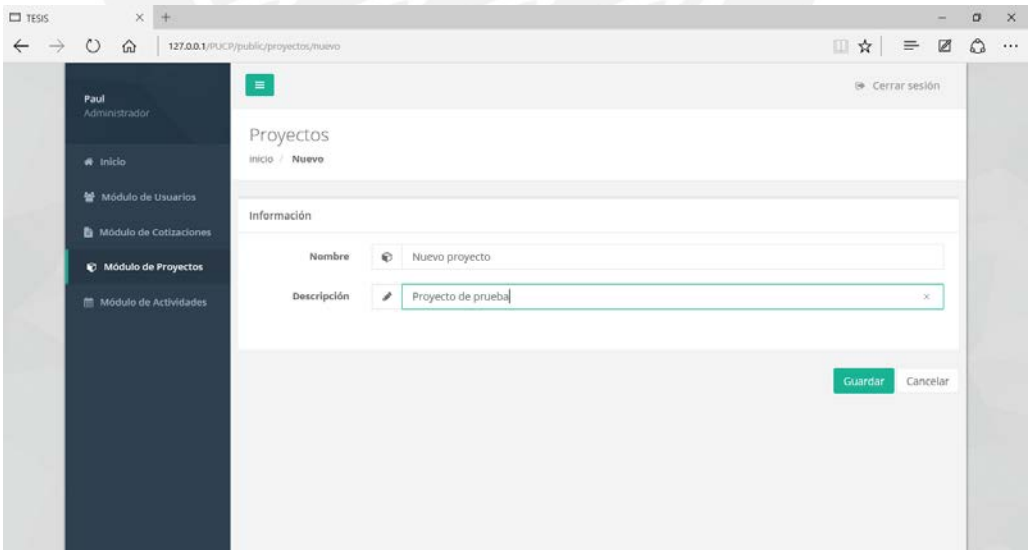

Figura 5.6: Pantalla de creación de proyecto. Elaboración propia.

5.3.3.2 *Registro de cotización*

Con un proyecto existente, se procede a crear una cotización asociada a dicho proyecto en particular. La Figura 5.7 muestra la pantalla de creación de una cotización.

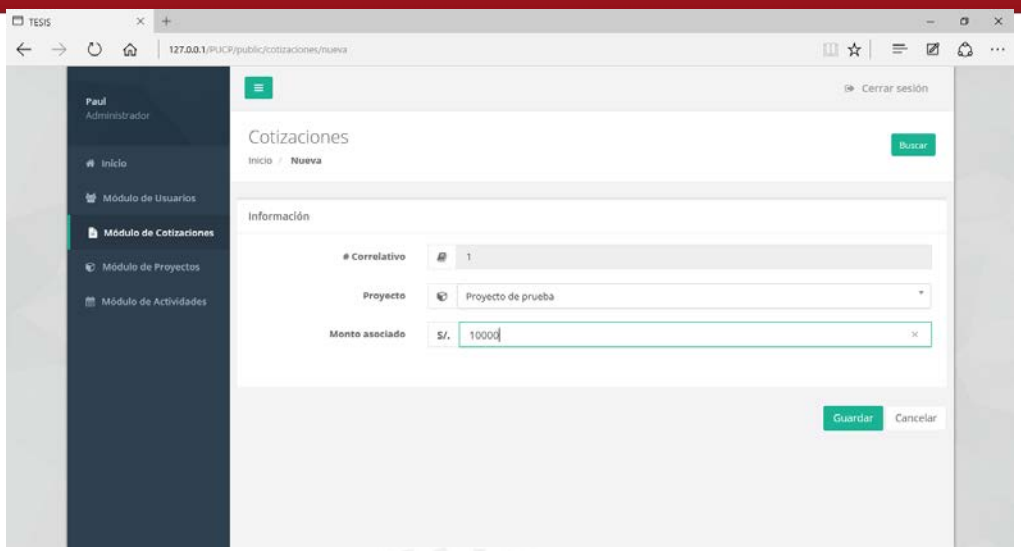

Figura 5.7: Pantalla de creación de cotización. Elaboración propia.

5.3.3.3 *Mantenimiento de cotizaciones*

Para una cotización de un particular, se pueden registrar tanto comentarios como archivos, estas tareas se realizan a través de ventanas emergentes que se muestran al interactuar con los botones de “Nuevo comentario” y “Nuevo archivo”. En la Figura 5.8 se muestra al lado izquierdo el popup de registro de un nuevo comentario y a la derecha el popup de registro y carga de un nuevo archivo.

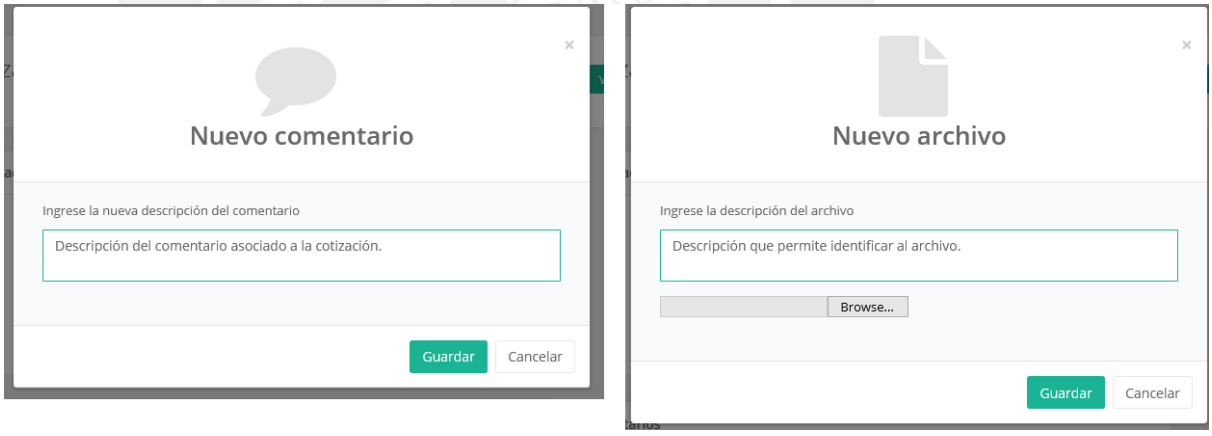

Figura 5.8: Ingreso de comentarios y archivos de una cotización. Elaboración propia.

Al finalizar el ingreso de la información asociada a una cotización se muestran los registros en la página de detalle de la cotización. Se cuenta con botones que permiten editar y eliminar cada uno de los ítems de manera independiente. En la Figura 5.9 se muestra la ventana de detalle de cotización con todas las opciones disponibles.

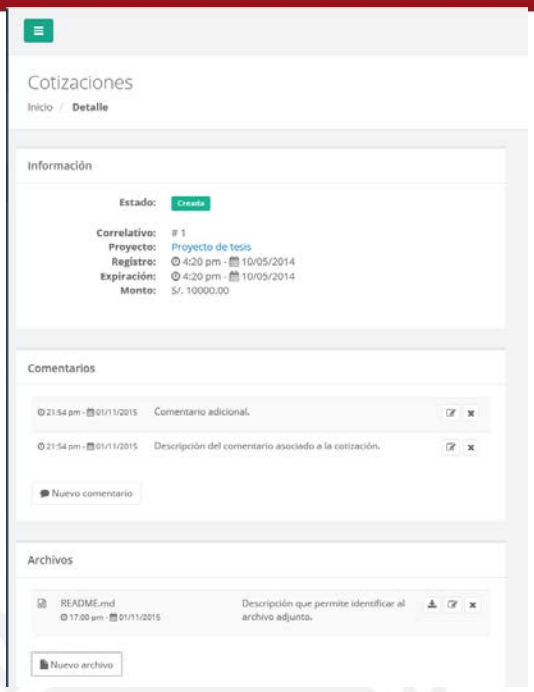

Figura 5.9: Pantalla de detalle de cotización. Elaboración propia.

5.3.3.4 *Detalle de un proyecto*

La Figura 5.10 muestra la ventana de detalle para un proyecto seleccionado, incluyendo un resumen de los elementos asociados a su gestión. Se presenta una cabecera con la información principal y un componente de tipo pestañas que muestran un resumen del estado actual. Se tienen listados los usuarios asignados al proyecto, los entregables registrados, las cotizaciones emitidas y adicionalmente, los incidentes y preguntas registradas a través del servicio de gestión de tareas de Trello.

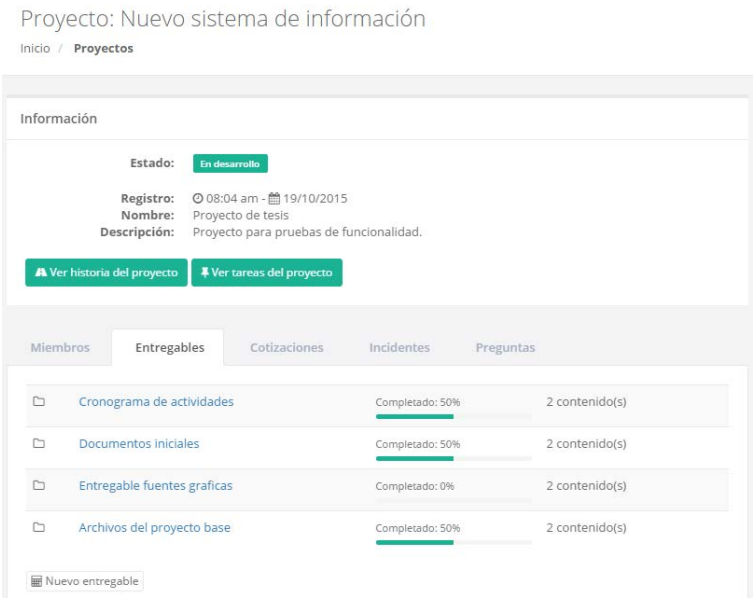

Figura 5.10: Pantalla de detalle de proyecto. Elaboración propia.

Cada elemento vinculado al control del proyecto, como los entregables o cotizaciones, cuenta con una vista dedicada que muestra en detalle su información. En la Figura 5.11 se muestra el detalle de uno de los entregables asociados.

Figura 5.11: Pantalla de detalle de un entregable. Elaboración propia.

Por otro lado, se puede visualizar el resumen de las actividades asignadas al proyecto a través del gestor Trello. En la Figura 5.12 se muestra la ventana que corresponde al visor de actividades en la cual se muestran todas las tareas registradas para el proyecto actual, filtradas y catalogadas por prioridades y fechas de creación. Además, se da la opción de visualizar dicha información en conjunto o segmentado según los usuarios a cargo del desarrollo.

Figura 5.12: Pantalla de tareas de un proyecto. Elaboración propia.

Capítulo 6: Construcción

En este capítulo se detallan los múltiples aspectos involucrados en el desarrollo de la solución web propuesta, además, se detalla la forma en que contribuyeron las herramientas seleccionadas, el proceso de desarrollo y las consideraciones tomadas en cuenta para concluir la implementación.

6.1 Herramientas

Dada la naturaleza web del proyecto, se selecciona el lenguaje de programación PHP para su desarrollo. Entre las ventajas ofrecidas por este lenguaje se encuentran la facilidad y rapidez con la cual se puede instalar, codificar y migrar el producto desarrollado [CNST1]. Además, posee una gran comunidad que lo respalda y un tiempo considerable en el mercado, lo cual se puede traducir en estabilidad y confianza. Por otro lado, ya que estructurar el código del proyecto y desarrollar todas las funcionalidades requeridas utilizando únicamente PHP sería una tarea complicada y repetitiva, se decidió utilizar un framework que agilice y ayude la implementación. La alternativa seleccionada es Laravel, dado que es un marco de trabajo que permite el desarrollo rápido y conciso con PHP, al mismo tiempo que trabaja bajo estándares modernos de las tecnologías web y proporciona utilidades a modo de plugins y herramientas que contribuyen a un mejor desarrollo [METD04].

Ya que la solución propuesta requiere implementar servicios web utilizando la arquitectura de tipo REST, se volvió necesario el contar con una herramienta que permita realizar pruebas de funcionamiento a la API en desarrollo, es por esto que se eligió Postman, aplicación que facilita un entorno de pruebas para servicios REST y cuenta con múltiples características que facilitan el desarrollo con esta tecnología [METD07].

Por otro lado, el framework Laravel viene integrado con algunos complementos que agilizan el desarrollo de aplicaciones web, a continuación se listan aquellos que son empleados en el proyecto:

- Artisan: Interfaz de línea de comandos incluída con el framework Laravel y que permite la integración rápida con dicho marco de trabajo [HERR1]. Proporciona diversos comandos que agilizan la creación de elementos, migración de contenidos e instalación.
- Eloquent ORM: Implementación del patrón de desarrollo ActiveRecord que permite que el trabajo con la base de datos sea más eficiente [HERR2].

6.2 Base de datos

En esta sección se detallan los pasos relacionados a la creación de la base de datos para el sistema propuesto. Este proceso se realizó en dos etapas: migración, para la creación de los modelos físicos en el motor, y poblamiento, para llenar información predefinida.

6.2.1 Migración

Con el diagrama de base de datos terminado, se procedió a generar la estructura física en el motor MySql. La herramienta Workbench con la cual se realizó el modelo permite la creación directa en la base de datos; sin embargo, se decidió utilizar el concepto de migración del framework Laravel dadas las ventajas que ofrece.

Una migración se asemeja a un versionamiento de la base de datos que permite la fácil modificación del modelo guardado [HERR1]. Entre sus ventajas destaca que la definición de las tablas se realiza utilizando instrucciones en PHP que transforman las consultas de creación en comandos en alto nivel, permitiendo que dicha migración sea fácil de realizar, controlar y principalmente que no dependa de un único motor, es decir, se abstrae la lógica de creación y los mismos comandos se pueden utilizar para múltiples motores de base de datos ya que Laravel viene incorporado con los mecanismos adecuados para comunicarse con Mysql, Postgres, SQLite y SQL Server [METD04].

En la figura 6.1 se muestra un ejemplo de un archivo de migración, en particular el de la tabla de usuarios:

```php
<?php

use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateUserTable extends Migration
{

    public function up()
    {
        Schema::create('user', function (Blueprint $table) {
            $table->increments('id_user');
            $table->string('name');
            $table->string('username')->unique();
            $table->string('password', 60);
            $table->rememberToken();
            $table->string('gmail_user', 45)->unique();
            $table->string('trello_user', 45)->unique();
            $table->string('telegram_user', 45)->unique();
            $table->string('phone', 45);
            $table->integer('id_user_role');
            $table->integer('id_user_status');
            $table->timestamps();
        });
    }

    public function down()
    {
        Schema::drop('users');
    }
}

```

Figura 6.1: Archivo de migración de la tabla de usuarios. Elaboración propia.

Las clases que contienen las migraciones poseen dos métodos: “up” que se ejecuta cuando la migración se instala y que contiene la definición de la tabla y sus campos, y “down” que se ejecuta al hacer rollback a la migración actual, permitiendo que se deshagan las últimas modificaciones. En el caso particular de la tabla usuarios el método “up” crea la tabla y el método “down” la destruye.

6.2.2 Poblamiento de datos

Existen algunas tablas que requieren datos predefinidos y que son necesarias para el correcto funcionamiento y definición de otras entidades, como es el caso de los estados disponibles para los usuarios, proyectos y cotizaciones, los tipos de archivos, entre otros. Estos datos no cuentan con un administrador que permita darles mantenimiento, pero deben encontrarse almacenados desde un inicio en la base de datos para que el sistema funcione correctamente y según lo definido en los requerimientos de la solución.

Para el llenado de esta información se utilizó otro concepto del framework Laravel llamado “semillas” que trabaja de manera conjunta con las migraciones. Una semilla es una clase que permite ingresar información en la base de datos y al igual que una migración no depende del motor de almacenamiento, se define utilizando una clase en PHP y se ejecuta a través del CLI Artisan [HERR2].

En la Figura 6.2 se muestra un ejemplo de una clase de tipo semilla que se utilizará para poblar la información de la tabla de estados de usuario.

```php
<?php

use Illuminate\Database\Seeder;
use Illuminate\Database\Eloquent\Model;
use App\UserStatus;

class UserStatusSeeder extends Seeder {

    public function run()
    {
        $this->command->info('Poblando tabla de estado de usuario.');
        DB::table('user_status')->delete();
        UserStatus::create(["name" => "Activo", "description" => "Usuario activo", "color" => "primary"]);
        UserStatus::create(["name" => "Inactivo", "description" => "Usuario inactivo", "color" => "warning"]);
        UserStatus::create(["name" => "Eliminado", "description" => "Usuario elminado", "color" => "danger"]);
        $this->command->info('Tabla de estados de usuario poblada.');
    }

}

```

Figura 6.2: Archivo de semilla para la tabla de estados de usuario. Elaboración propia.

Una clase de tipo semilla cuenta con un método llamado “run” en el cual se definen las instrucciones de llenado de las tablas, creando a través de los modelos ya definidos los registros en la base de datos. También se pueden realizar otras acciones como el borrado de toda la información de una tabla o registro de mensajes informativos en el log de acciones del framework.

Finalmente, las herramientas de migración y semillas permiten que la definición de las tablas así como los datos iniciales que requieren ser registrados se realicen de manera eficiente y automática [METD04]. Además, facilita el proceso de despliegue de la aplicación en un nuevo entorno ya que todas las operaciones de ejecución de estas clases se pueden realizar con instrucciones en la línea de comandos, a través del cliente Artisan [HERR1].

6.3 Conceptos del framework

El marco de trabajo Laravel cuenta con diversos elementos dentro de su arquitectura que facilitan la organización del código y la implementación de aplicaciones web. A continuación se listan los principales componentes de su arquitectura [METD04]:

- MVC (Model-View-Controller): El patrón de diseño empleado por Laravel es el de Modelo, vista y controlador, el cual permite la separación de la lógica y las responsabilidades de los componentes dentro de la aplicación.

- IoC (Inversion of control): Concepto que permite el manejo de dependencias dentro de clases diferentes. También conocido como inyección de dependencias, facilita el trabajo con clases que deben instanciarse dentro de otras ya que la inyección de dichas dependencias se realiza en tiempo de ejecución en lugar de ser escritas dentro del código. En la Figura 6.1 se muestra un ejemplo del modo en el que se realiza la inclusión de una dependencia en la clase Servicio.

```php
<?php

class Servicio {

    protected $dependencia;

    //La dependencia se coloca como un parámatro del constructor de la clase
    //y el framework se encarga de interpretarla y asignar la instancia correcta
    public function __construct(InterfazDeLaDependencia $dependencia)
    {
        $this->dependencia = $dependencia;
    }

}
```

Figura 6.1: Ejemplo de inyección de dependencias. Elaboración propia.

- Templates (Plantillas): Contienen el código HTML que la aplicación web retorna al navegador del usuario final y que separan la lógica de presentación de la lógica del negocio. El motor implementado por Laravel es Blade, el cual permite la especificación de vistas utilizando una sintaxis fácil de usar y que trabaja de manera conjunta con el código PHP tradicional. En la Figura 6.2 se muestra un ejemplo de una plantilla escrita con la sintaxis del motor Blade, se describen ejemplos de las funcionalidades disponibles tales como inclusión de otras plantillas, iteración de contenidos y ejecución de código en PHP.

```
<div>
    @include('incluir_el_contenido_de_otra_vista')
    <a href="{{ url('/cotizaciones/nueva') }}">Creación de URL</a>
    <h1>Iteración de contenidos</h1>
    @foreach($coleccionDeItems as $item)
    <p>{{ $item }}</p>
    @endforeach
    <h2>Funciones PHP: {{ str_replace('.', ',', $texto) }}</h2>
</div>
```

Figura 6.2: Ejemplo de vista elaborada con Blade. Elaboración propia.

- Routing (Ruteo): Mecanismo que permite la fácil definición una URL soportada por la aplicación web. Laravel implementa esta funcionalidad de manera que en una sola definición se puede incluir la estructura de la URL, el verbo HTTP que soporta y otros patrones de contenido que deben estar presentes. En la Figura 6.5 se muestra la definición de varias rutas utilizando la clase Route proporcionada por el framework.

```
<?php

//Archivo que define las rutas de la aplicación

//Es posible definir las rutas dentro de un grupo de modo que todas inicien desde el mismo punto
//En este caso todas las ruta contenidas en esta definición tendrán como base "/grupo/"
Route::group(['prefix' => 'grupo'], function() {
    //URL: /grupo/operacion1 utilizará el controlador "Operaciones" y el método "método1"
    //cuando la petición se haga usando el verbo GET
    Route::get('/operacion1', 'Operaciones@metodo1');
    //URL: /grupo/operacion2 utilizará el controlador "Operaciones" y el método "método2"
    //cuando la petición se haga usando el verbo POST
    Route::post('/operacion2', 'Operaciones@metodo2');
});
//URL: /grupo/operacion2/{numeroDeOperacion} utilizará el controlador "Operaciones" y el método "método3"
//cuando la petición se haga usando el verbo PUT
//Además, la URL podrá contener un parámetro variable llamado "numeroDeOperacion"
//Ejemplo: /grupo/operacion2/123
Route::put('/operacion3/{numeroDeOperacion}', 'Operaciones@metodo3');
```

Figura 6.3: Ejemplo de definición de URL. Elaboración propia.

6.4 Construcción de los proyectos base

Como se mencionó en el capítulo de análisis, el presente proyecto cuenta con dos componentes: uno que actúa como middleware y expone un listado de servicios y otro que consume dichas funcionalidades y provee la interfaz para el usuario final. Ambos componentes serán implementados de manera separada, pero utilizarán el mismo framework, herramientas y consideraciones de desarrollo. En esta sección se detallan los procedimientos comunes realizados para ambas partes mientras que las características particulares serán definidas y explicadas en secciones posteriores.

6.4.1 Creación de controladores

Los controladores que actuarán cuando se acceda a una determinada URL se encuentran divididos según los paquetes del sistema definidos en el capítulo de análisis. La convención de nombres usada será “[verboHttp][NombreDelMetodo]”, por ejemplo: getListadoNombres. Esta nomenclatura permitirá identificar rápidamente el tipo de petición HTTP al que corresponde el método, además de marcar un estándar entre las clases de este tipo, ya que según el alcance de la solución se tendrán múltiples controladores con una gran variedad de métodos dentro de ellos.

6.4.2 Creación de rutas

Al igual que con los controladores, las rutas URL de la aplicación se encontrarán agrupadas por módulos y además seguirán un mismo patrón de nombres, por ejemplo: para el listado de entidades el nombre de la ruta será “listar[NombreDeLaEntidad]” y para la inserción de una nueva entidad se tendrá la URL "nuevo[NombreDeLaEntidad]”. Por otro lado, en caso se requiera solicitar información de una entidad en particular se usará el formato “detalle/[idDeLaEntidad]”, es decir, como parte de la URL se tendrá el identificador único de la entidad requerida.

6.5 Construcción del componente de servicios

La construcción del componente de servicios se realizó siguiendo los pasos que se mencionan a continuación:

6.5.1 Definición de controladores

Dado que este componente trabajará con múltiples entidades, tanto del sistema como de los servicios web de terceros, se agruparon los controladores de modo que se encapsule la lógica de elementos similares. La lista de controlares implementados es la siguiente:

- ProjectController: encargado de recibir las peticiones relacionadas a la administración y consulta de proyectos, entregables y las actividades asociadas.

- QuotationController: encargado de recibir las peticiones relacionadas a la administración y consulta de cotizaciones, así como de los archivos y comentarios asociados a estas.
- UserController: encargado de administrar las peticiones relacionadas a la administración y consulta de usuarios, grupos de usuarios y sus actividades asociadas.
- OAuthController: encargado de administrar las peticiones relacionadas a la autenticación y verificación de la identidad de los usuarios a fin de establecer una conexión con los servicios web de terceros.
- TelegramController: encargado de administrar las peticiones provenientes de la aplicación de Telegram resultantes de la interacción de un usuario con el Bot implementado.

6.5.2 Definición de rutas

Al igual que con los controladores, las rutas han sido agrupadas según las entidades a las cuales hacen referencia, además, para facilitar la escalabilidad de la solución, todas las rutas comprendidas en este componente cuentan con el prefijo “v1”, el cual indica que se trata de la versión 1 y que permite aislar las funcionalidades de esta primera iteración. En la Figura 6.4 se muestra parte de la definición de rutas de la aplicación, indicando el prefijo “v1”. En posteriores fases del proyecto se agruparían las rutas bajo un nuevo prefijo, evitando que los clientes que consuman la versión anterior se vean afectados. La lista completa de las rutas que serán soportadas se encuentra en el capítulo 4 (Servicios Web).

```
29 Route::group(['prefix' => 'v1'], function() {
30     Route::group(['prefix' => 'cotizaciones'], function() {
31         Route::get('/estados', 'QuotationStatusController@index');
32         Route::get('/estados/{statusId}', 'QuotationStatusController@show');
33
34         Route::get('/', 'QuotationController@index');
35         Route::post('/', 'QuotationController@store');
36         Route::get('/{quotationId}', 'QuotationController@show');
37         Route::patch('/{quotationId}', 'QuotationController@update');
38         Route::delete('/{quotationId}', 'QuotationController@destroy');
```

Figura 6.4: Definición de rutas del componente de servicios. Elaboración propia.

6.5.3 Definición de servicios

Dado que toda la funcionalidad de comunicación entre los diferentes servicios web se encontrará centralizada en este componente y se requerirá implementar una gran cantidad de métodos extensos, se dividieron las responsabilidades necesarias usando los lineamientos del patrón IoC. Los grupos de funcionalidades identificadas fueron implementadas en diferentes clases, cada una de ellas actuando como un “Servicio” según la nomenclatura del framework

Laravel [METD04]. Los controladores tendrán como dependencia a estas clases que serán inyectadas en tiempo de ejecución. La lista de los Servicios implementados es la siguiente:

- DriveService: encargado de las operaciones de interacción con Google Drive para la creación, modificación y consulta de archivos.
- TrelloService: encargado de las operaciones de interacción con Trello para la creación, modificación y consulta de tarjetas y actividades.
- CalendarService: encargado de las operaciones de interacción con Google Calendar para la creación, modificación y consulta de eventos.
- TelegramService: encargado de las operaciones de interacción con Telegram para la consulta y envío de actividades a los usuarios a través del Bot.
- ProjectService: encargado de las operaciones de interacción con las entidades relacionadas a los proyectos.
- UserService: encargado de las operaciones de interacción con las entidades relacionadas a los usuarios.
- QuotationService: encargado de las operaciones de interacción con las entidades relacionadas a las cotizaciones.

En la Figura 6.5 se muestra la relación de dependencia que existe entre los Servicios listados anteriormente y los controladores definidos, la flecha señala la dependencia requerida.

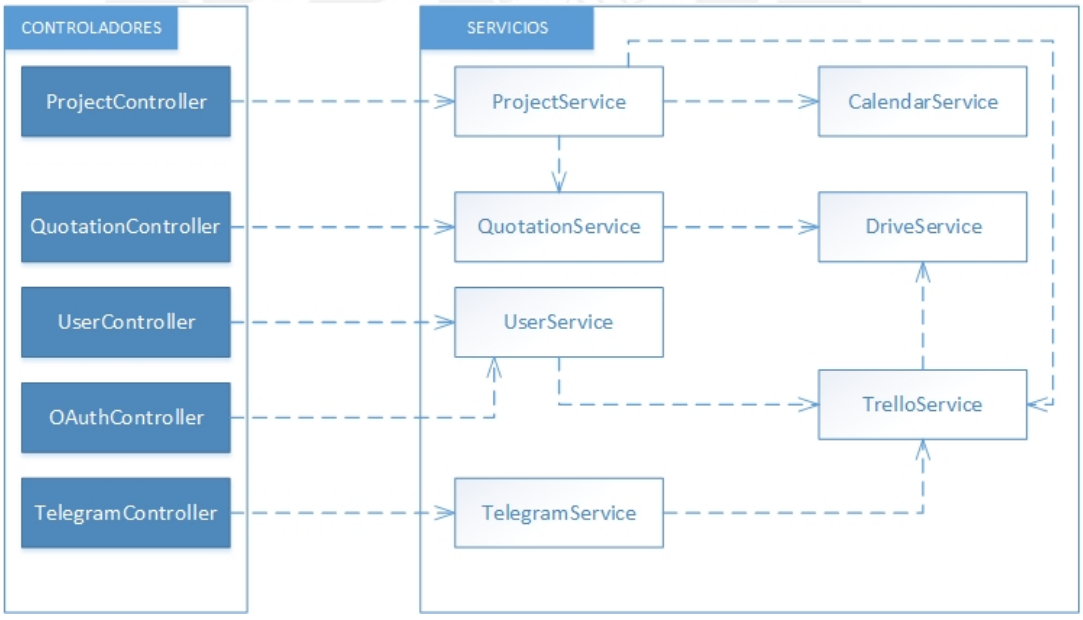

Figura 6.5: Diagrama de dependencias. Elaboración propia.

6.5.4 Implementación

Luego de la definición de las estructuras y componentes se procedió a realizar la codificación del producto. Los servicios que interactúan con la información propia del sistema utilizando

el ORM Eloquent para administrar los datos almacenados en la base de datos. En la Figura 6.6 se muestra el método “addComment” del Servicio “QuotationService”, encargado de insertar un nuevo comentario y asignarlo a una cotización. La creación del comentario se realizar mediante el modelo “QuotationComment”, el cual recibe la información relevante, crea la entidad y devuelve el nuevo comentario insertado.

```php
    public function addComment($userId, $quotationId, $description)
    {
        $comment = QuotationComment::create([
            'description' => $description,
            'id_quotation' => $quotationId,
            'id_user' => $userId
        ]);
        return QuotationComment::with(['user'])->find($comment->id_quotation_comment);
    }
```

Figura 6.6: Trabajo con Eloquent. Elaboración propia.

Del mismo modo, la consulta de la información almacenada se realizó de manera transparente, sin recurrir a la escritura de sentencias en SQL. En la Figura 6.7 se muestra un ejemplo de búsqueda de usuarios, el cual soporta filtros según diferentes criterios tales como nombres de usuario o roles, además de permitir la consulta a relaciones de otras tablas.

```php
public function searchUser($filters)
{
    $users = User::where('id_user', '>', '0');
    if ($filters->input('name') != '') {
        $users->where('name', 'like', '%' . strtolower($filters->input('name')) . '%');
    }
    if ($filters->input('username') != '') {
        $users->where('username', strtolower($filters->input('username')) . '%');
        $users->orWhere('gmail_user', 'like', '%' . strtolower($filters->input('username')) . '%');
        $users->orWhere('trello_user', 'like', '%' . strtolower($filters->input('username')) . '%');
        $users->orWhere('telegram_user', 'like', '%' . strtolower($filters->input('username')) . '%');
    }
    if ($filters->input('userRole') != '') {
        $users->orWhere('id_user_role', $filters->input('userRole'));
    }
    return $users->with(['role', 'status'])->get();
}
```

Figura 6.7: Consultas con Eloquent. Elaboración propia.

El modelo de trabajo descrito anteriormente es usado para todas las operaciones necesarias de los Servicios que trabajan con la información propia del sistema.

Por otro lado, para el trabajo con los servicios web de terceros se utilizó un complemento del framework que permite la llamada a servicios REST. Este mecanismo facilita la comunicación a través del protocolo HTTP al ofrecer un método estático que recibe el verbo HTTP que

identifica la operación, la URL de destino y los parámetros asociados a la petición. En la Figura 6.8 se muestra un ejemplo de llamada a la API de Trello en la que se utiliza el método "addBoard" definido en el Servicio "TrelloService" y que permite agregar un tablero al administrador de Trello. El verbo usado es POST y junto a la llamada se envía el token de acceso y los datos del tablero a crear. La respuesta de la petición es un JSON que es procesado y devuelto como valor para que sea reutilizado posteriormente.

```php
public function addBoard($idOrganization, $name, $description)
{
    $response = Rest::post($this->apiUrl . 'boards', [
        'body' => [
            'key' => $this->key,
            'token' => $this->token,
            'idOrganization' => $idOrganization,
            'name' => $name,
            'description' => $description
        ]
    ]);
    $board = json_decode($response->getBody());
    return $board;
}
```

Figura 6.8: Invocación a servicios REST. Elaboración propia.

Por otro lado, la respuesta que produce este componente es un objeto JSON acorde a las peticiones que se realicen, la definición de los parámetros de entrada así como de las respuestas correspondientes se encuentra en el documento de anexos. En la Figura 6.9 se muestra el resultado de consultar los estados disponibles para un usuario. Para probar las peticiones se utilizó la herramienta Postman.

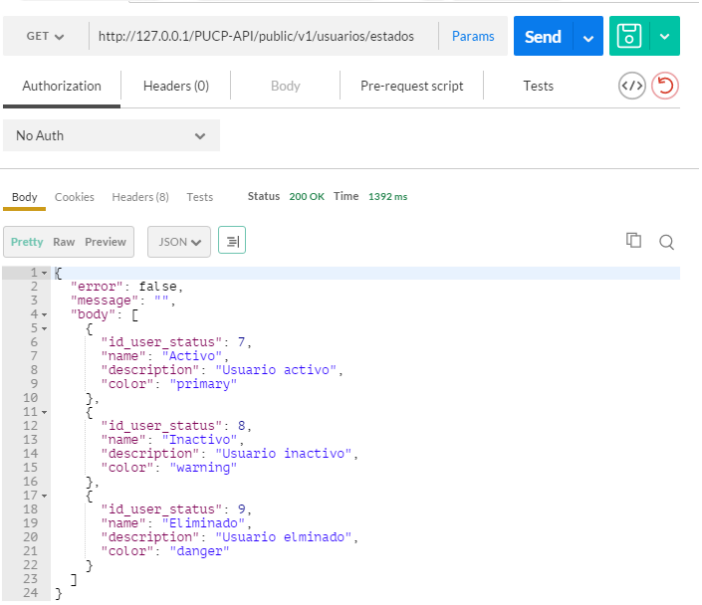

Figura 6.9: Prueba del componente de servicios. Elaboración propia.

Finalmente, con la definición de la forma de trabajo descrita anteriormente y la especificación de la API del capítulo 4 se concluyó la implementación del componente de servicios encargado de exponer servicios REST que podrá ser consumida por diferentes aplicaciones a fin de generar integraciones de diversos tipos.

6.6 Construcción del sistema

Luego de implementar el componente de servicios, se procedió con la construcción del sistema de información que consumirá dichas funcionalidades y permitirá al usuario final el interactuar con los datos almacenados, cubriendo las necesidades definidas en el capítulo 3 de análisis.

El procedimiento de construcción de este componente sigue los mismos primeros pasos que la construcción del servicio REST, iniciando con la definición de controladores y las rutas URL desde las cuales se accederán a las funcionalidades. Nuevamente, las direcciones URL se encuentran agrupadas según las entidades a las cuales hacen referencia usando un prefijo; sin embargo, dado que este componente muestra el administrador de trabajo se requiere el acceso únicamente a usuarios registrados, por lo que es necesario proteger las rutas a usuarios no autenticados. Para este fin se utiliza un “middleware”, concepto del framework Laravel que consiste en funciones que se ejecutan antes de que se procese la petición a una URL [METD04]. En la Figura 6.10 se muestra la definición de rutas del sistema, incluyendo la indicación de usar el middleware llamado “auth”, el cual se encarga de validar que la petición proviene de un navegador con un usuario autenticado. En caso no se cuente con un usuario activo, el framework muestra un mensaje de error indicando que la ruta a la cual se desea acceder se encuentra protegida.

```php
<?php
Route::group(['prefix' => 'usuarios', 'middleware' => 'auth'], function() {
    Route::get('/', function() {
        return redirect('/usuarios/buscar');
    });
    Route::get('/grupos', 'UserGroupController@listGroup');
    Route::get('/grupos/nuevo', 'UserGroupController@addGroup');
    Route::post('/grupos/nuevo', 'UserGroupController@processAddGroup');
    Route::get('/grupos/detalle/{id}', 'UserGroupController@getGroup');
    Route::get('/nuevo', 'UserController@addUser');
    Route::post('/nuevo', 'UserController@processAddUser');
    Route::post('/editarUsuario', 'UserController@editUser');
    Route::get('/buscar', 'UserController@searchUser');
    Route::post('/buscar', 'UserController@processSearchUser');
    Route::get('/detalle/{id}', 'UserController@getUser');

});
```

Figura 6.10: Definición de rutas del sistema. Elaboración propia.

Para el consumo de la API del proyecto REST se utiliza el complemento de Laravel que permite la invocación de este tipo de servicios. La URL de la API es un parámetro configurable por lo que en las llamadas realizadas se utiliza la función “config” para extraer dicha opción. En la Figura 6.11 se muestra el método “addUser” del controlador “UserController” encargado de mostrar el formulario a través del cual se puede agregar un nuevo usuario. Para dicho fin primero se realiza una llamada a la API, con el fin de obtener los roles disponibles y que podrán ser asignados al nuevo usuario.

```php
public function addUser()
{
    $response = Rest::get(config('app.APIEndpoint') . 'usuarios/roles');
    $roles = json_decode($response->getBody())->body;
    return view('users.userAdd', [
        'menu' => 'usuarios',
        'submenu' => 'usuarios',
        'roles' => $roles
    ]);
}
```

Figura 6.11: Consumo de la API de servicios. Elaboración propia.

Por otro lado, la interfaz mostrada a los usuarios se encuentra escrita en HTML, utilizando el motor de vistas Blade. Entre las características usadas se encuentra el concepto de “layout” el permite definir una estructura básica del contenido que puede ser reutilizada por múltiples vistas [METD04]. En la Figura 6.12 se muestra la implementación del layout básico del sistema en la cual se define la estructura que las demás vistas utilizarán, incluyendo las definiciones de la estructura HTML, inclusión de librerías Javascript y archivos CSS. Los bloques “@section” denotan las partes dinámicas que podrán ser reemplazadas por contenido de otras vistas que extiendan el layout, mientras que los bloques “@include” sirven para incluir la totalidad del contenido de una vista diferente.

```html
<!DOCTYPE html>
<html lang="es">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>TESIS</title>
        <link rel="stylesheet" href="{{ asset('css/bootstrap.min.css') }}">
        <link rel="stylesheet" href="{{ asset('css/style.css') }}">
        </head>
        <body class="boxed-layout fixed-sidebar">
            <div id="wrapper">
                @include('.common.menu')
                @section('contenido')
                @show
            </div>
            <script src="{{ asset('js/jquery-2.1.1.js') }}"></script>
            <script src="{{ asset('js/bootstrap.min.js') }}"></script>
            @section('javascript')
            @show
        </body>
</html>
```

Figura 6.12: Implementación de un Layout en Blade. Elaboración propia.

Las vistas de la aplicación cuentan con la estructura mostrada en la Figura 6.13, en la cual se indica que se extiende el layout básico y además se indican los contenidos a reemplazar en las secciones “contenido” y “javascript”.

```
@extends('layout')

@section('contenido')
<div>
    <!-- HTML específico para esta vista -->
</div>
@stop

@section('javascript')
<script>
    // Javascript específico para esta vista
</script>
@stop
```

Figura 6.13: Implementación de una Vista en Blade. Elaboración propia.

El modo de trabajo descrito anteriormente permitió la implementación de las diferentes vistas que conforman la aplicación, favoreciendo la reutilización y organización de contenidos.

Capítulo 7: Conclusiones, recomendaciones y observaciones

En este último capítulo se presentan las conclusiones obtenidas luego del desarrollo del proyecto de tesis, las recomendaciones para futuros trabajos con una temática similar y algunas observaciones respecto al desarrollo.

7.1 Conclusiones

A continuación se listan las conclusiones que se pudieron obtener como resultado de la elaboración de este trabajo, con énfasis en los objetivos que se lograron alcanzar.

- Se realizó la especificación de los procesos de la organización que son soportados por el producto desarrollado, organizando los requerimientos necesarios de modo que cada funcionalidad específica pueda ser delegada a un servicio web de un tercero especializado en dichas tareas.
- Se definió la lista de servicios web necesarios para el cumplimiento de los requisitos contemplados en el análisis de los procesos de la organización. Dicha lista se elaboró tomando en cuenta las herramientas usadas actualmente por la empresa, así como otras que permitan satisfacer las necesidades planteadas. Además, se diseñó una arquitectura orientada a servicios que integró todos los componentes de los diferentes proveedores y que además favoreció la escalabilidad y flexibilidad de la solución.
- Se realizó la especificación de servicios que la solución propuesta expone para que otras aplicaciones se integren con ella y realicen las operaciones de administración contempladas en el alcance.
- Se implementó un componente de servicios REST que actúe como middleware entre los múltiples servicios web elegidos y que además brinde soporte a la lógica del negocio específica de la empresa.
- Se implementó una plataforma web que consuma la API de servicios expuesta por el componente REST del proyecto, proporcionando al usuario final una interfaz capaz de gestionar sus operaciones y ofrecer un entorno visual de administración.

7.2 Recomendaciones

En esta sección se describen algunas recomendaciones a tomar en cuenta para la realización de trabajos con una temática similar a la del presente proyecto.

- Dado que el presente proyecto contempló un componente encargado de exponer una API de servicios REST, aislando las múltiples funcionalidades ofrecidas por diversos proveedores, resulta viable el desarrollo de otras aplicaciones que consuman dicha API. La solución propuesta contempló la implementación de un cliente web de

administración; sin embargo, dada la arquitectura planteada se podrán construir clientes en otras plataformas, tales como aplicaciones móviles nativas o clientes de escritorio para diversos sistemas operativos.

- Si bien la solución propuesta trabajó con proveedores específicos de servicios web determinados por las necesidades actuales de la organización, es posible ampliar el producto de manera que ofrezca al usuario múltiples opciones de servicios web con los cuales integrarse. Por ejemplo, la alternativa de almacenamiento usada es Google Drive; sin embargo, en una futura etapa podría implementarse el soporte para otras opciones, como Dropbox o Box, dando la opción al cliente de que especifique con cual desea trabajar y así ajustarse mejor a sus necesidades.
- El proyecto actual abarcó sólo un grupo de los procesos de la organización, dejando abierta la opción de dar soporte a otras áreas y a sus respectivos procedimientos. Dada la arquitectura planteada, así como los estándares de desarrollo usados, podrían agregarse nuevas integraciones con otros proveedores de servicios web de modo que el producto construido en esta primera etapa sirva de base para crear un ecosistema más completo de administración para la empresa.

7.3 Observaciones

Este proyecto se realizó bajo ciertas condiciones que limitaron el alcance y las tecnologías usadas. A continuación se describen las principales observaciones recopiladas.

- Los criterios de selección de los servicios web consideraron un catálogo de herramientas enfocadas a resolver tareas particulares; sin embargo, con un tiempo mayor se hubiera podido considerar el utilizar proveedores cuyos productos abarcaran más necesidades y tuvieran un mayor nivel de complejidad, con el fin de dar soporte a otros procesos de la organización de una manera más completa.
- Para el trabajo con los servicios web seleccionados se utilizaron los planes gratuitos que no limitaron de ningún modo el desarrollo y características del servicio consumido, pero con un alcance mayor se podría analizar el beneficio que traen los planes pagados de los proveedores con el fin de integrar posibles componentes que se encuentran disponibles únicamente con la compra de una licencia.
- Se contempló dentro del alcance que el sistema de información encargado de consumir la API funcione correctamente en todos los navegadores web líderes del mercado y si bien esto implica que no existan errores al utilizarlos desde plataformas móviles, no existe una versión dedicada para dispositivos móviles que ofrezca la mayor comodidad y usabilidad. En una futura iteración del proyecto se podría considerar que desde un primer momento el sistema cuente con soporte para móviles.

Bibliografía

[PWC1]: PricewaterhouseCoopers (PwC)

Consulta: 06 de abril del 2014

http://www.thecompleteuniversityguide.co.uk/careers/employers/pricewaterhousecoopers-(pwc)/

[PWC2]: Survey: use of cloud computing services within businesses 2012-2013

Consulta: 06 de abril del 2014

http://www.statista.com/statistics/258738/use-of-cloud-computing-services-withinbusinesses/

[CISCO]: Use of cloud computing services within businesses

Consulta: 06 de Abril del 2014

http://www.cisco.com/c/dam/en/us/solutions/enterprise-networks/2012_Cisco_Global_Cloud_Networking_Survey_Results.pdf

[METD01]: Business Process Model and Notation

Consulta: 3 de Agosto del 2015

http://www.bpmn.org/

[METD02]: BPMN software for process modeling

Consulta: 3 de Agosto del 2015

http://www.bizagi.com/en/bpm-suite/bpm-products/modeler

[METD03]: ¿Qué es PHP?

Consulta: 3 de Agosto del 2015

http://php.net/manual/es/intro-whatis.php

[METD04]: Laravel

Consulta: 3 de Agosto del 2015

http://laravel.com/

[METD05]: What is Mysql?

Consulta: 3 de Agosto del 2015

https://dev.mysql.com/doc/refman/5.1/en/what-is-mysql.html

[METD06]: MySQL Workbench

Consulta: 3 de Agosto del 2015

https://www.mysql.com/products/workbench/

[METD07]: Postman

Consulta: 3 de Agosto del 2015

https://www.getpostman.com/

[DEF1]: Definition of: Web application

Consulta: 06 de Abril del 2014

http://www.pcmag.com/encyclopedia/term/54272/web-application

[DEF2]: Gartner IT Glossary – Web Service

Consulta: 06 de Abril del 2014

http://www.gartner.com/it-glossary/web-services

[DEF3]: Software as a Service (SaaS)

Consulta: 06 de Abril del 2014

http://searchcloudcomputing.techtarget.com/definition/Software-as-a-Service

[DEF4]: Service-Oriented Architecture

Consulta: 06 de Abril del 2014

http://www.techopedia.com/definition/24591/service-oriented-architecture-soa

[DEF5]: Middleware

Consulta: 06 de Abril del 2014

http://www.webopedia.com/TERM/M/middleware.html

[DEF6]: API – Application program interface

Consulta: 7 de Agosto del 2015

http://www.webopedia.com/TERM/A/API.html

[DEF7]: HTTP – HyperText Transfer Protocol

Consulta: 10 de Agosto del 2015

http://www.webopedia.com/TERM/H/HTTP.html

[DEF8]: What is REST?

Consulta: 06 de Abril del 2014

http://rest.elkstein.org/2008/02/what-is-rest.html

[DEF9]: OAuth

Consulta: 10 de Agosto del 2015

http://oauth.net/

[DEF10]: The OAuth 2.0 Authorization Framework

Consulta: 10 de Agosto del 2015

http://tools.ietf.org/html/rfc6749

[GAPP1]: ¿Qué es Google Apps? – Google Apps

Consulta: 06 de abril del 2014

http://www.arrobasystem.com/que-es-google-apps.php

[GAPP2]: Google Apps for Business

Consulta: 06 de abril del 2014

http://www.google.com/enterprise/ex/6800292/apps/business/products.html

[ODRIVE]: One place for all your work files — introducing OneDrive for Business

Consulta: 06 de abril del 2014

http://blog.onedrive.com/one-place-for-all-your-work-files-introducing-onedrive-for-business/

[ZIMBRA1]: Zimbra Collaboration - Zimbra

Consulta: 08 de abril del 2014

http://files.zimbra.com/website/docs/Zimbra%20Collaboration%20Product%20Overview.pdf

[ZIMBRA2]: Features - Zimbra

Consulta: 08 de abril del 2014

http://www.zimbra.com/products/zimbra-collaboration/features.html

[ZOHO1]: Business Apps - Zoho

Consulta: 08 de abril del 2014

https://www.zoho.com/business-apps.html

[ZOHO2]: Collaboration Apps - Zoho

Consulta: 08 de abril del 2014

https://www.zoho.com/collaboration-apps.html

[ZOHO3]: Productivity Apps - Zoho

Consulta: 08 de abril del 2014

https://www.zoho.com/productivity-apps.html

[DBOX]: What's Dropbox?

Consulta: 08 de abril del 2014

https://www.dropbox.com/tour/1

[ENOTE]: Evernote

Consulta: 12 de abril del 2014

http://searchconsumerization.techtarget.com/definition/Evernote

[TLLO1]: What is Trello? - Trello

Consulta: 09 de abril del 2014

http://help.trello.com/customer/portal/articles/887702-what-is-trello-

[TLLO2]: Trello Help

Consulta: 11 de Agosto de 2015

http://help.trello.com/

[TLLO3]: Trello API Documentation

Consulta: 13 de Agosto de 2015

https://trello.com/docs/

[GAPP3]: Drive REST API

Consulta: 13 de Agosto de 2015

https://developers.google.com/drive/v2/reference/

[GAPP4]: Google Calendar API

Consulta: 13 de Agosto de 2015

https://developers.google.com/google-apps/calendar/v3/reference/

[GAPP5]: Gmail API

Consulta: 13 de Agosto de 2015

https://developers.google.com/gmail/api/v1/reference/

[TGRM1]: Telegram Messenger

Consulta: 14 de Agosto de 2015

https://telegram.org/

[TGRM2]: Telegram Bot Platform

Consulta: 14 de Agosto de 2015

https://telegram.org/blog/bot-revolution

[TGRM3]: Bots: An introduction for developers

Consulta: 14 de Agosto de 2015

https://core.telegram.org/bots

[TGRM4]: Telegram Bot API

Consulta: 14 de Agosto de 2015

https://core.telegram.org/bots/api

[HERR1]: Artisan Console

Consulta: 17 de Agosto de 2015

http://laravel.com/docs/5.1/artisan

[HERR2]: Eloquent: Getting started

Consulta: 17 de Agosto de 2015

http://laravel.com/docs/5.1/eloquent

[CNST1]: What can PHP do?

Consulta: 20 de Agosto de 2015

https://secure.php.net/manual/es/intro-whatcando.php

[PAPER1]: Kaur S., Bhathal G. "Overview of Saas in cloud computing". International Journal for MultiDisciplinary Engineering and Business Management. India, 2014, Volumen 2, Número 1.

[PAPER2]: Braglia, M., Frosolini, M. "An integrated approach to implement Project Management Information Systems within the Extended Enterprise". International Journal of Project Management. Italia, 2014, Volumen 32, Número 1, pp. 18-2