UNIVESIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATERMATICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

UNA PLATAFORMA WEB PARA REDUCIR LA INCERTIDUMBRE ENTRE EL DESARROLLADOR Y EL CLIENTE

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL EN COMPUTACIÓN

JOSÉ MANUEL AGUILERA ILLANES

PROFESOR GUÍA:
SERGIO F. OCHOA DELORENZI

MIEMBROS DE LA COMISIÓN:
AGUSTÍN VILLENA MOYA
LUIS MATEU BRULÉ
JOCELYN SIMMONDS WAGEMANN

SANTIAGO DE CHILE
2018

Resumen Ejecutivo

Synaptic es una empresa de desarrollo de software que se caracteriza por ahondar en el negocio de sus clientes, con el fin de brindar soluciones que aporten valor en poco tiempo, utilizando metodologías ágiles. El mayor y principal cliente de Synaptic es el Coordinador Eléctrico Nacional (CEN), entidad encargada de la coordinación entre los proveedores de energía y el sistema interconectado eléctrico. Por muchos años Synaptic trabajó con su contraparte del CEN bajo un esquema personalizado y de confianza mutua entre las partes, donde Synaptic le vende “horas de desarrollo” al CEN. Los cambios recientes en el funcionamiento del CEN han llevado a dicho organismo a replantear la forma de llevar adelante los proyectos de desarrollo, y particularmente el control y monitoreo de hora asignadas por Synaptic a dicha labor.

Los principales desafíos que se generaron a partir de este cambio en el escenario de trabajo fueron dos: (1) la transparencia del trabajo realizado, en términos de tareas realizadas en el desarrollo de proyectos, y (2) la justificación de horas de trabajo invertidas por la empresa en dichas tareas. Con el fin de mejorar la transparencia, el seguimiento de proyectos y la comunicación con la contraparte, Synaptic implementó una solución que toma la información de control de horas desde Trello, y se la pasa a un servicio pagado (Everhour) para que este último la disponibilice a través de una serie de reportes. Aunque esta solución ha servido temporalmente, su forma de funcionamiento impide que Synaptic maneje la información interna de sus proyectos de manera fácil y apropiadamente. Para dar solución a ese problema, este trabajo de memoria desarrolló una aplicación Web que se integra con Trello, y permite a Synaptic manejar la información de sus proyectos de manera apropiada, sin tener que dar a sus clientes más visibilidad de lo requerido respecto a los desarrollos que están llevando adelante.

Esta aplicación brinda varias funcionalidades para potenciar la transparencia en el proceso de desarrollo, potenciando la visibilidad para el cliente, la visibilidad interna y disponibiliza información importante del ciclo de vida del software desarrollado por Synaptic, a ambas partes según corresponda. Esta iniciativa de Synaptic ha sido muy bien recibida por la contraparte del CEN, siendo reconocida su utilidad para permitir a ambas organizaciones tomar decisiones en conjunto con mayor velocidad.

La solución se encuentra actualmente en producción y operativa, siendo utilizada ya por el equipo en reemplazo de Everhour. Se espera que la aplicación y la información que ésta maneja sean la base para seguir construyendo soluciones que potencien la dinámica de trabajo entre las partes, la confianza, y por lo tanto una mejora continua en el tiempo para el producto y la relación con el cliente.

Agradecimientos

Sin la motivación, cariño y energía de mis fieles amigos Jonathan y Paulo este trabajo no habría sido posible. Agradezco que me hayan acompañado durante todo el proceso universitario hasta el final. Todas las tristezas y alegrías vividas con ellos me salvaron de mis peores momentos y crearon los mejores recuerdos de mi estadía en esta facultad.

No puedo dejar de mencionar a las personas que hicieron de este trabajo posible: Agustín Villena que me guio y enseñó nuevas formas de ver los problemas además de encaminarme en mi vocación a través de la comunidad y el trabajo en equipo, y Sergio Ochoa que me apadrinó en un momento crítico, dando de su tiempo y dedicación para que este trabajo pudiera realizarse.

Finalmente agradezco a mis padres y a mi mejor amigo por el amor, paciencia, consejos y apoyo incondicional que siempre me han dado durante los años, a quienes dedico todo el esfuerzo de este trabajo.

Tabla de Contenido

Índice de tablas

Índice de ilustraciones

1. Introducción

Synaptic es una empresa que brinda soluciones de software utilizando metodologías ágiles y un involucramiento importante en el negocio de los clientes, con el fin de aportar el mayor valor posible en el menor tiempo posible. Actualmente el mayor cliente de Synaptic es el Coordinador Eléctrico Nacional [1] (de ahora en adelante CEN), organismo encargado de la coordinación de la operación del conjunto de instalaciones del Sistema Eléctrico Nacional, para que éstas operen interconectadas entre sí. Generalmente los proyectos en curso con CEN suelen estar en una de dos fases: (1) inicial o (2) de evolución. Durante la fase inicial (que usualmente dura un par de meses) el desarrollo tiene mucha fuerza, y se busca atacar los problemas que son más importantes y aportan mayor valor al cliente. Luego que se considera que el proyecto cumple con lo necesario para entrar en total funcionamiento dentro del negocio, se pasa a una segunda fase donde disminuye la cantidad de personas del equipo asignadas al proyecto, y se aborda el ajuste de funcionalidades y evolución del producto.

Usualmente la primera fase funciona con iteraciones de 2 semanas, con reuniones donde participan miembros del equipo de desarrollo, y gente del negocio involucrado en el proyecto, entre otras personas. El objetivo es discutir y priorizar las tareas que deben ser resueltas y desarrolladas para aportar mayor valor al negocio del cliente. En esta etapa no se consideran las horas de trabajo utilizadas por los miembros del equipo al momento de desarrollar, pues están asignados de manera fija y a tiempo completo a ese proyecto (durante un período de tiempo determinado). El proyecto se paga de manera mensual.

En el caso del CEN, cuando el proyecto pasa a estado de evolución, y por lo tanto el producto ya se encuentra en producción, las reuniones periódicas y las iteraciones dejan de tener protagonismo. En esa etapa el cliente le compra a Synaptic un conjunto de horas mensuales asociadas al proyecto, y las usan para seguir evolucionando y puliendo posibles detalles del producto. Por lo tanto, el seguimiento de horas asociadas al proyecto cobra importancia.

Actualmente el CEN tiene 9 proyectos en evolución, cada uno de los cuales tiene una cantidad de horas asociadas mensualmente. Cuatro de estos proyectos comparten un conjunto de horas mensuales más grande, debido a que fue negociado de esa manera. En ese caso, las horas dedicadas a las tareas evolutivas de los proyectos, deben ser asignadas por parte del jefe de proyecto asociado desde el CEN, priorizándolas según las necesidades contingentes del negocio.

Dado lo anterior, en Synaptic están asociados ciertos miembros del equipo, a los proyectos en evolución, con el objetivo de disminuir la cantidad de cambios de contexto que deben realizar los desarrolladores. Los miembros asociados a los proyectos en

evolución van rotando luego de un tiempo. Para el manejo de tareas y para apoyar la comunicación acerca del avance del proyecto de cara al cliente, se utiliza la herramienta Trello [2], la cual permite potenciar la transparencia y la participación del área del negocio en dichas actividades. Por otra parte, la empresa también utiliza la herramienta Slack [3] y correo electrónico como medios válidos para apoyar la comunicación al interior de los equipos de desarrollo, y con otros miembros de Synaptic.

Para facilitar el uso de los servicios de Trello por parte de usuarios no experimentados (por ejemplo, los representantes del cliente), se utiliza un servicio externo llamado Everhour [4] que permite contabilizar las horas utilizadas en las tarjetas de un tablero. Esta herramienta también disponibiliza una serie de reportes en una plataforma externa, mostrando información de las horas utilizadas.

1.1 Problema abordado

Hoy en día Synaptic presenta variados problemas principalmente asociados a la modalidad de la segunda fase mencionada en la sección anterior. Por ejemplo, existe un descontento general de los miembros del equipo asociado a un proyecto en estado de evolución, debido al cambio de dinámica entre la primera y la segunda fase de los proyectos en desarrollo. A diferencia de la primera fase (la cual tiene muchas reuniones presenciales y periódicas), la segunda fase se pierde un poco ese nexo, y por lo tanto el cliente CEN comienza a solicitar el desarrollo de nuevas funcionalidades (es decir, requerimientos funcionales) mediante múltiples canales de comunicación. En ese momento la contraparte empieza a perder el compromiso de participar en la priorización de las tareas, además de propiciar un desorden al equipo de desarrollo, puesto que este último debe hacerse cargo de pasar las solicitudes a Trello de manera manual. Por lo tanto, si alguien se olvida de hacer esto, los requisitos asociados a esa funcionalidad es muy probable que se pierdan.

Además de lo anterior, y a pesar de que CEN está en completo conocimiento de que cuenta con un equipo limitado de personas para trabajar sus horas de evolución, frecuentemente se pierde la visión global del estado de los proyectos en evolución, y se generan expectativas de trabajo (del lado del cliente) que superan las capacidades del equipo de desarrollo asignado. Es decir, CEN espera que se realicen más tareas de las realmente abordables, lo cual genera malestar en el equipo de desarrollo, y fricción con el cliente. Sumado a lo anterior, es frecuente que se distribuya de manera poco equilibrada las horas vendidas en un grupo de proyectos (recordemos que estos comparten una cantidad de horas mensuales), y se espera que estos evolucionen a ritmos similares, lo cual no es factible.

Si bien Synaptic ofrece y vende un servicio de evolución de software para lo que corresponde a la etapa de evolución, naturalmente se hace cargo de los errores o bugs que puedan surgir. Normalmente el esfuerzo de los miembros asociados a los proyectos

de evolución está enfocado en desarrollar nuevas funcionalidades, o evolucionar partes del software que van cambiando junto con el negocio. Sin embargo, la percepción del cliente es que se trabaja menos en evolución de lo que realmente se hace, y que se trabaja más en resolución de errores. Debido a lo anterior, desde el punto de vista de CEN, Synaptic ofrece un servicio de “mantención” de funcionalidad, a precios muy elevados para el mercado.

Finalmente, a principios de 2017 la empresa CEN comenzó un proceso de fusión, lo que tuvo un gran impacto sobre las personas del área del negocio que se hacían cargo y participaban de distintos proyectos. Esto significó para Synaptic la pérdida de un escenario de confianza entre las personas de ambas partes, el cual había sido construido durante años de trabajo conjunto. Esto representa un acentuador de los problemas anteriores, debido a que las personas que luego de la fusión se harán cargo de los distintos proyectos, difícilmente accederán a involucrarse en el proceso de desarrollo tanto (y desde un inicio) como las personas anteriormente a cargo.

1.2 Objetivos de la memoria

Dada la problemática planteada, este trabajo de memoria buscó construir una plataforma Web que permita a Synaptic transparentar información relevante de los proyectos, de cara a sus clientes. Esta plataforma será además el único mecanismo de control e ingreso de información entre las partes involucradas; es decir Synaptic y el CEN. Aunque el objetivo general está planteado en base a la relación que hoy mantienen estas dos organizaciones, el mismo problema aparece también en la relación de Synaptic con otros clientes. Por lo tanto, el problema abordado es bastante transversal a la empresa desarrolladora.

Los objetivos específicos que se desprenden del objetivo general son los siguientes:

- Crear un espacio compartido (Web) donde los clientes obtengan información relevante de todos los proyectos que Synaptic está desarrollando para ellos.
- Crear servicios que transparenten las horas trabajadas asignadas por los equipos de desarrollo a los respectivos proyectos, estas horas estarán categorizadas de distintas maneras. Estos servicios deben permitir diferenciar la naturaleza de las tareas realizadas y por realizar en los respectivos proyectos.
- Proveer reportes fácilmente comprensibles para los clientes, acerca del estado general de los proyectos que Synaptic tenga ellos.
- Integrar toda esta funcionalidad en una plataforma Web que sea fácilmente utilizable, tanto por los desarrolladores de Synaptic, como por los miembros de la contraparte, respetando los privilegios de acceso a la información que tendrán los diferentes tipos de usuarios.

1.3 Estructura del documento

Este documento de memoria involucra siete capítulos. El capítulo 2 describe las soluciones existentes analizadas para problemas similares, las tecnologías consideradas previo al desarrollo de la solución, la selección de las tecnologías utilizadas junto con su justificación, y la metodología de desarrollo utilizada para este trabajo.

El capítulo 3 explica los perfiles de usuario presentes en el sistema, los principales requisitos y los procesos que apoyan la solución desarrollada. El capítulo 4 presenta la arquitectura del software desarrollado, su diseño y el ambiente de producción donde fue desplegada la aplicación. En el capítulo 5 se explica y se muestran las interfaces de usuario del software, viendo paso a paso las pantallas de la aplicación. En el capítulo 6 se explica la manera en que se validó la solución desarrollada y los resultados obtenidos de esta validación. Finalmente en el capítulo 7 se presentan las conclusiones a las que se llegaron a partir del trabajo realizado y el trabajo futuro para el sistema.

2. Marco Teórico

En este capítulo se analizan las principales soluciones existentes para resolver el problema planteado, las tecnologías disponibles para llevar a cabo un desarrollo web ad hoc a la necesidad, y una descripción general de la metodología de desarrollo utilizada por Synaptic.

2.1 Descripción de soluciones existentes

A continuación se listan distintas herramientas que podrían haber sido usadas como base para la formulación de la solución al problema planteado.

2.1.1 Jira

Jira [5] es un software de pago por suscripción, que busca resolver los problemas asociados al ciclo de vida del software y de los proyectos. Permite el manejo de flujos de trabajo y monitoreo de proyectos a través de las muchas funcionalidades que posee, dentro de las cuales está el seguimiento de tareas, categorización de estas, y extensos reportes acerca del comportamiento de los proyectos, entre muchas otras cosas. Si bien Jira es una solución bastante completa y muy utilizada por muchas empresas del mercado, para muchas empresas (sobre todo con equipos pequeños) resulta demasiado grande. Como plataforma ya tiene mucho tiempo en el mercado, y es un software al cual las organizaciones deben adaptarse, y no al revés.

En varias oportunidades Synaptic consideró la utilización de esta plataforma para sus proyectos, pero lo descartó principalmente por su complejidad, dado que esa es una gran barrera de entrada para la utilización en conjunto con clientes durante las reuniones. Este aspecto es muy importante para la empresa, a la hora de decidir qué metodología de trabajo implementar para mantener la coordinación y transparencia entre las partes.

2.1.2 Everhour

Everhour es un servicio utilizado para hacer seguimiento de las horas utilizadas en las tareas presentes en otros software. Se utiliza en conjunto con herramientas que apoyan el flujo de trabajo en desarrollos de software, tales como Trello, Jira, Basecamp y muchos otros. Este servicio permite utilizar un cronómetro para computar el tiempo efectivamente asignado a las tareas. El usuario es quien activa este cronómetro para comenzar a tomar el tiempo, y también puede detenerlo en cualquier momento que lo desee.

Este software además es capaz de recibir entradas de los usuarios que quieren ingresar manualmente el tiempo que tomó una tarea. A partir de esta información, Everhour disponibiliza muchos reportes acerca de los proyectos, sincronizándose con las distintas plataformas con las cuales es capaz de comunicarse, y pudiendo filtrar por distintos campos para la presentación de la información de los proyectos. La herramienta funciona como una extensión del navegador, y se comunica con el software de flujo de trabajo que se utilice el equipo de desarrollo para realizar el control y seguimiento de los proyectos.

Ninguna de las dos soluciones satisfacía las expectativas y necesidades de Synaptic, por lo tanto se decidió desarrollar una aplicación Web ad hoc para resolver el problema planteado. La siguiente sección describe brevemente las tecnologías que se analizaron para llevar a cabo este desarrollo.

2.2 Tecnologías consideradas para el desarrollo

A continuación se listan las tecnologías que fueron consideradas al momento de desarrollar la solución reportada en este documento.

2.2.1 Ruby on Rails

Ruby on Rails [6] (también llamado RoR) es un framework para aplicaciones web basado en Ruby [7], con una fuerte inclinación al desarrollo amigable, el cual le da un fuerte énfasis a las convenciones por sobre la configuración. Dentro de sus funcionalidades el framework posee herramientas para generar una aplicación web de manera rápida y fácil, además de poder generar controladores, modelos y vistas. Adicionalmente funciona con el sistema de gemas de Ruby (o ruby gems) que son similares a librerías. Éstas son fácilmente importables e integrables a un proyecto, acelerando el desarrollo de funcionalidades, y utilizando soluciones particulares para problemas resueltos por otros.

Rails se destaca por tener una de las comunidades open source más grandes en el mundo web, además de tener una gran cantidad de gemas para múltiples funcionalidades. Debido a lo anterior, se vuelve un buen candidato a usar a la hora de levantar una aplicación web nueva. A pesar de que el framework tenga montones de capacidades para generar código de manera automática y lograr un desarrollo rápido, este código generado suele ser bastante genérico, y en el caso de las vistas la mayoría de las veces suele desecharse casi por completo. Esto se hace con el fin de construir las vistas necesarias para la aplicación web objetivo, y no agregar componentes o código extra que no es estrictamente necesarios. No obstante, esta capacidad de generación de código es extremadamente útil en caso de querer construir una API, dado que presenta la opción de generar un nuevo proyecto de este tipo, sólo utilizando la función *--only-api* al momento de crear un proyecto. Esto permite dejar de lado todos los paquetes para

vistas, y servir contenido estático, Javascript, etc., que para una API serían innecesarios. Por lo anterior, Rails es muy buena opción para levantar un back-end robusto, de manera rápida, que provea una API REST [8] estándar y con buena velocidad.

2.2.2 Django

Django [9] es un framework web basado en Python que compite directamente con Ruby on Rails en su propio nicho. Éste es un framework robusto, que posee muchas funcionalidades para generar código rápidamente. A diferencia de Rails, Django está diseñado para favorecer mucho más la configuración por sobre las convenciones. Cuenta con una gran comunidad de desarrolladores, lo que hace el proceso de resolver dudas y errores muy expedito, en general. Además posee un toolkit muy completo para generar REST APIs de manera robusta, rápida y fácil. Este toolkit, llamado REST framework para Django, es generalmente la alternativa principal elegida por la comunidad para generar APIs bajo las buenas prácticas existentes.

2.2.3 ReactJS

En el mundo del front-end actual, Javascript ha cobrado gran protagonismo gracias a la evolución de las librerías que han desarrollado distintas compañías y comunidades, con el fin de agilizar y abstraer la a veces ardua tarea de desarrollar un front-end complejo. Con la continua mejora de los teléfonos inteligentes y los computadores que maneja la gente hoy en día, en conjunto con el crecimiento y enriquecimiento de los navegadores, se ha acrecentado la demanda de funcionalidades más ricas y complejas. Esta ha aumentado la necesidad de procesamiento del contenido web.

En respuesta a esta necesidad, el desarrollo de aplicaciones Web ha tendido a enfocarse en darle protagonismo y más responsabilidad lógica al front-end, quitándole carga al back-end, y aprovechando el poder de procesamiento que manejan los dispositivos actualmente. ReactJS [10] es una librería que permite abordar este desafío, facilitando el desarrollo de interfaces de usuario interactivas y complejas a través de diversos componentes. Esta librería fue desarrollada por Facebook para satisfacer las necesidades propias de su sitio, teniendo que manejar la comunicación entre múltiples partes de la interfaz de usuario, compartiendo información entre éstas y manteniéndolas sincronizadas.

La librería propone un desarrollo basado en componentes, siendo estas piezas de software conscientes de su propio estado, independientes, y con un muy bajo nivel de acoplamiento entre ellas, lo cual permite su reutilización y relocación de manera fácil.

React utiliza una sintaxis especial para la definición de sus componentes. Esta sintaxis, llamada *JSX,* se asemeja a HTML y permite abstraer la construcción de los componentes, de su sintaxis en Javascript. Esto hace a los componentes entendibles

para los desarrolladores. Actualmente ReactJS es la librería de *Javascript* front-end más utilizada, la cual ha cobrado mucha fuerza debido a la velocidad de desarrollo que permite, a pesar de que resulte en interfaces complejas y con múltiples interacciones.

2.2.3.1 Webpack

Con el gran incremento de las tecnologías de Javascript front-end para generar aplicaciones de una sola página, y la tendencia de llevar cada vez más lógica de las aplicaciones hacia el lado del cliente, nace la necesidad de manejar la gran cantidad de dependencias y archivos Javascript que se generan para esto. A la hora de servir una aplicación de una sola página, es necesario que el punto de entrada de la aplicación sea un sólo archivo Javascript que despliegue la aplicación completa, junto con todo el CSS y los demás assets que sean necesarios para que la carga realizada por el navegador al ingresar a la aplicación sea de un solo archivo por tipo (un archivo Javascript, un archivo CSS, etc.). De esa manera el navegador no estará consultando por recursos una y otra vez.

Para apoyar el desarrollo de este tipo de soluciones nacen tecnologías como Webpack [11], que es un agregador de recursos (o assets*),* capaz de tomar gran cantidad de archivos Javascript, CSS, Coffeescript, etc., resolver sus dependencias, consolidarlos en un sólo archivo para cada extensión y minificarlos (proceso donde se remueven caracteres innecesarios para la ejecución del código). Esto permite optimizar los recursos del navegador y facilitar el despliegue de aplicaciones estáticas de una sola página.

2.2.3.2 Grommet

Grommet [12] es una librería de componentes para ReactJS. Estos componentes pueden ser importados y utilizados rápidamente en una aplicación, con el fin de agilizar el diseño de interfaces de usuario modernas y de calidad estética. Para instalar Grommet sólo es necesario incluirla en el archivo *package.json* que maneja las dependencias de la aplicación, y correr el comando *npm install* bajo la raíz de la aplicación ReactJS.

2.2.4 VueJS

VueJS [13] es un framework de front-end orientado a componentes de interfaz de usuario (UI). Este framework representa una alternativa que ha ganado gran popularidad y respaldo por la comunidad, gracias a su simplicidad de sintaxis y facilidad de aprendizaje, debido a su cercanía a HTML con Javascript estándar. Este framework cuenta con una librería de componentes de VueJS pre-hechos, que tienen un look-and-feel moderno que permite agilizar el comienzo del desarrollo de interfaces web a través de sus múltiples opciones. La librería se llama Vuestic [14] y es muy fácil de instalar, debido a que los desarrolladores incluyeron un instalador para integrar todo lo necesario a un proyecto Vue.

2.2.5 Trello Power-ups

Los Power-up de Trello [15] son módulos independientes que permiten modificar la experiencia de los tableros de Trello, o agregar nuevas funcionalidades a través de la API de estos. Los módulos son básicamente archivos HTML y Javascript simples, como una página web estática, que se inserta en un tablero de Trello a través de iFrames. La API de Power-ups permite elegir distintas *capacidades* (como lo nombran los desarrolladores de Trello) que definen el Power-up. Las *capacidades* son áreas de la experiencia de usuario que los desarrolladores pueden acceder y manipular, añadiendo funcionalidades al tablero.

Existen numerosos Power-up que pueden ser agregados de manera gratuita a través de la plataforma, los cuales son provistos por diferentes desarrolladores o por compañías de software. También es posible utilizar Power-ups privados, alojados en dependencias de terceros, los cuales pueden ser anclados a un equipo de Trello para que éste pueda añadirlo a sus tableros. Esta nueva área de la API de Trello está siendo fuertemente utilizada por muchos usuarios alrededor del mundo, debido a que permite enriquecer la experiencia y la manera de operar de cada organización.

2.2.6 Heroku

Heroku [16] es una plataforma en la nube que permite desplegar aplicaciones web de manera fácil y rápida, utilizando distintos lenguajes o tecnologías de contenedores. Esta plataforma tiene planes gratuitos y pagados, pudiendo adaptar las capacidades de una instancia según las necesidades del desarrollador y de los usuarios.

La plataforma es ampliamente utilizada por desarrolladores debido a que es muy rápida para desplegar aplicaciones de complejidad relativamente baja, debido a que Heroku sabe desplegar aplicaciones web de la mayoría de los framework open source populares por defecto. A esto lo hace simplemente agregando un archivo de texto descriptor, que le dice a Heroku cómo tiene que interpretar y desplegar el código que el desarrollador le provee a la instancia.

Si bien esta plataforma permite desplegar aplicaciones de manera muy simple, a través de un push desde un repositorio Git por ejemplo, tiene ciertas limitaciones debido a que la instancia de la aplicación no es fija. Es decir, Heroku redistribuye y despliega las instancias de aplicaciones de manera dinámica por demanda, y éstas no tienen almacenamiento ni todas las capacidades normales de un servidor web común y corriente.

Como Heroku no provee de almacenamiento, no es posible tener una base de datos directamente en el mismo ambiente que la aplicación. Sin embargo, la plataforma provee de un sistema de Add-ons que permite anclarle, a una instancia, distintos servicios de bases de datos como Postgres o MongoDB en la nube, además de otras capacidades como mailing, monitoreo, analítica, etc.

2.2.7 Now

Similar en características a Heroku, Now [17] es una plataforma que ofrece servicios de hospedaje para aplicaciones con despliegue rápido. Al ser mucho más joven que Heroku, ésta tiene soporte para menos tecnologías. Now soporta aplicaciones basadas en NodeJS, y también ofrece soporte para contenedores Docker [18], lo que abre el abanico de lenguajes a utilizar, siempre y cuando la aplicación esté montada sobre un contenedor.

Now ofrece un sistema de versionamiento de aplicaciones muy útil para dar visibilidad a las aplicaciones de manera rápida, y validar con los usuarios. Al subir una nueva versión de la aplicación a la plataforma, ésta ofrece un link de acceso rápido directo a la versión del software, único para dicha versión. Esto evita la necesidad de tener que eliminar las versiones anteriores, permitiendo comparar directamente las diferencias en tiempo real en un ambiente productivo.

2.2.8 Redis

Redis [19] es una estructura de datos de almacenaje en memoria primaria, ampliamente utilizado como base de datos o agente de mensajes (*broker*). Soporta diversas estructuras de datos como strings, listas, hashes, entre otras. Esta herramienta se usa comúnmente como base para generar sistemas de colas para mensajes, que deben ser procesados en segundo plano por una máquina o una instancia de servidor.

2.2.9 Resque

Resque [20] es una librería para Ruby, basada en Redis, y diseñada para manejar trabajos y tareas en segundo plano a través de colas. Las tareas pueden ser procesadas de manera paralela o con un desfase de tiempo.

Las tareas o trabajos en Resque son clases de Ruby, lo que hace bastante intuitivo para un desarrollador de esa tecnología diseñar y escribir las tareas modulares, que se utilizarán para resolver los problemas que necesita. Para consumir y procesar los trabajos, es necesario que coexista con una instancia de Redis corriendo, que es la que provee la estructura para almacenar y manejar la cola de tareas.

2.2.10 Amazon Web Services (S3 Bucket)

Amazon Web Services o AWS es una plataforma de servicios en la nube popular y robusta, que permite a los desarrolladores y organizaciones abstraerse de las necesidades de hardware, y utilizar los procesos e instancias en la nube que AWS provee para sus necesidades. La plataforma ofrece múltiples servicios, uno de estos es S3 [21], pensado para el almacenamiento de contenido en la nube de manera eficiente. Los contenidos que pueden ser almacenados en S3 son documentos, archivos, entre otras cosas, además de contenido web estático, dado que no necesita de lógica compleja.

AWS es ampliamente utilizado debido a su relativa simplicidad, velocidad y amplia documentación y comunidad detrás que facilita su uso en distintas tecnologías, servicios de integración continua, etc.

2.2.11 SemaphoreCI

Semaphore [22] es un servicio en la nube, destinado a apoyar la Integración Continua de aplicaciones. Este servicio permite probar el código de una aplicación web, en una instancia web virtual, pudiendo correr tests, instalación de dependencias, etc. en anticipación a un paso a producción. Esto se hace con el fin de prevenir errores y despliegues fallidos.

Este servicio separa con éxito los pasos de prueba del código en una instancia, con el paso a producción, permitiendo al usuario elegir si quiere solamente probar sus pasos de construcción y no realizar un despliegue de la aplicación. La plataforma destaca por su velocidad, simplicidad de uso, modalidad gratis y numerosas integraciones con servicios de terceros para agilizar el proceso.

2.2.12 Tecnologías escogidas

En esta sección se discutirá brevemente el por qué se eligieron las tecnologías para el desarrollo de la aplicación reportada en esta memoria.

2.2.12.1 Back-end

Se escogió Ruby on Rails por sobre Django debido a la familiaridad del memorista con la tecnología, además de la gran rapidez para generar el código base necesario para construir una REST API. Por otro lado, el uso de Ruby on Rails en este sistema tiene el propósito de impulsar el aprendizaje de la tecnología en el equipo de Synaptic, debido a que en general, Python es la tecnología preferida y Django el framework preferido.

2.2.12.2 Front-end

En el caso del front-end, ReactJS resultó la tecnología elegida debido a que se perfila como la tecnología líder en su ámbito, además de ser de conocimiento del alumno memorista. Por otro lado, el uso de ReactJS por Facebook (empresa que desarrolló la librería) da más confianza en su uso y mucha más fuerza en la comunidad, debido a que tienen incentivos claros para arreglar errores y empujar el progreso de la tecnología, por lo que se consideró una buena oportunidad para ahondar y practicar en esta tecnología.

2.3 Metodología de desarrollo

El memorista trabaja en Synaptic desde hace un año, y por lo tanto, conoce en detalle la problemática abordada y el ámbito de negocio en el cual ésta se inserta. Dada esa situación, se siguió un proceso de desarrollo flexible al cambio, involucrando a los usuarios activamente. A continuación se detallan las principales actividades realizadas.

La metodología utilizada no es la aplicada por el equipo de desarrollo de Synaptic, pues este trabaja en iteraciones de dos semanas, contra entrega a cliente y reunión de avance. Debido a que este trabajo fue realizado en las oficinas de la empresa, se prefirió una metodología más simple, buscando feedback en la medida de lo posible durante su desarrollo.

2.3.1 Realizar el análisis del problema

En una fase inicial, fue importante que el equipo de Synaptic se reuniera y describiera el problema en su totalidad, con el fin de identificar cuáles son los síntomas y dolores más importantes con respecto al problema. Tener una imagen detallada del problema a resolver permite al equipo plantear un objetivo claro y conciso, evitando abordar requerimientos que no se alinean con el objetivo del proyecto, o no apuntan a la resolución del problema de manera eficiente y eficaz (o directamente no ayudan a resolverlo). Una vez que estuvo claro el problema a resolver, el equipo pudo comenzar a decidir qué funcionalidades y qué capacidades debía tener la plataforma en una primera versión. Es importante destacar que el autor de esta memoria, además de ser desarrollador de esta solución, lideró el proyecto.

2.3.2 Generar la descripción de funcionalidades de la plataforma a construir

Tener una discusión de las funcionalidades deseables una vez que el problema a resolver está claro es de gran utilidad, independiente de que todas estas funcionalidades estén implementadas o no en el producto final desarrollado. Esto permite iniciar una discusión más aterrizada sobre la viabilidad de implementarlas, su real propósito y en qué

grado resuelven o atacan partes del problema planteado. Luego de obtener una buena idea de qué funcionalidades eran deseables, fue posible comenzar a decidir la arquitectura del sistema, y las tecnologías a utilizar para soportar las funcionalidades deseadas. Finalmente, fue necesario tener una priorización inicial de los requerimientos a abordar, para enfocar los esfuerzos y desarrollar en base a la entrega de valor percibido de las funcionalidades escogidas.

2.3.3 Desarrollo de la solución

El desarrollo de la solución se realizó en distintas iteraciones, tomando las funcionalidades que el equipo consideró como claves para la base del software en una primera aproximación, dando tiempo para madurar y obtener feedback de la plataforma en los distintos incrementos. Las iteraciones no fueron planificadas, debido a que éstas se dieron de forma natural en base a la utilización y feedback obtenido de la plataforma.

Gracias a estos tiempos de maduración, observación y feedback, varias veces se replanteó la manera de abordar los requerimientos iniciales durante el tiempo de desarrollo. Esto significó en una mejora constante en la calidad del software construido.

2.3.4 Validación del producto

Luego de tener una primera versión con el conjunto de requerimientos decididos, el equipo pudo hacer una validación más completa de las funcionalidades planteadas y desarrolladas con los clientes, para así evaluar el impacto con respecto a los objetivos propuestos en un inicio. Los resultados de esta validación permitieron decidir cómo seguir construyendo el software en base a esta versión inicial y los cambios a realizar.

3. Concepción de la solución

En este capítulo se discuten los perfiles de usuario presentes en la solución, los principales requisitos y los procesos que apoya el sistema desarrollado.

3.1 Perfiles de usuario

Se consideraron dos perfiles de usuario para la plataforma, el perfil Administrador y el perfil Cliente. La diferenciación de perfiles se realizó con el fin de limitar a los usuarios tipo Cliente la visualización de información de otras organizaciones y que no puedan configurar la aplicación.

3.1.1 Usuario Administrador

El usuario Administrador es capaz de tener control absoluto sobre la aplicación, es decir, configurar organizaciones (clientes), crear, asignar y eliminar proyectos, además de crear usuarios para las distintas organizaciones. Todos los miembros del equipo de Synaptic tienen el mismo poder, no hay una diferenciación especial dado que el equipo es horizontal, por lo que no se consideraron perfiles intermedios. Todos los usuarios del equipo de Synaptic pertenecen a este perfil.

3.1.2 Usuario Cliente

El usuario tipo Cliente es capaz de ver los reportes, gráficos y detalles relacionados a los proyectos de su propia organización asignada. En particular, también puede generar sus propios reportes personalizados. Este tipo de usuario no es capaz de entrar a la vista de administrador.

3.1.3 Tabla Resumen Perfiles de Usuario

A continuación se incluye la Tabla 1 donde se compara el acceso a funcionalidades que tienen los distintos perfiles de usuario del sistema.

Tabla 1: Resumen de permisos de usuario por perfiles

Funcionalidad / Usuario	Administrador	Cliente
Administración y configuración	Acceso total.	Sin acceso.
Reporte de tareas y horas trabajadas por proyecto o grupo de proyectos	Acceso total. Pueden acceder a los datos de cualquier proyecto.	Acceso parcial. Limitado a los proyectos que pertenecen a la organización del usuario.
Gráficos y analítica	Acceso total. Pueden acceder a los datos de cualquier proyecto.	Acceso parcial. Limitado a los proyectos que pertenecen a la organización del usuario.
Resumen	Resumen de las tareas realizadas en el último tiempo.	Resumen de las tareas realizadas en los proyectos de la organización del usuario.
Listado de proyectos	Acceso total. Pueden acceder a los datos de cualquier proyecto.	Acceso parcial. Limitado a los proyectos que pertenecen a la organización del usuario.
Detalle de proyecto	Acceso total. Pueden acceder a los datos de cualquier proyecto.	Acceso parcial. Limitado a los proyectos que pertenecen a la organización del usuario.

3.2 Principales requisitos de la solución

A continuación se listan los principales requisitos de la plataforma, obviando ciertos requisitos relativamente estándar, como por ejemplo autenticación, diferenciación por roles, etc.

3.2.1 Proporción de tareas evolutivas contra tareas error por proyecto

Una funcionalidad importante es desplegar de manera clara y visible la proporción de las tareas evolutivas y las tareas de error en los proyectos, mostrándose en cada uno de estos individualmente y de manera global entre todos los proyectos de un cliente.

Lo anterior permite que el cliente pueda ver cuál es la naturaleza de las tareas resueltas por el equipo de Synaptic para sus proyectos, enfocado en primer lugar en que idealmente la gran mayoría de tareas resueltas sean de tipo evolutiva, ya que un gran porcentaje de tareas de error puede indicar un problema mayor en un proyecto. De esta manera, el cliente tiene mayor control con respecto al estado de sus proyectos en curso, pudiendo tomar decisiones de manera informada y más rápida que enfrentando las tareas y requisitos día a día sin una visión comparativa. Además, permite identificar qué proyectos requieren mayor atención, pues un proyecto con mayor grado de tareas de error puede ser indicador de que ese proyecto necesita especial atención y mayor energía impresa por el equipo.

3.2.2 Power-up de Trello para ingreso de horas

Con el fin de reemplazar las funcionalidades provistas por el servicio de Everhour, es necesario un Power-up de Trello similar al anterior pero propio de Synaptic. Esto con el fin de mantener como base las capacidades que goza el equipo de Synaptic con Everhour, pero con un mayor grado de control y mejor capacidad de personalización. Este plugin tiene las siguientes finalidades:

1. **Darle control a Synaptic sobre la información que Synaptic genera**: Dado que Everhour es un servicio pagado mensualmente que almacena las horas asociadas a las tareas de Trello en dependencias propias, el equipo de desarrollo de Synaptic tiene poco control sobre la información de las horas ingresadas. Everhour no provee de una API por la cual se pueda obtener esta información y solo es posible obtenerla a través de los reportes que genera (PDF, Excel, CSV), lo que tiene como consecuencia que si Synaptic quiere obtener un análisis personalizado o migrar esa información a otro lado, es necesario generar un desarrollo sobre estos reportes.

2. **Darle poder de personalización e independencia al equipo de Synaptic:** Tener una herramienta propia para el ingreso de horas a través de Trello le permite al equipo de Synaptic cambiar la lógica en que manejan el tiempo invertido por tarea de manera interna, a través de un desarrollo interno.

3. **Soporte con Trello de escritorio:** Debido a que Everhour transmite la información de las horas a sus servidores a través de una extensión de Google Chrome, el servicio no funciona con la nueva versión de escritorio de Trello. Por otro lado, al llevar toda la lógica de almacenaje de horas en las tarjetas a un plugin nativo de Trello, funcionará en todas sus versiones, y tendrá mejor probabilidad de funcionar en futuras versiones de Trello por utilizar solamente tecnología provista por su API.

3.2.2.1 Edición de horas invertidas en tarea

Es necesario que el Power-up de Trello tenga una forma de editar las horas ingresadas a una tarea del tablero, indicando la fecha en que se invirtieron esas horas, además de las horas y minutos invertidos en dicha tarea.

3.2.2.2 Añadir horas invertidas a tarea

Además de poder ingresar el total de tiempo utilizado en una tarea, es necesario que exista una manera de agregar tiempo al total de tiempo invertido existente a una tarea. La razón de esto es más que nada por simplicidad y evitar que el usuario se equivoque al sumar las horas. Además, esta funcionalidad está presente en la versión provista por Everhour y se utiliza internamente en Synaptic de manera regular.

3.2.3 Contabilización de horas trabajadas por proyecto

Es necesario que el sistema permita contabilizar las horas registradas por el equipo de desarrollo de Synaptic. Las horas asociadas a las tareas del tablero de Trello deben estar almacenadas en la plataforma en una base de datos de propiedad de Synaptic, con el fin de mantener el control de esta información y poder realizar cualquier tipo de cálculos o reportes con esta información.

Adicionalmente, es importante que el sistema identifique a qué tarea pertenecen las horas invertidas, y a qué proyecto pertenece dicha tarea, además de saber en qué momento se registraron esas horas y el momento en que fueron invertidas. De esta manera el equipo de Synaptic puede obtener las horas invertidas en un proyecto e identificar en qué momento fueron invertidas.

3.2.4 Distribución de horas trabajadas en los proyectos durante el año

Debido a que muchos proyectos del Coordinador Eléctrico Nacional comparten carga horaria de trabajo por temas contractuales es natural que estos no evolucionen todos al mismo tiempo de manera uniforme, ya que los requerimientos del usuario tienen distintas prioridades y la fuerza que se le da al desarrollo varía en el tiempo.

Si bien lo anterior está claro para ambas partes, y la priorización de las horas compartidas es realizada por el área de TI del CEN, muchas veces no hay claridad de la fuerza comparativa que se le ha dado a los proyectos durante el tiempo, por lo que nace la necesidad de tener una manera de comparar la carga horaria entre proyectos durante el tiempo. De esta manera el cliente es capaz de tomar decisiones de manera informada con respecto a cómo utilizar sus recursos basándose en el historial pasado de sus proyectos.

3.2.5 Reporte de tareas y horas invertidas por proyecto (PDF y Excel)

Por razones de control y transparencia, es necesario que Synaptic genere un reporte para el CEN donde informe en qué se utilizó el tiempo del mes en los proyectos. Normalmente el cliente conoce el estado de sus proyectos, la priorización y el tiempo aproximado utilizado en el desarrollo de las funcionalidades, pero el CEN tiene muchos usuarios/clientes además del área de TI que maneja los proyectos, por lo que es muy útil tener esta información de manera resumida y a la mano.

Debido a lo anterior es necesario que se puedan generar reportes de las tareas realizadas y la carga horaria utilizada para cada una de ellas de manera personalizada, es decir, utilizando filtros sobre las fechas deseadas y los proyectos deseados.

3.2.6 Resumen de trabajo realizado

Una funcionalidad muy utilizada por el equipo cuando se utilizaba Everhour era el resumen de tareas realizadas (en la semana actual, semana siguiente, u otras categorizaciones) debido a que permite visualizar de manera clara el avance del trabajo de un proyecto en el corto plazo sin necesidad de mantener ordenado el tablero de Trello.

Por otro lado, muchos miembros del equipo de Synaptic suelen estar trabajando en más de un proyecto de manera simultánea, por lo que es de utilidad poder ver el avance personal consolidado entre los diferentes proyectos, sin necesidad de hacer comparaciones de tableros distintos.

El resumen de trabajo realizado de un proyecto le permite a los clientes ver que realmente se están desarrollando las funcionalidades que se priorizaron con el equipo de Synaptic, además de hacer seguimiento del estado de estas funcionalidades para en algunos casos poder revisar en los servidores Beta de esos proyectos.

3.2.7 Ingreso de nuevos requerimientos a proyecto

Durante mucho tiempo se han utilizado los tableros de Trello en conjunto con personas que participan del proyecto por el lado del cliente con el fin de que sean parte del proceso del ingreso de requerimientos a los tableros, prioricen, etc., pero es común que otros usuarios y jefes de proyecto transmitan sus inquietudes a través de otros medios, como correos electrónicos.

Por otro lado, es frecuente que los usuarios comuniquen errores a través de correos electrónicos, los cuales muchas veces van dirigidos a ciertas personas que están llevando los proyectos afectados. En estos casos muchas veces las personas que reciben el correo no pueden atender los errores de manera inmediata, y tienen que comenzar a transmitir esta necesidad hacia el equipo. Por lo tanto, contar con un lugar centralizado

que permita a los usuarios alertar estos incidentes es bastante útil, automatizando la comunicación de la situación e ingresando una tarjeta en el tablero de Trello.

3.3 Procesos involucrados

A continuación se detallan los principales procesos de negocio que la aplicación apoya.

3.3.1 Ingreso de horas trabajadas en una tarea

Uno de los principales procesos de negocio es ingresar las horas trabajadas en una tarjeta de Trello. Las personas del equipo de Synaptic deben ingresar cuanto le tomó realizar una tarea en el detalle de una tarjeta de Trello con el fin de poder transparentar esa información con el cliente, y poder llevar un seguimiento del rendimiento del equipo con respecto al desarrollo de los proyectos. El proceso descrito puede verse en la Figura 1.

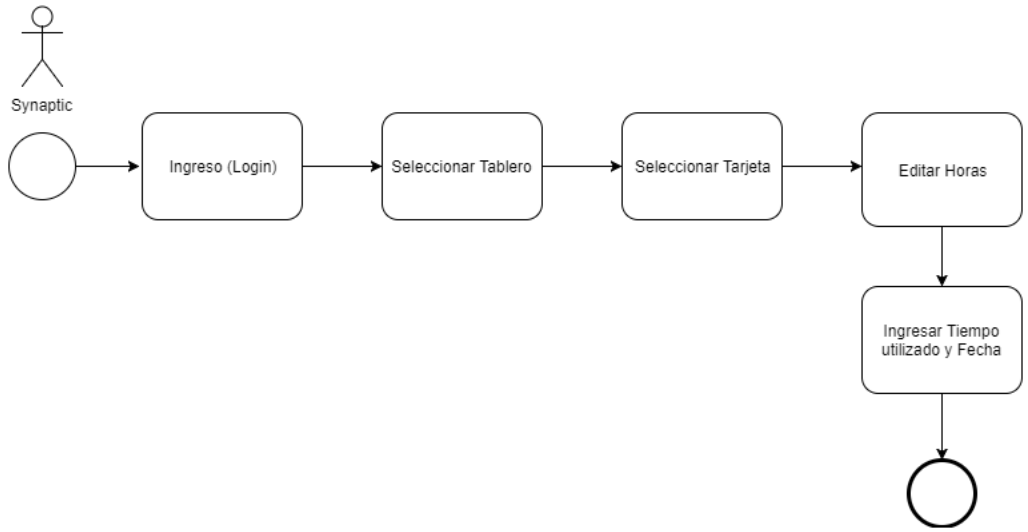

Figura 1: Proceso de ingreso de horas

3.3.2 Conteo de horas trabajadas en el mes por proyecto o grupo de proyectos

Considerando a los miembros del equipo de Synaptic, y miembros clave de la contraparte de los proyectos (como jefes de proyecto), es normal que realicen un conteo de las horas ingresadas en las tareas de un proyecto (o varios de estos en conjunto), con el fin de realizar seguimiento y cuadratura esencial del trabajo concretado. El proceso descrito puede verse en la Figura 2.

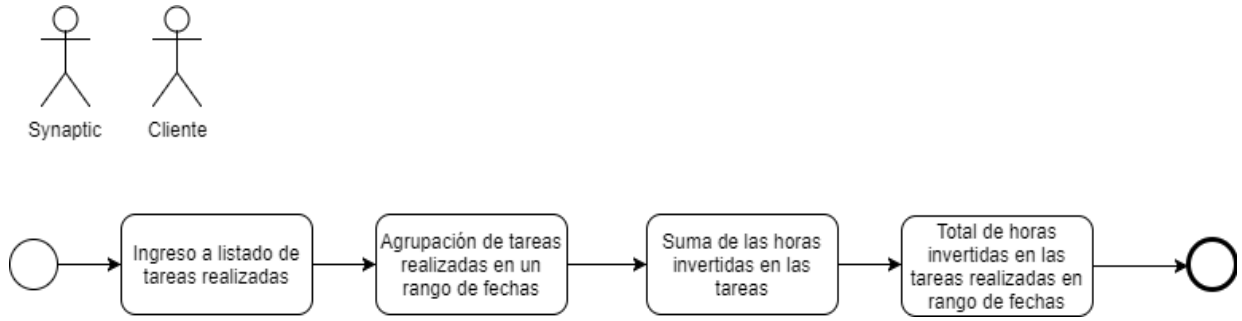

Figura 2: Proceso de conteo de horas trabajadas

3.3.3 Revisión de errores en el mes por proyecto o grupo de proyectos

Parte de lo que se considera al evaluar el estado de los proyectos es una revisión de los errores que emergen durante el transcurso de estos. Para esto, es necesario hacer un recuento de los errores y poder identificar la cantidad y el peso proporcional que tienen dentro del proyecto y dentro de un ciclo temporal de este. El proceso descrito puede verse en la Figura 3.

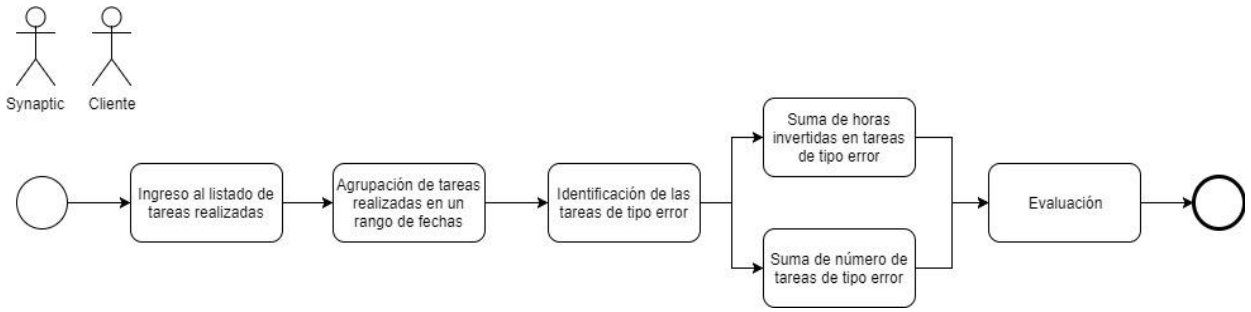

Figura 3: Proceso de revisión de errores

3.3.4 Revisión de tareas realizadas en la semana

Si bien Synaptic suele operar en iteraciones de dos semanas, es normal que cada semana se consolide el trabajo realizado y se haga una pre-validación previa antes de cerrar la iteración. El proceso descrito puede verse en la Figura 4.

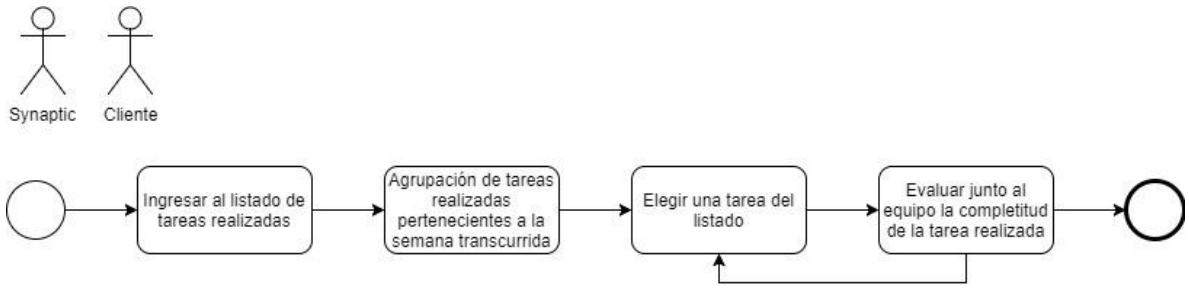

Figura 4: Proceso de revisión de tareas realizadas en la semana

3.4 Modelo de datos

En esta sección se explican los modelos de datos utilizados tanto en el Back-end de la aplicación, como la forma de guardar los datos en el Power-up de Trello.

3.4.1 Modelo de datos en el Back-end RoR

El problema se modeló de manera casi completamente relacional, exceptuando pequeños casos donde se incluyen documentos dentro de otros para facilitar el acceso a información presente en relaciones.

En una primera aproximación de la solución, se utilizó un modelo netamente documental, donde la estructura de organización, proyecto y tarjeta estaba completamente anidada. Esta manera de modelar tenía mucho sentido desde un punto de vista teórico, debido a que las tareas dependen y pertenecen netamente al proyecto que las gatilla, y los proyectos a sus clientes/organizaciones. Si bien esta forma es adecuada para ciertos escenarios, se realizó a mitad de proyecto una refactorización y cambio de modelo completo con el fin de mejorar el acceso a la información y permitir consultas más flexibles a menor costo. El modelo de datos se puede apreciar en la Figura 5.

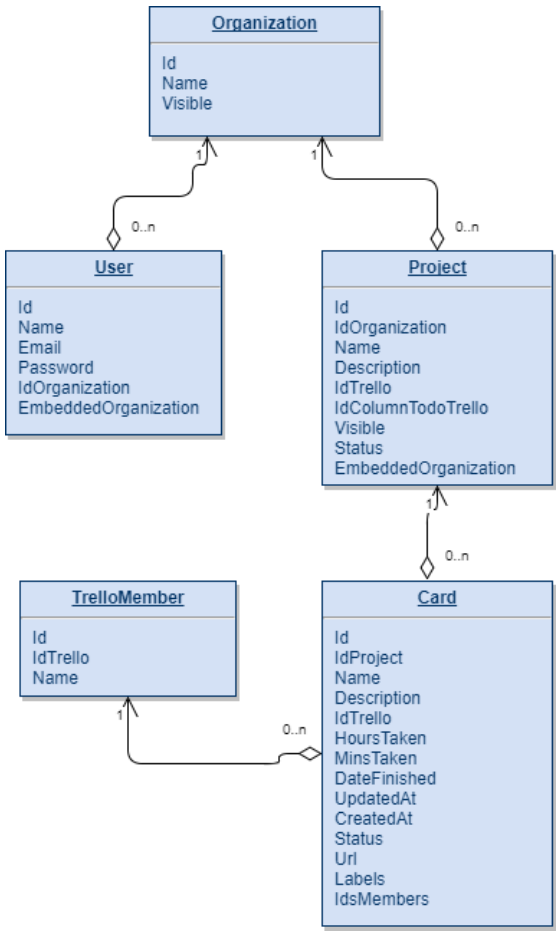

Figura 5: Modelo de datos del Back-end

3.4.2 Modelo de datos en Trello Power-up

Los Power-up de Trello permiten almacenar información en distintos alcances, como tableros o tarjetas, con el fin de mantener estados personalizados dependientes de la funcionalidad que desea implementar el Power-up particular. Esta información solo puede ser almacenada como JSON y es manipulada a través de los eventos generados por el Power-up. Debido a lo anterior, se decidió utilizar un esquema simple y conciso de información con el fin de que este sea fácil de utilizar. Dentro de la tarjeta se almacenan en un objeto Javascript las horas, minutos, la fecha de realización de la tarea, el *timestamp* del momento en que se generó la operación y un listado histórico de los cambios que se han realizado en la tarjeta. La estructura se puede visualizar en la Figura 6.

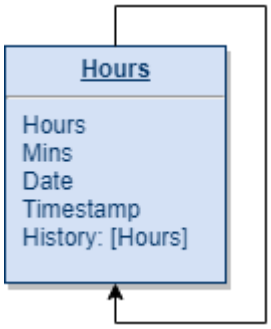

Figura 6: Modelo de datos Power-up Trello

4. Descripción de la solución

En este capítulo se ahonda en las decisiones arquitecturales, de diseño y el ambiente de producción escogido para el despliegue de la solución desarrollada.

4.1 Arquitectura del software

El sistema desarrollado se basó en una arquitectura de una aplicación Back-end servidor que se preocupa principalmente de proveer y persistir datos a través de consultas HTTP, una aplicación web front-end estática que concentra parte importante de la lógica y una cola basada en Redis para tareas rutinarias periódicas de sincronización de los datos. Un diagrama general de la arquitectura del sistema se puede observar en la Figura 7.

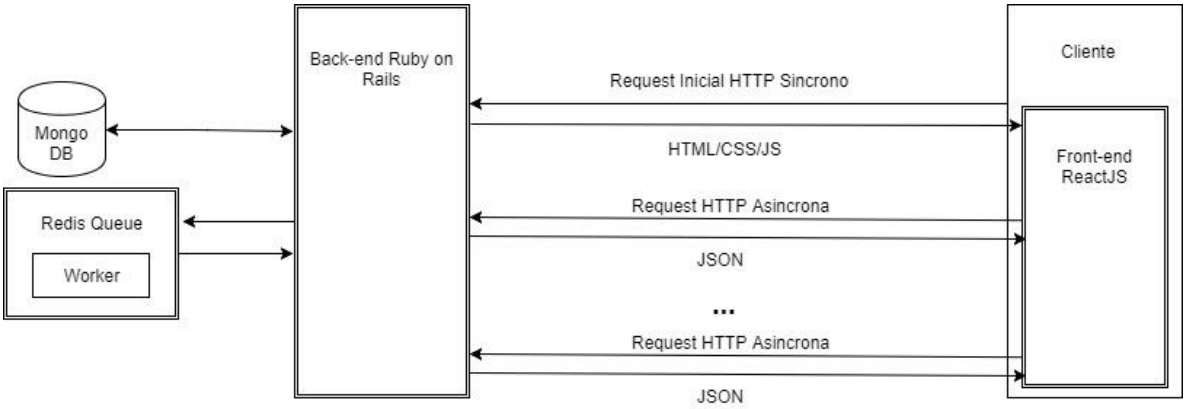

Figura 7: Arquitectura general

4.1.1 Back-end

La arquitectura por el lado del servidor se basó en API REST (ver Figura 8) con el fin de que la comunicación con el front-end fuera simple y basada en recursos accesibles directamente desde una URI, y alimentar así la capa del cliente. Se generaron controladores para los tipos de recursos accesibles del modelo de datos en el front-end, incluyendo recursos para las operaciones CRUD (create, read, update, delete) y otros más dependiendo de las necesidades de las vistas y componentes del front-end.

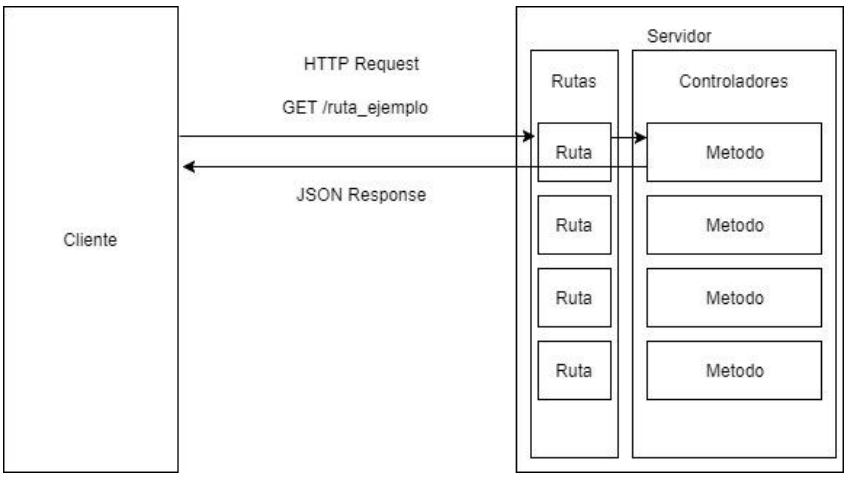

Figura 8: Estructura de la API REST

4.1.2 Cola

Parte de la arquitectura de la solución es una cola FIFO (first in, first out) que se encarga de tomar ciertos trabajos que requieren un nivel de carga un poco mayor debido a su volumen. La principal y única tarea de la cola es mantener una sincronización de los proyectos con sus tableros de Trello correspondientes, por lo que se provee una instancia de Redis junto con un worker que consume la cola a medida que entran trabajos específicos.

La razón de la necesidad de tener una cola que se haga cargo de estos trabajos es que la API de Trello no permite consultar por los datos extra que agrega un Power-up a un recurso (en este caso a una tarjeta) de manera masiva, es decir, no es posible preguntar por los datos agregados por Power-up de todas las tarjetas, ni esta información está presente en la consulta estándar por las tarjetas de un tablero. Debido a lo anterior, es necesario preguntar por cada tarjeta del tablero si se requiere obtener la información añadida por un Power-up, por lo que considerando el volumen de tarjetas, dejaría a la aplicación corriendo un ciclo muy largo en caso de no tener una cola.

4.1.3 Front-end

Los componentes de React manejan su propio estado interno (objetos Javascript) y son capaces de transmitir esta información a través de propiedades a sus hijos. Cuando las aplicaciones React crecen, a veces esta capacidad no es suficiente para mantener un orden coherente en la información que maneja, además de no ser capaz de compartir información obtenida por un componente de manera más global con otros componentes no necesariamente hijos.

Para resolver esto, se utiliza una arquitectura utilizada por Facebook llamada Flux [23], que propone un flujo de datos unificado para el front-end. Existen varias implementaciones de esta arquitectura para React, pero la más popular y utilizada es Redux [24].

Redux separa el flujo de datos en distintos actores: un *store* que contiene el estado completo de la aplicación, acciones que permiten gatillar cambios en el *store* desde los componentes y *reducers*: funciones que hacen match las acciones dependiendo de estas para cambiar el estado del *store*. Típicamente las acciones se encargan de las request HTTP, pudiendo así informar al *store* de los cambios. Con esta librería los componentes pueden suscribirse a ciertos elementos del estado global de la aplicación e importar acciones que le resultan útiles para realizar cambios, enviar o traer información.

Por convención de la comunidad y utilidad general, se suelen separar los componentes en dos tipos: componentes inteligentes y componentes tontos. Se llama componente inteligente a un componente que es consciente del estado global de la aplicación (se suscribe a cambios del *store*), mientras que los componentes tontos solo están conscientes de su propia información, aunque pueden recibir información de sus padres dinámicamente. Los componentes inteligentes suelen llamarse contenedores o containers. El flujo general del front-end para los contenedores puede verse en la Figura 9 [25].

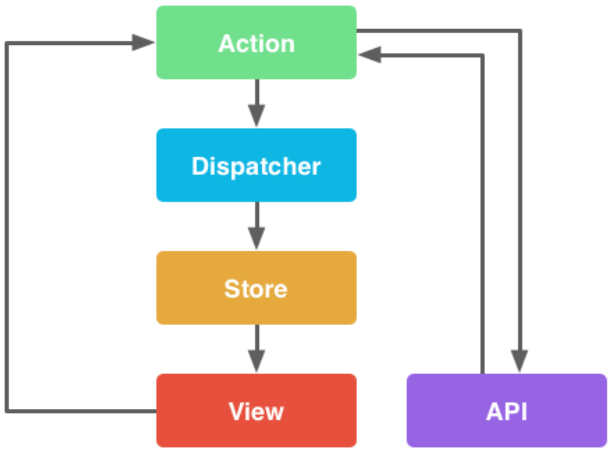

Figura 9: Flujo del front-end

4.1.4 Motor de base de datos

La base de datos MongoDB utilizada para esta aplicación se encuentra alojada por Mongo Atlas [27], servicio de MongoDB que permite tener clústeres de Mongo en la nube, accesibles por parámetros provistos por ellos.

Se escogió una base de datos externa en la nube con el fin de tener cierta aislación de la base de datos con la capa del back-end y que estos sean independientes. De esta manera, es fácil migrar la capa de servidor desde un lugar a otro, debido a que se conecta a la base de datos mediante internet, y no es necesario tener una instancia de Mongo corriendo en el ambiente del servidor. Por otro lado, esta separación hace que sea muy fácil conectar nuevas aplicaciones a la base de datos.

4.2 Ambiente de producción

Para desarrollo ambas partes del software (back y front) viven en el mismo directorio y el mismo repositorio, pero en la práctica para el despliegue estas partes son dos aplicaciones independientes y aisladas que se comunican principalmente por requests HTTP, por lo que al desplegarse se separan de manera automatizada a través del servicio de entrega continua de SemaphoreCI.

4.2.1 SemaphoreCI

La plataforma de Semaphore permite configurar distintos proyectos para ser construidos en sus instancias en la nube. En este caso, se crearon dos proyectos distintos, uno para el back-end y otro para el front-end, debido a que separa mejor la posibilidad de hacer despliegue en producción de cada uno de ellos a demanda del equipo de Synaptic, además de simplificar la configuración de despliegue en general.

En ambos proyectos Semaphore se encarga de que puedan descargar e instalar sus dependencias de manera correcta en un ambiente distinto al de desarrollo, otorgando una capa de seguridad a la hora de desplegar el front-end o el back-end en un ambiente productivo.

4.2.2 RoR y más sobre Heroku

El back-end realizado en RoR se desplegó en la plataforma cloud de Heroku debido a su simpleza y conocida facilidad para integrar proyectos basados en Rails. Como las instancias de Heroku no son instancias con almacenamiento asociado, no es posible instalar directamente otros servicios para que interactúen con la aplicación. No obstante, Heroku provee de un sistema de extensiones para las aplicaciones que permite conectar la instancia con servicios externos en la nube para obtener conexiones a bases de datos, mailing, analíticas, entre otras cosas. Utilizando las bondades de la plataforma, se le agregó una instancia de Redis a la aplicación back-end que permite el funcionamiento de la cola para procesar los trabajos de sincronización de las tarjetas.

4.2.3 React sobre AWS S3 Bucket

Para desplegar la aplicación front-end se utilizó un Bucket de S3 en Amazon Web Services. S3 permite el despliegue de páginas estáticas de manera simple y rápida, además de exponer el contenido a buena velocidad hacia internet. Como en Synaptic se utilizan varios servicios de AWS de manera cotidiana, fue rápida la configuración necesaria para el despliegue de la aplicación ReactJS.

4.3 Diseño de la solución

En esta sección se ahonda sobre las consideraciones que se tomaron en el diseño de la solución y la explicación de las principales partes del sistema en el front-end y el back-end.

4.3.1 Back-end

En esta sección se abordan los aspectos importantes del diseño del back-end de la solución. En particular, la manera de autenticar a los usuarios y mantener la comunicación de ese aspecto a través del uso de la aplicación, además de listar y explicar brevemente los endpoints o las rutas que expone el servidor para ser consultadas por el front-end.

4.3.1.1 Autenticación

La autenticación de la aplicación se maneja por JWT [26] (JSON Web Token). Cuando el usuario ingresa a la plataforma, envía sus credenciales al único endpoint abierto de la aplicación (*/user_token*) que valida las credenciales y la existencia del usuario. En caso de estar todo correcto, envía como respuesta un token único encriptado que es almacenado por el front-end en el almacenamiento local. Este token guardado localmente se utiliza para validar al usuario ingresado en la aplicación siendo enviado como header en todas las demás consultas posibles del sistema, habilitando al back-end para identificar al usuario y decidir qué responder acorde a eso.

4.3.1.2 Endpoints

A continuación se listan todos los endpoint o rutas posibles a consultar en el back-end del sistema. Todos los endpoint listados requieren que dentro de la consulta HTTP se envíe un header incluyendo el token de autenticación del usuario actual, salvo */user_token*, como se explicó en la sección anterior.

- ***GET /organizations:*** Retorna el listado de organizaciones visibles del sistema.
- ***GET /organizations/:org_id****:* Retorna la organización con el identificador: org_id.

- ***POST /organizations***: Recibe un JSON con la información necesaria (nombre) para crear una organización, la crea y retorna el nuevo listado de organizaciones visibles del sistema.
- ***PUT /organizations/:org_id****:* Recibe un JSON con la información necesaria de una organización, busca la organización con identificador: org_id, y la actualiza con dicha información.
- ***DELETE /organizations/:org_id:*** Vuelve invisible la organización con el identificador: org_id. No se elimina directamente de la base de datos.
- ***GET /organizations/:org_id/yearly_hours:*** Retorna un JSON con un arreglo representando los 12 meses del último año. Dentro de cada mes viene la información de las horas trabajadas en dicho mes.
- ***GET /organizations/:org_id/projects/cards:*** Retorna el listado de proyectos de una organización con el identificador: org_id, incluyendo las tarjetas de las tareas terminadas en cada uno.
- ***GET /projects:*** Retorna el listado de proyectos visibles en el sistema.
- ***GET /projects/:pr_id:*** Retorna el proyecto particular con identificador: pr_id.
- ***POST /projects:*** Recibe un JSON con la información necesaria (nombre, ID de organización, identificador de tablero de Trello), crea el proyecto y retorna el nuevo listado de proyectos visibles.
- ***PUT /projects/:pr_id:*** Recibe un JSON con la información necesaria de un proyecto, busca el proyecto con identificador: pr_id, y lo actualiza con dicha información.
- ***DELETE /projects/:pr_id:*** Vuelve invisible el proyecto con el identificador: pr_id. No se elimina directamente de la base de datos.
- ***POST /projects/:pr_id/new_card:*** Recibe un JSON con la información para ingresar una nueva tarjeta/tarea a un proyecto (nombre, descripción, identificador de proyecto y tipo/label), creando la nueva tarjeta.
- ***GET /projects/:pr_id/labels:*** Retorna las etiquetas del proyecto con identificador: pr_id.
- ***GET /projects/:pr_id/last_tasks:*** Retorna las últimas tareas del proyecto, separándolas entre la semana actual y lo anterior, limitando éstas a 30.
- ***GET /users:*** Retorna el listado de usuarios visibles del sistema.
- ***GET /users/:usr_id:*** Retorna el usuario con identificador: usr_id.
- ***POST /users:*** Recibe un JSON con la información para ingresar un nuevo usuario en el sistema (correo electrónico, identificador de organización y rol), creándolo y retornando el nuevo listado de usuarios visibles.
- ***PUT /users/:usr_id:*** Recibe un JSON con la información necesaria de un usuario, busca el usuario con identificador: usr_id, y lo actualiza con dicha información.
- ***DELETE /users/:usr_id:*** Vuelve invisible el usuario con identificador: usr_id. No se elimina directamente de la base de datos.
- ***POST /user_token:*** Recibe un JSON con las credenciales (email y password) de un usuario, valida si el usuario existe en la base de datos y en caso de existir retorna un token asociado a este para ser utilizado como autenticación.

- ***GET /is_token_valid:*** Valida el token presente en el request y retorna una respuesta acorde.
- ***GET /user_data:*** Retorna información básica del usuario actual.
- ***GET /roles:*** Retorna los roles asociados al usuario actual.
- ***GET /last_tasks:*** Si el usuario es Synaptic, retorna las últimas tareas de los proyectos visibles, con la separación análoga a las últimas tareas de un proyecto. En caso de que el usuario sea Cliente, retorna las últimas tareas de los proyectos visibles de la organización del cliente, en el mismo formato explicado.

4.3.1.2 Jobs

Existe un solo job o trabajo representado por una clase Ruby, que se encarga de obtener los datos extra ingresados a una tarjeta de Trello a través del Power-up. Este trabajo consulta una URI específica con el identificador único de la tarjeta de Trello que entrega esta información particular, para luego actualizar la tarjeta con las horas de trabajo ingresadas además de la fecha en que se realizó el trabajo según lo ingresado por un usuario del equipo de Synaptic.

4.3.2 Front-end

En esta sección se discuten las consideraciones de diseño principales que se tomaron para el desarrollo del front-end.

4.3.2.1 Ruteo interno

Debido a que la aplicación front-end que genera ReactJS es una sola página (un index.html con todo el Javascript), el ruteo convencional no funciona, pues no va a alcanzar ningún otro recurso ingresando a más recursos por la URL.

Para manejar esto, existe una librería llamada React Router [28], la cual permite parear URLs con componentes de React. De esta manera, al ingresar por ejemplo a /summary, renderiza la componente de resumen sin volver a cargar la página (recordemos que la página se carga una sola vez). La librería permite anidar rutas y por lo tanto componentes, dando mucha flexibilidad.

4.3.2.2 Autorización

El front-end del sistema maneja la autorización de los usuarios, es decir una vez ingresados, manejar los permisos de qué partes pueden ver y cuáles no. La autorización se consiguió generando un componente de ReactJS que se utiliza para encapsular otros componentes. Este no renderiza nada por sí solo, pero renderiza a su hijo (componente que encapsula) en caso de cumplirse ciertas condiciones.

A las rutas de React se les ingresó una propiedad donde se lista qué tipo de usuarios pueden ingresar al componente que renderiza dicha ruta. Al comenzar el flujo de montaje del componente de autorización, este revisa la validez del token ingresado contra el back-end, manteniendo la autorización por el lado del front-end, como también valida los permisos del usuario en caso de ser válido para decidir si continuar el flujo y renderizar al componente hijo o enviar al usuario a una vista de no autorizado o derechamente al login en caso de no tener un token válido.

4.3.2.3 Componentes

El diseño de la estructura de componentes se basó en la idea de tener un componente inteligente por cada pantalla importante de la aplicación, delegando las responsabilidades importantes de suscripción a la información del *store* a estos en particular. La aplicación completa está englobada por un componente corriente llamado *Main* que inyecta las dependencias necesarias para que la librería Grommet afecte al resto de las componentes de manera efectiva.

En el diagrama general de componentes (Figura 10) se puede ver la estructura general de la aplicación y como los componentes encapsulan a sus hijos. En el cuadro de cada componente, se listan las propiedades que manejan estos, marcadas con un **+**, además de las acciones que pueden gatillar para afectar el estado de la aplicación con un **-**. Los componentes corrientes o tontos no importan acciones para manejar el store, pero sí pueden recibir funciones, objetos y todo tipo de información a partir de sus padres, en forma de propiedades.

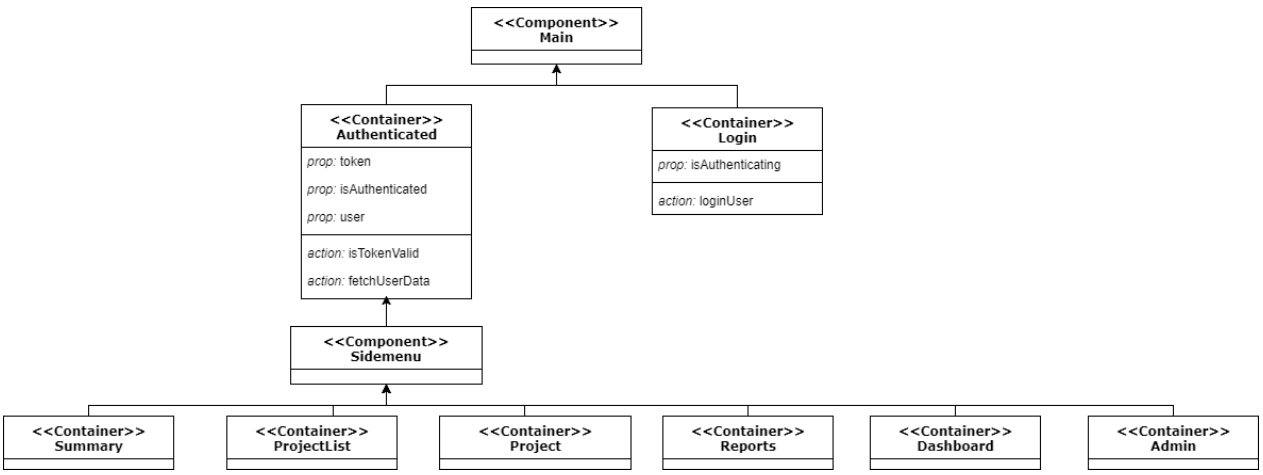

Figura 10: Estructura general de componentes

5 Implementación de la solución

En esta sección se hablará de la solución implementada, además de mostrar las pantallas de la aplicación, diferenciando por tipos de usuarios.

5.1 Trello Power-up

El nuevo Power-up se puede agregar a un equipo de Trello mediante la configuración del tablero. Una vez agregado a este, se vuelven disponibles nuevas opciones en las tarjetas que permiten al usuario del tablero interactuar con nueva información para las tareas (en particular, las horas trabajadas). La Figura 11 muestra los botones mencionados en el detalle de una tarjeta.

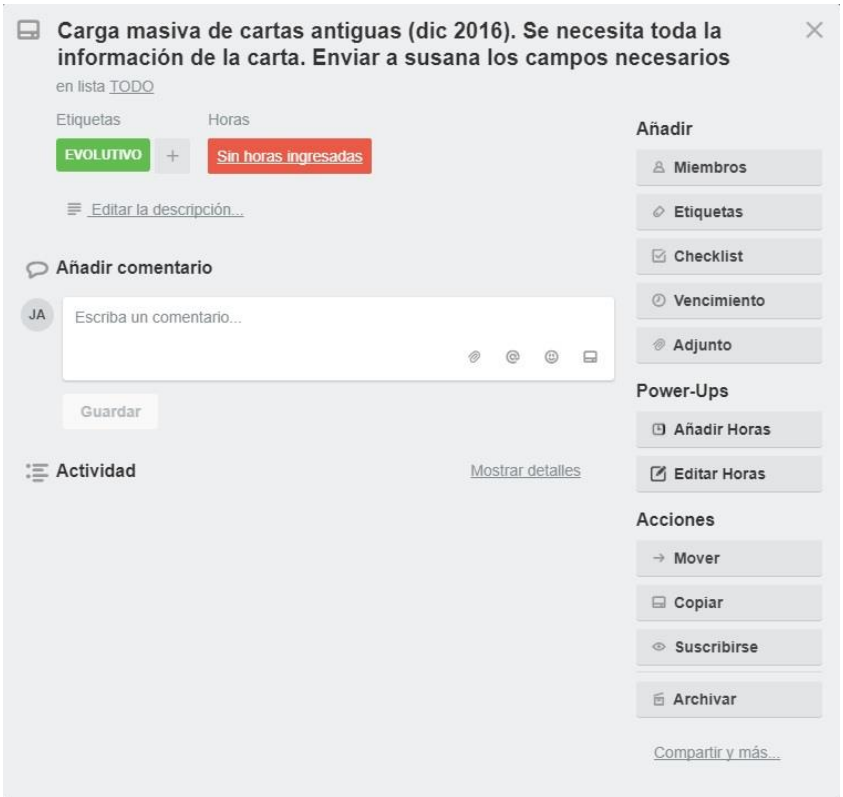

Figura 11: Tarjeta de Trello con Power-up

Se disponibilizan dos botones nuevos: (1) añadir horas, y (2) editar horas. Añadir horas permite al usuario sumar tiempo de trabajo a la tarea por sobre lo agregado anteriormente, mientras que Editar horas permite al usuario reemplazar lo ingresado por nuevos valores. El caso de Añadir horas se puede ver en la Figura 12, mostrando un

calendario con la fecha del trabajo y los campos para agregar horas y minutos. El formulario para editar horas es muy similar al de añadir, por lo que se omitirá en este caso.

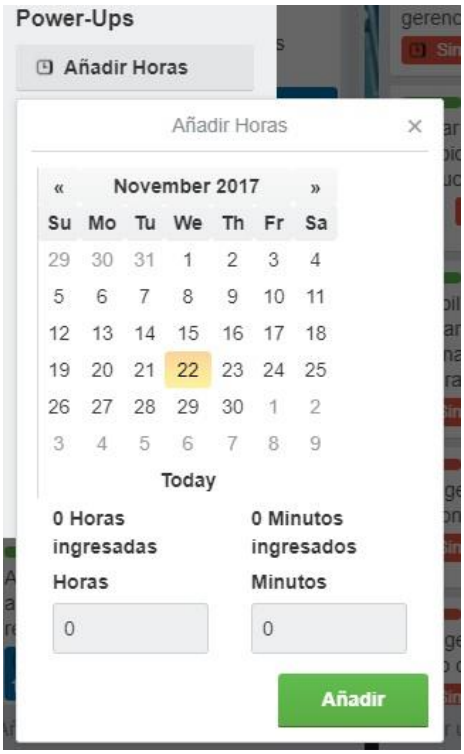

Figura 12: Power-up Trello - Añadir horas

5.2 Usuario Cliente

Esta sección relata el uso de la aplicación desde el rol de usuario del cliente, mostrando cada vista que es accesible para este tipo de usuario.

5.2.1 Resumen de tareas

Al ingresar a la aplicación con sus credenciales, el usuario se encuentra con una vista de resumen que lista las últimas tareas realizadas de los proyectos pertenecientes a su organización. Las tareas del resumen se separan en la semana actual y lo anterior a la semana, dándole énfasis al trabajo que está siendo realizado.

Como se puede ver en la Figura 13, las tareas aparecen ordenadas cronológicamente, partiendo de la más actual. Desde el listado de tareas el usuario puede acceder fácilmente al detalle de cada proyecto en caso de que quiera verlo.

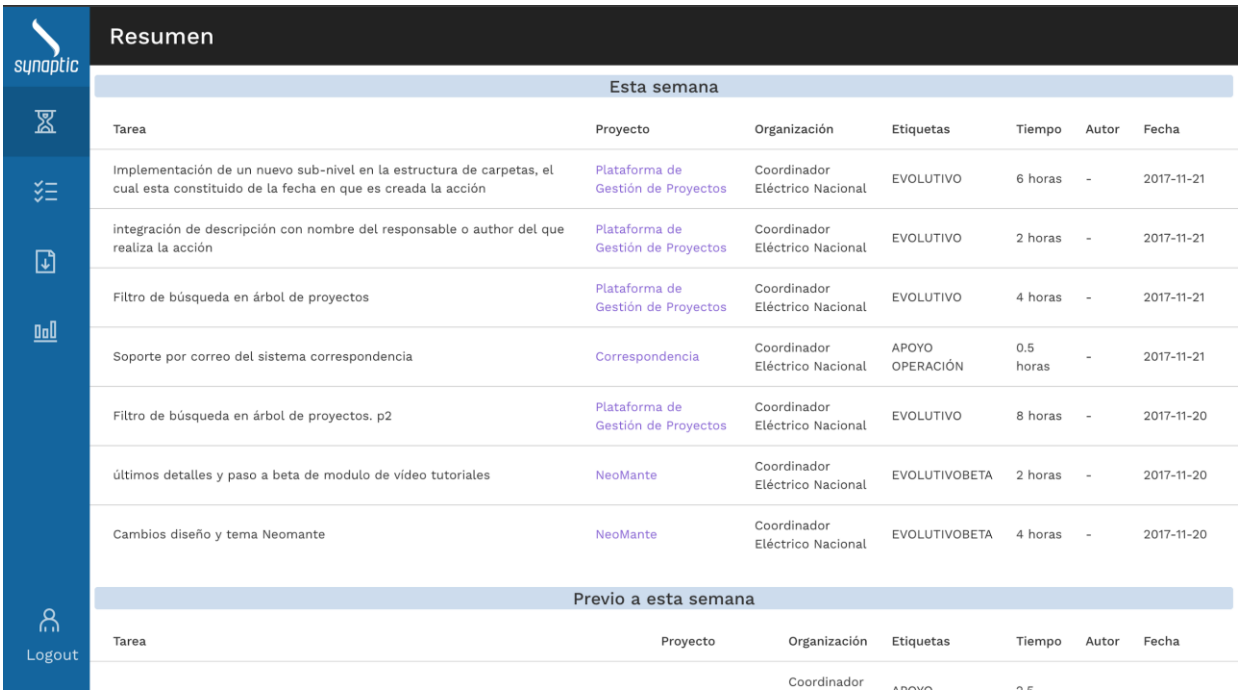

Figura 13: Resumen de tareas

5.2.2 Listado de proyectos

Al clickear en el segundo ícono del menú lateral, el usuario ingresa al listado de proyectos activos. Como se puede ver en la Figura 14, esta lista muestra los proyectos que pertenecen a la organización del usuario, desde la cual puede acceder al detalle de cada uno de ellos haciendo click en cualquiera.

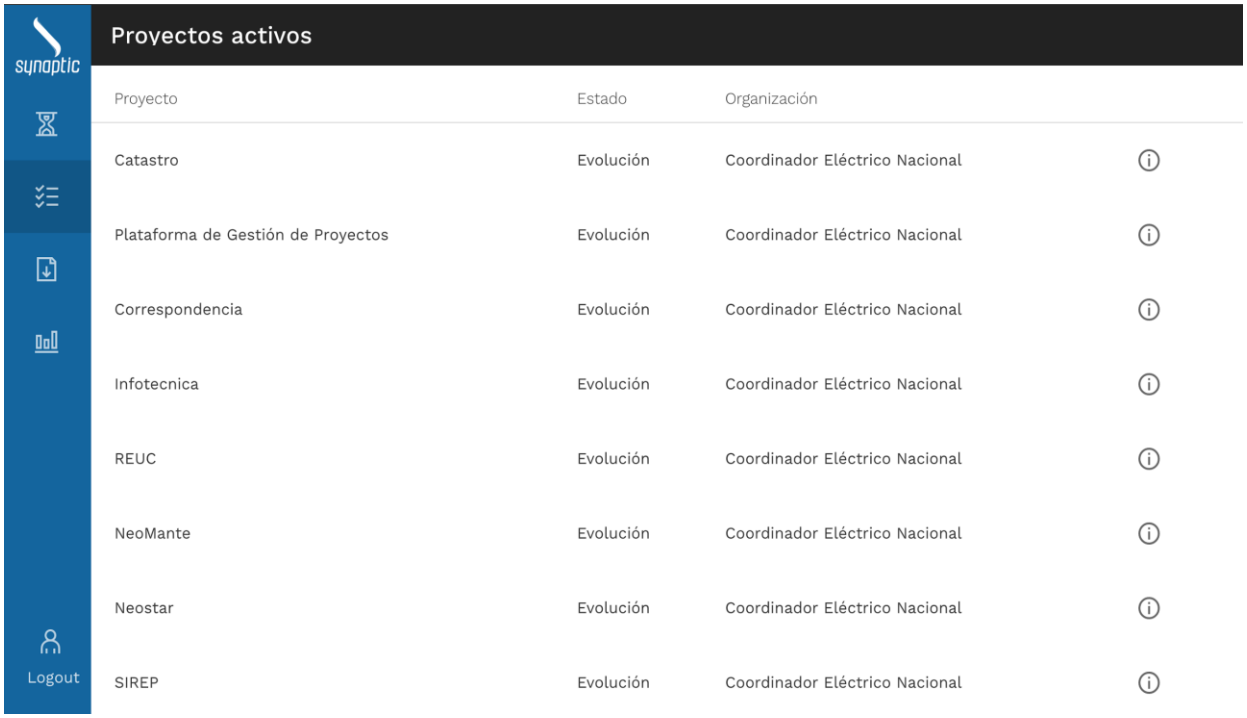

Figura 14: Listado de Proyectos

5.2.3 Detalle de proyecto

Al entrar en el detalle del proyecto, se muestra un gráfico que muestra el total de tareas realizadas en ese proyecto, además de las horas trabajadas en este. Además, separa la naturaleza de las tareas y las horas trabajadas, separando los tipos evolutivos, de error y otros.

Por otra parte, se incluye un resumen de las últimas tareas realizadas de dicho proyecto, haciendo separación en la semana actual y lo anterior a ésta. Lo descrito con previamente se puede apreciar en la Figura 15.

El detalle del proyecto muestra un botón que permite al usuario ingresar nuevos requerimientos a dicho proyecto, mostrándose un modal con el formulario con la información necesaria que deberá llenar el usuario para generar una tarea en el tablero de Trello correspondiente. El formulario se puede ver en la Figura 16.

Figura 15: Detalle de proyecto

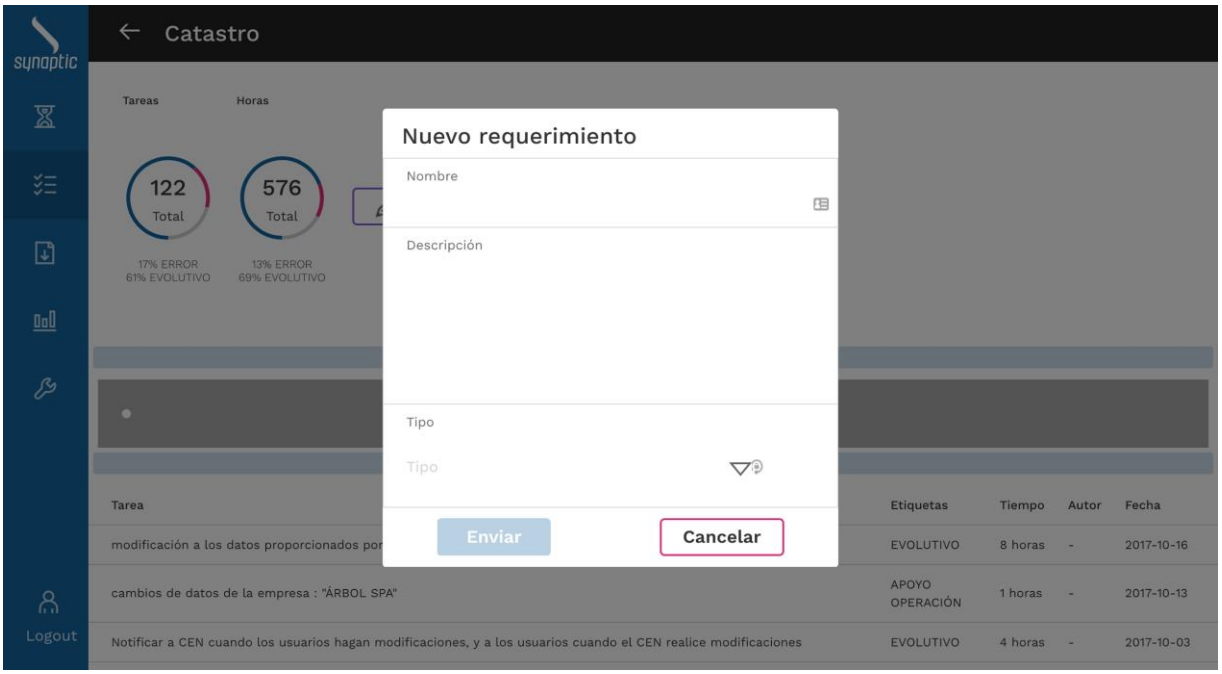

Figura 16: Ingreso de nuevo requerimiento

En el formulario, el usuario debe ingresar el nombre del requerimiento, una descripción libre y el tipo de requerimiento, que se divide según las etiquetas estándar que utiliza el equipo de Synaptic en todos sus tableros. En caso de que el usuario ingrese un requerimiento de tipo Error, se envía un correo electrónico con la información del requerimiento y con la URL directa a la tarjeta de Trello, informando al equipo de Synaptic.

5.2.3 Reportes

La tercera opción del menú lleva al usuario a la pantalla de reportes, donde puede escoger múltiples proyectos de los pertenecientes a su organización para agregarlos a una vista previa de la información que será agregada al reporte.

La pantalla cuenta con un filtro de fechas que permite al usuario acotar el rango temporal para reporte, filtrando también la vista previa de este. Sumado a lo anterior, al seleccionar un proyecto del listado, se agrega un gráfico que muestra el total de tareas y horas del proyecto seleccionado, además de su proporción en naturaleza de error o evolutiva, como se puede apreciar en la Figura 17.

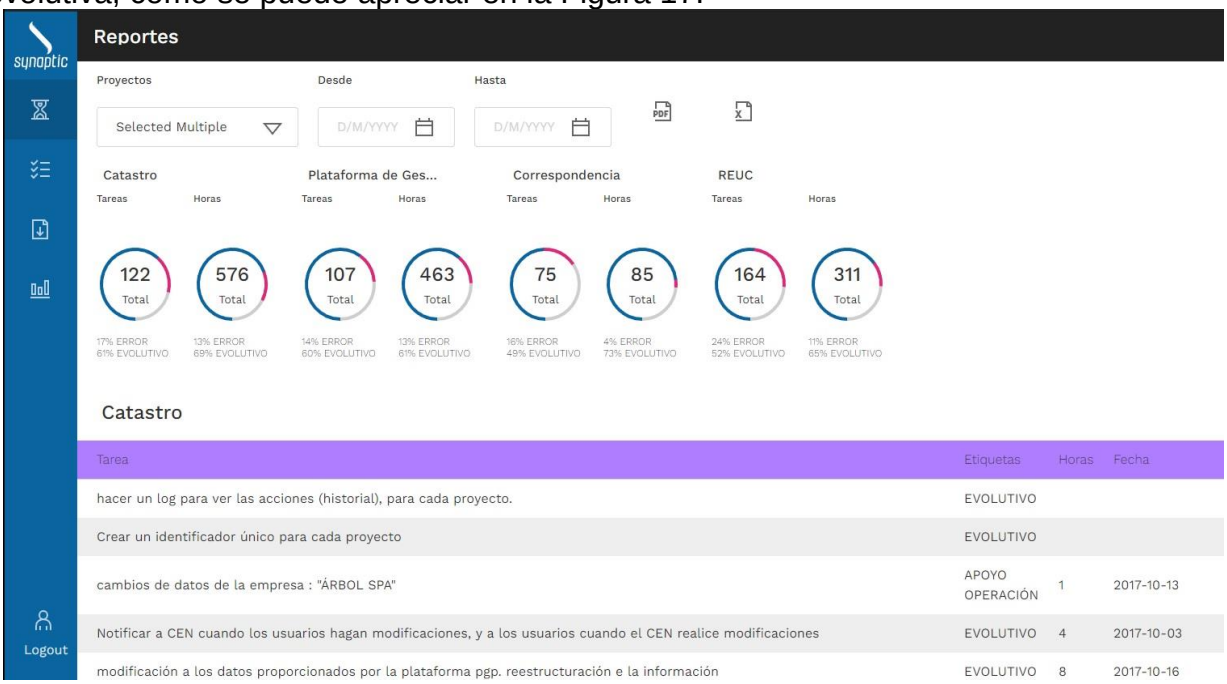

Figura 17: Reportes de la herramienta

Luego de seleccionar el o los proyectos sobre los cuales desea generar un reporte, el usuario puede escoger si desea un reporte en formato PDF o en Excel, haciendo click en los iconos respectos que aparecen en la figura anterior.

En la Figura 18 se puede ver un ejemplo del reporte en formato PDF generado por la selección realizada en la figura anterior (Figura 17). Este reporte muestra un resumen inicial con el total de totales de las horas y tareas de los proyectos durante el periodo de tiempo escogido, además de la proporción de evolución y error que presentan. Luego del resumen, se lista el detalle de las tareas realizadas agrupadas por cada proyecto. El reporte en formato Excel contiene la misma información que el reporte en formato PDF, pero distribuye el resumen y los proyectos en hojas de cálculo distintas.

Resumen

Fecha: Inicial: 1/11/2017 Fecha Final: 22/11/2017
Proyectos considerados: Plataforma de Gestión de Proyectos, Correspondencia, REUC
Total Horas: 161 Horas Error: 0 (0%) Horas Evolutivas: 141 (88%) Horas Otros: 20 (12%)
Nº Tareas: 41 Tareas Error: 0 (0%) Tareas Evolutivas: 32 (78%) Tareas Otros: 9 (22%)

Plataforma de Gestión de Proyectos

Fecha Inicial: 1/11/2017 Fecha Final: 22/11/2017
Total horas: 125 Horas Error: 0 (0%) Horas Evolutivas: 108 (86%) Horas Otros: 17 (14%)
Nº tareas: 29 Tareas Error: 0 (0%) Tareas Evolutivas: 25 (86%) Tareas Otros: 4 (14%)

Tarea	Etiquetas	Horas	Fecha
Filtro de búsqueda en árbol de proyectos. p2	EVOLUTIVO	8	2017-11-20
Filtro de búsqueda en árbol de proyectos. p1	EVOLUTIVO	5	2017-11-16
Paso a beta de últimos cambios	APOYO OPERACIÓN	1	2017-11-15
Añade Manual de Usuario p1	APOYO OPERACIÓN	7	2017-11-13
Añade Manual de Usuario p2	APOYO OPERACIÓN	6	2017-11-14
integración de la funcionalidad que devuelve el listado de las interconnection_request, filtradas por el usuario suplente logueado	EVOLUTIVO	2	2017-11-06
funcionalidad que recibe la información del suplente y procesa para ser almacenada en el modelo interconnection_request	EVOLUTIVO	4	2017-11-03
listar usuarios administradores para asignar como suplente	EVOLUTIVO	2	2017-11-06
modificación de funcionalidad para enviar las request con información del substitutos al back	EVOLUTIVO	4	2017-11-03
cambios en el front para ingresar substitutos del administrador	EVOLUTIVO	4	2017-11-06
diseñar en el front de la aplicación la opción de editar el correlativo del proyecto	EVOLUTIVO	3	2017-11-09
integracion de las llamadas a la api del coordinador para obtener información de los correlativos del proyecto	EVOLUTIVO	2	2017-11-09
modificacion de los endpoint que reciben la informacion de un proyecto al crearlo o editarlo (correlativos)	EVOLUTIVO	2	2017-11-07
integración de la vista que muestra el listado de interconnection_request por categoria de "Administrador" y "Suplente"	EVOLUTIVO	8	2017-11-02
funcionalidad para modificar los correlativos, mapear los ya existentes	EVOLUTIVO	6	2017-11-08
funcionalidad para sumar correlativos a los proyectos que se van creando	EVOLUTIVO	2	2017-11-07
funcionalidad para generar los correlativos de un proyecto	EVOLUTIVO	4	2017-11-07
Implementación de un nuevo sub-nivel en la estructura de carpetas, el cual esta constituido de la fecha en que es creada la acción	EVOLUTIVO	6	2017-11-21
integración de descripción con nombre del responsable o author del que realiza la acción	EVOLUTIVO	2	2017-11-21
integración de filtro de busqueda para a estructura de empresas	EVOLUTIVO	8	2017-11-17
implementación en el front que muestra una vista con la estructura de carpetas de un proyecto	EVOLUTIVO	8	2017-11-13
creación de estructura virtual de las carpetas, el cual se encarga de representar de forma visual, donde estarian ubicados los archivos. en nodos	EVOLUTIVO	6	2017-11-14
creación y modificación de los endpoint que responden la estructura de arbol de las carpetas	EVOLUTIVO	4	2017-11-15
modificación del modelo de Accion, se le implementan nuevos atributos y funciones para retornar el valor de referencia a los template(task y requirement)	EVOLUTIVO	4	2017-11-15

Figura 18: Ejemplo reporte PDF

5.2.4 Dashboard

El cuarto ítem del menú lleva al usuario a la pantalla de dashboard, donde el usuario puede ver gráficos relacionados al desarrollo de los proyectos en curso de su organización. Se presentan dos gráficos, uno relacionado al comportamiento de las horas invertidas en los proyectos y otro a la naturaleza de las tareas en estos.

5.2.4.1 Horas

En la Figura 19 se puede ver un gráfico de distribución de horas en los proyectos de la organización, acotado a un rango de un año calendario hasta un año atrás. Este gráfico permite al usuario comparar el esfuerzo en horas impartido en cada proyecto, permitiendo tener una mejor imagen global y tomar decisiones a partir de esto.

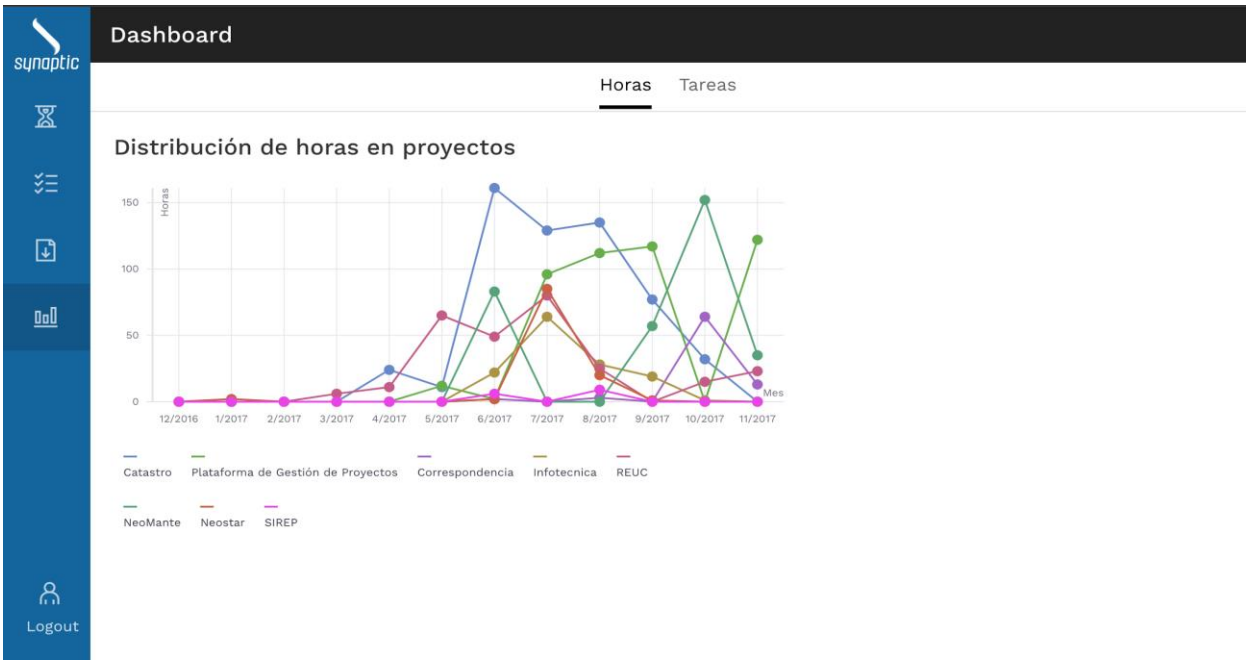

Figura 19: Dashboard distribución de horas

5.2.4.2 Tareas

En la pestaña de Tareas, se muestra un gráfico que compara todos los proyectos de la organización, mostrando su proporción de tareas evolutivas y de error lado a lado. Esto permite al usuario visualizar de manera simple cuando un proyecto puede necesitar atención (por ejemplo, un proyecto que tiene una cantidad de tareas de error considerable

en comparación al resto). En la Figura 20 es posible ver el gráfico de Distribución de Tareas.

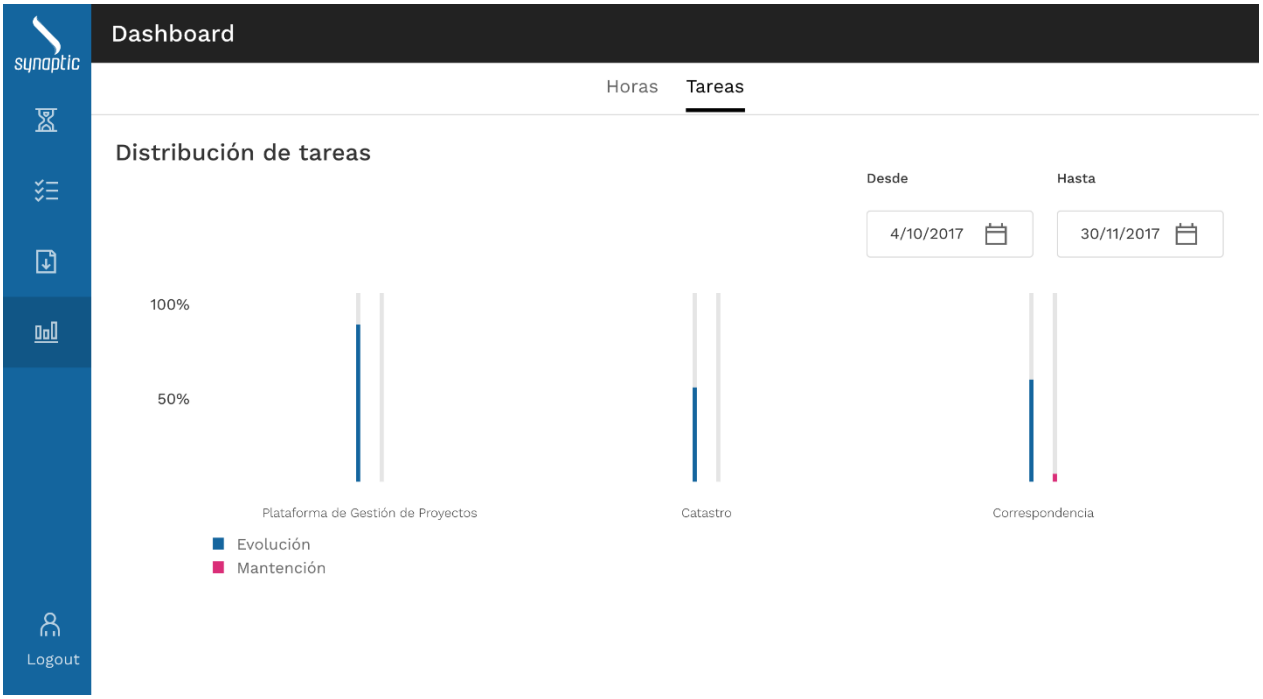

Figura 20: Dashboard distribución de tareas

5.3 Usuario Synaptic

El usuario Synaptic tiene acceso a todas las pantallas e información presente en las secciones anteriores, con la diferencia de que tiene un selector de organizaciones para filtrar. Además de lo anterior, tiene acceso a una quinta sección de la aplicación: el administrador. El administrador se divide en tres pestañas: organizaciones, proyectos y usuarios. Se explicará lo anterior a continuación.

5.3.1 Administrador de organizaciones

Al entrar al administrador el usuario se enfrenta a la pestaña de organizaciones. En esta parte del administrador se pueden crear, editar y eliminar organizaciones del sistema. Las organizaciones aparecen en manera de listado, como puede verse en la Figura 21. Allí al hacer click en el botón de crear, o en el lápiz para editar en cada proyecto, se despliega un modal con el formulario para crear o editar una organización. Para ambos casos el formulario es el mismo, solo que en el caso de editar el modal viene pre-llenado con la información de la organización seleccionada, lo que se puede apreciar en la Figura 22.

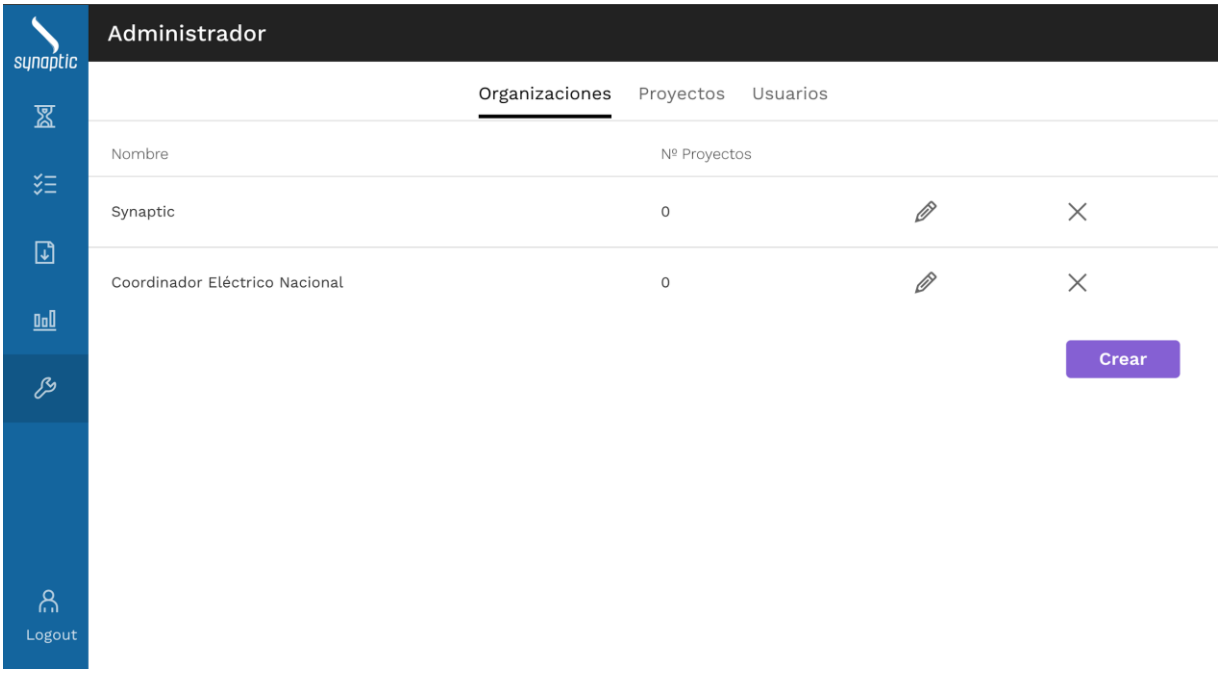

Figura 21: Administrador de organizaciones

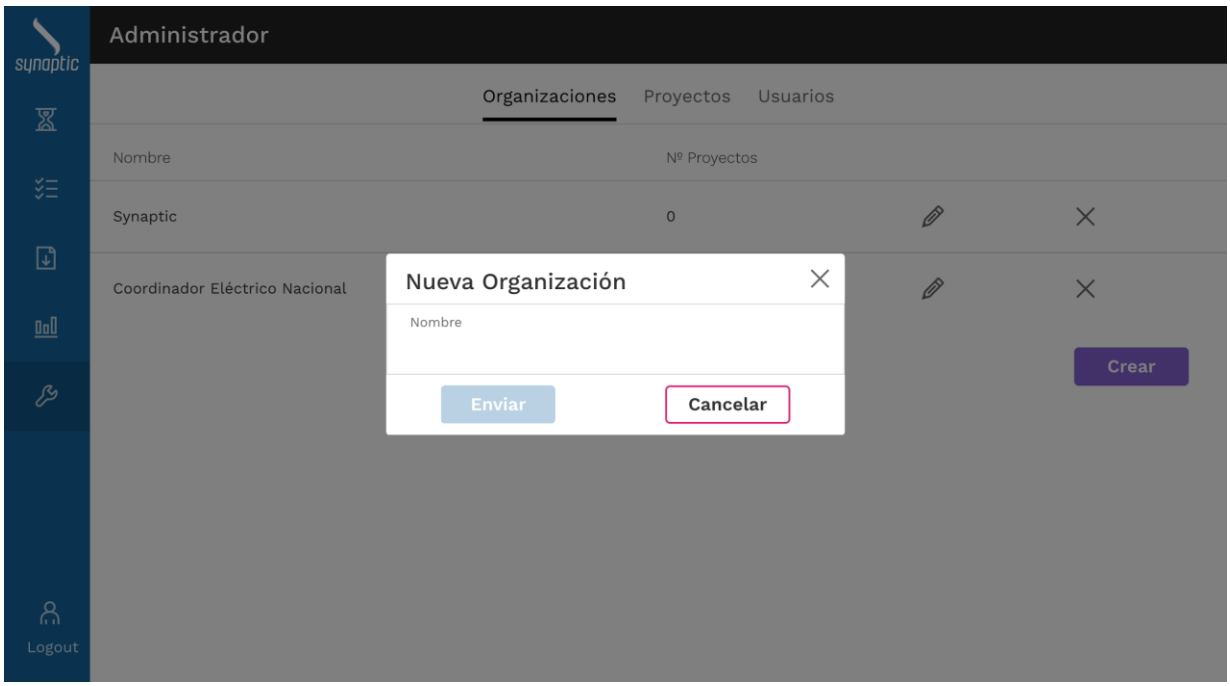

Figura 22: Modal de creación/edición de organizaciones

5.3.2 Administrador de proyectos

La segunda pestaña despliega el administrador de proyectos, que muestra el listado de los proyectos ingresados en el sistema (Figura 23). En esta vista el usuario puede crear, editar o eliminar proyectos de la plataforma.

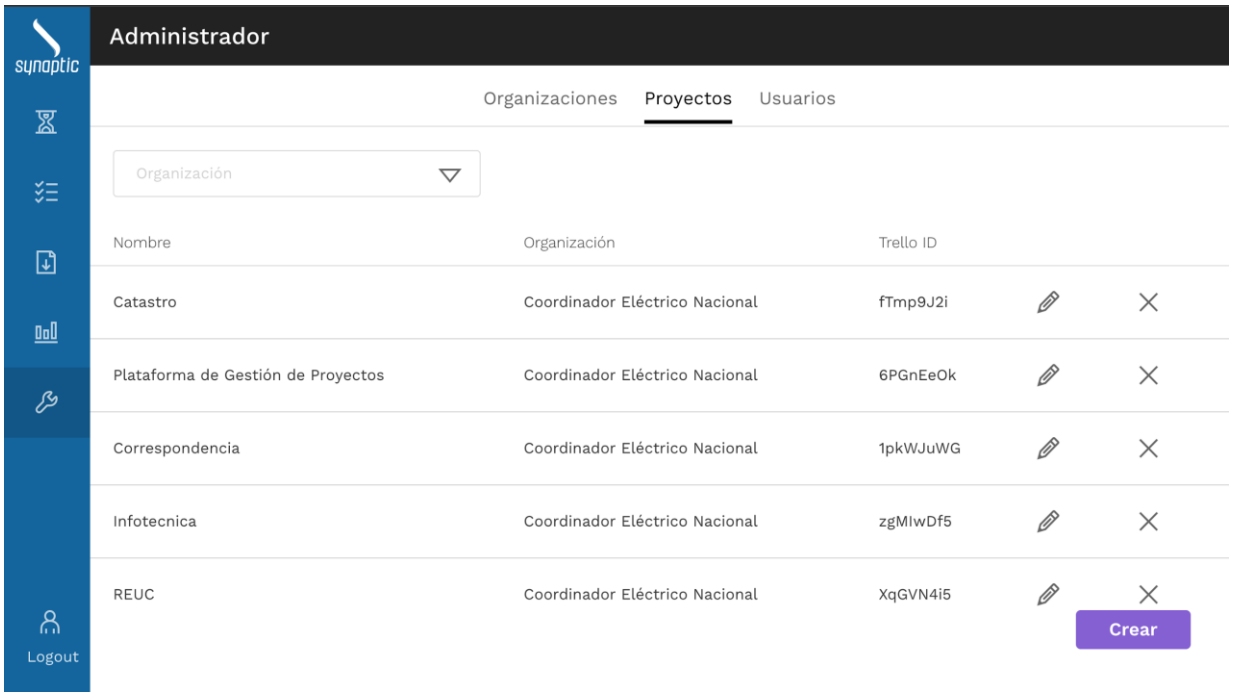

Figura 23: Administrador de proyectos

Al clickear en el botón crear o en el icono de lápiz en cada proyecto, se despliega un modal con el formulario para ingresar o editar un proyecto. La información necesaria para ingresar un proyecto se puede ver en la Figura 24. Para que la plataforma asocie la información de Trello a un proyecto es necesario que se ingrese el identificador único de tablero.

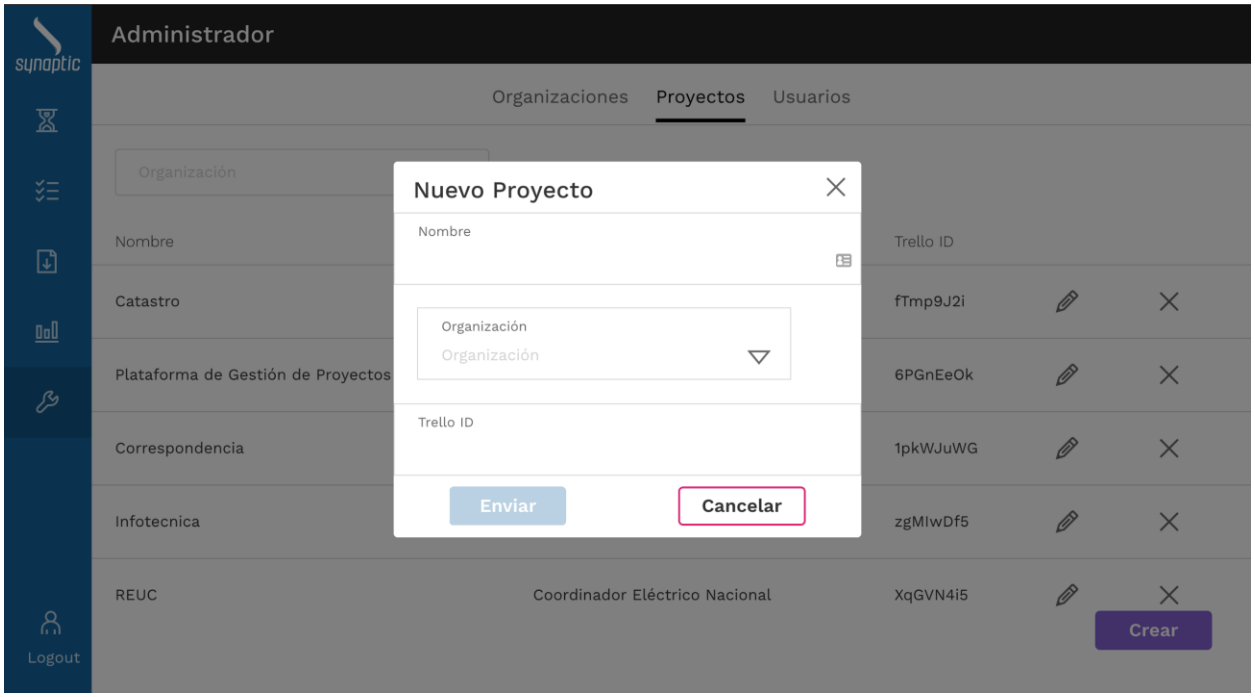

Figura 24: Modal de creación/edición de proyectos

5.3.3 Administrador de usuarios

La última pestaña del administrador lleva al administrador de usuarios, donde se listan estos con su información más importante (Figura 25). Acá se pueden ingresar, editar o eliminar usuarios del sistema.

Al hacer click en el botón de crear o en el ícono de lápiz de editar en un usuario, se despliega el modal con el formulario para la información necesaria para un usuario, como se puede ver en la Figura 26.

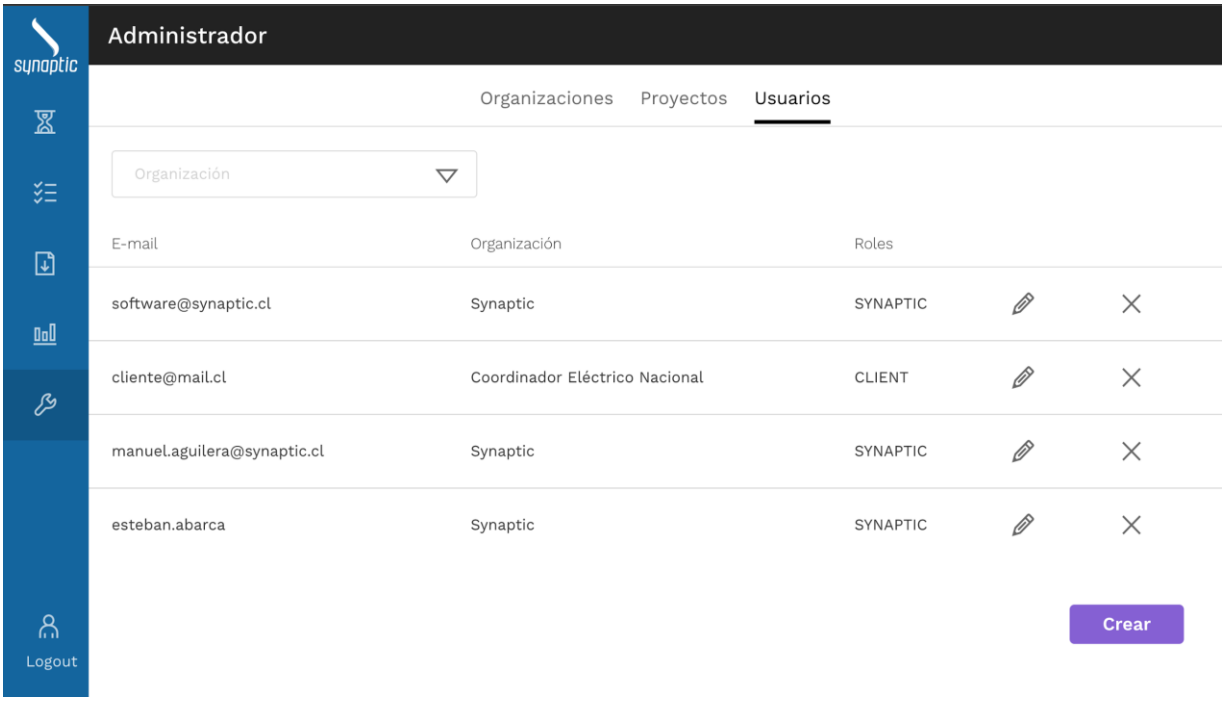

Figura 25: Administrador de usuarios

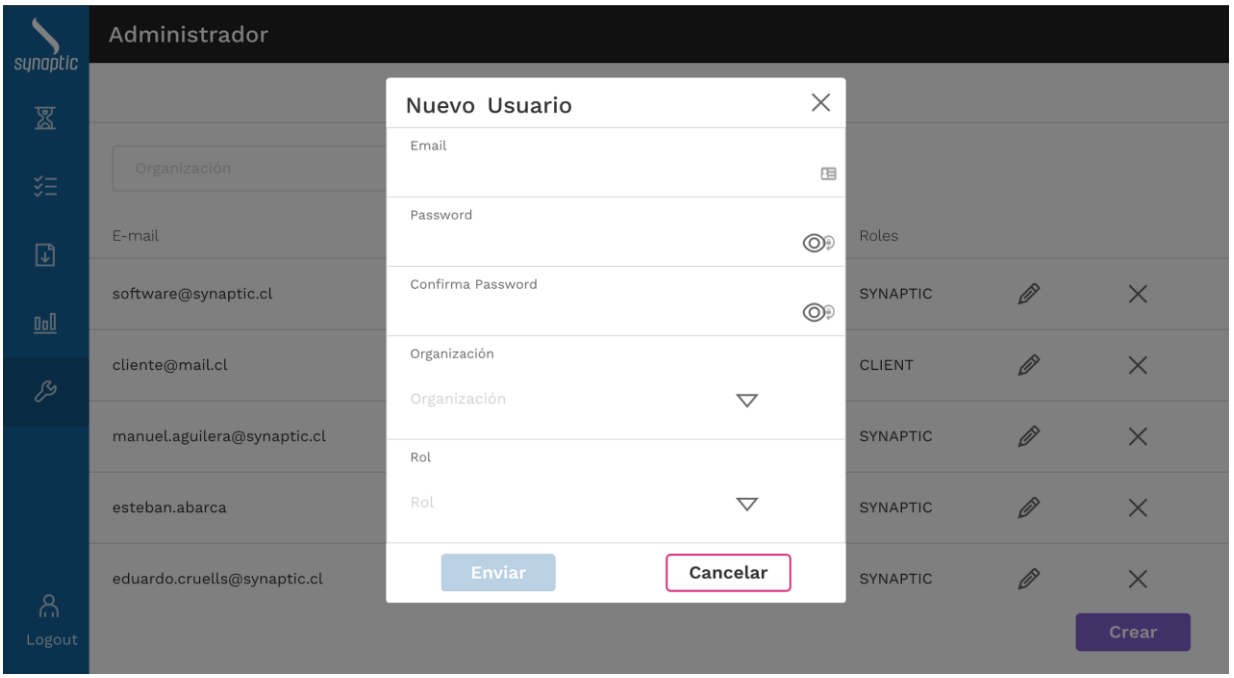

Figura 26: Modal de creación/edición de usuarios

6. Validación de la solución

Para la validación y evaluación de la plataforma se realizaron diversas pruebas de uso durante el proceso de desarrollo, una encuesta final y se recogió feedback por parte de jefes de proyecto del lado del cliente, o sea, del Coordinador Eléctrico Nacional.

La plataforma se disponibilizó de manera temprana en la nube, con el fin de ir comparando el funcionamiento de ésta con el servicio de Everhour, tomando en cuenta la validez de los datos ingresados en el Power-up de Trello, la consistencia de los mismos, la reportería generada y el uso general de la aplicación. Esto permitió validar la usabilidad y encontrar errores de manera temprana.

Con la ayuda del equipo de Synaptic se evaluó el diseño de las partes de la aplicación de manera frecuente, poniendo énfasis en el diseño del modelo de datos y la categorización de la información mostrada. Esto permitió asegurar que la plataforma tenga un diseño escalable para el desarrollo futuro.

Durante todo el proceso de desarrollo la plataforma sufrió modificaciones influenciadas por el equipo, validando grupalmente el diseño de esta de manera iterativa en múltiples ocasiones. Por otra parte, la plataforma se utilizó y revisó hasta llegar al nivel aceptable para dejar de utilizar el servicio de Everhour y usar directamente esta plataforma. Así se brindó la transparencia de horas y tareas trabajadas, además de la reportería del estado de los proyectos para el cliente, resultando en un éxito este proyecto.

Por parte de la evaluación de los jefes de proyecto en el Coordinador Eléctrico Nacional, se mostró entusiasmo al ver la iniciativa. Se evaluó de manera satisfactoria la utilidad y la usabilidad del sistema, con buena recepción de las funcionalidades de reporte, resumen de tareas y gráficos generados. La plataforma despertó mucho interés por el lado de la contraparte, dado que comunicaron múltiples cosas que les gustaría desarrollar sobre la plataforma en el futuro.

Finalmente, se realizó una encuesta para evaluar la percepción de la usabilidad, utilidad, cumplimiento del objetivo general de la plataforma y recoger feedback de la aplicación. La encuesta se entregó a miembros de Synaptic y algunos actores clave de la contraparte. Los resultados de la encuesta pueden encontrarse en el Anexo A.

De manera general, la encuesta arrojó buenos resultados en la usabilidad y utilidad del sistema, además de considerar que la plataforma permite llevar un mejor seguimiento de los proyectos. No obstante, hay consenso general de que las interfaces de usuario de la plataforma podrían necesitar ayuda en el aspecto del diseño, y que deben revisarse ciertos aspectos de compatibilidad con diferentes navegadores, debido a que algunos

gráficos se muestran de manera distinta en diferentes navegadores. Las respuestas también indicaron que la plataforma debe resolver ciertos aspectos para ser visualizada correctamente en dispositivos móviles pequeños, como teléfonos inteligentes. Los resultados de la encuesta pueden verse en los Anexos.

7 Conclusiones y trabajo a futuro

Los cambios estructurales que sufrió el principal cliente de Synaptic (o sea, el Coordinador Eléctrico Nacional), golpearon el proceso de desarrollo en varias aristas, por lo que la necesidad de actuar de alguna manera para fortalecer la transparencia, la confianza y el seguimiento de los proyectos se hizo inminente. La plataforma desarrollada logró ser una buena solución a varios de los problemas planteados en el documento, facilitando la labor de jefes de proyecto de la contraparte y de los miembros del equipo de Synaptic, pues les permite visualizar con mayor rapidez la información relevante acerca del estado de los proyectos, además de sentar una base importante de propiedad de la información.

En primer lugar, la plataforma logró desplazar al servicio de Everhour de manera efectiva, cubriendo las funcionalidades que se utilizaban de este último de manera interna. Además, el equipo de Synaptic obtuvo, a través del trabajo del alumno memorista, valiosa información, reflexión y aprendizaje acerca de cómo abordar este problema, y cómo sentar las bases para desarrollos futuros.

Por otro lado, la solución desarrollada presenta una mejor integración con el servicio de Trello que Everhour, debido a que el ingreso de horas trabajadas está integrado de manera nativa en los tableros, funcionando en la aplicación de escritorio y en cualquier navegador; a diferencia de Everhour que funciona a través de una extensión del navegador Google Chrome.

En segundo lugar, se aprendió acerca de cómo integrar funcionalidades personalizadas al servicio otorgado por Trello, a través de su API de Power-ups. Esto definitivamente podría resultar útil en el futuro, para resolver problemas y potenciar nuevas capacidades del proceso de desarrollo.

En tercer lugar, el hecho de que ahora Synaptic sea propietario total de la información de lo trabajado en los proyectos aportó mucho valor actual, y potencial valor en el futuro. Esta información es clave para el seguimiento de proyectos, además de ser de vital importancia en caso de necesitar revisar información del pasado, por lo que depender de información almacenada por terceros representaba un riesgo. Por ejemplo, en caso de que Everhour decidiera no prestar más sus servicios.

En cuarto lugar, se propuso un canal único de comunicación para el ingreso de requerimientos, lo que resultó en un mayor orden y claridad para los clientes a la hora de manifestar sus inquietudes concretas para los proyectos en desarrollo. Esto significó en un mejor proceso de comunicación entre las partes, sobre todo a la hora de ingresar errores con algún nivel de urgencia, debido a que se automatizó la alerta para estos.

En quinto lugar, se comenzó a procesar la información relacionada a las tareas realizadas mediante gráficos. Esto le permite al cliente tener una mejor idea del estado de los proyectos individualmente, y en comparación con los demás. Esto no se hacía en el pasado y sólo podía hacerse a través de un desarrollo interno, tomando los datos provistos por Everhour de manera manual (exportando CSV o Excel). Además, integrar la generación de reportes a la plataforma interna incentiva el uso de esta funcionalidad por parte de los clientes, debido a que ahora no necesita estar consultando múltiples fuentes de información para obtener las mismas conclusiones que en el pasado.

Finalmente, el desplazamiento del servicio de Everhour por un sistema interno significó un ahorro en costos fijos mensuales para Synaptic, puesto que Everhour es un servicio pagado con un costo por usuario en el sistema. Este costo podría haber seguido escalando con el tiempo, en caso de crecer el equipo de desarrollo, y aumentar el número de usuarios interesados en utilizar este servicio por parte del cliente.

Durante el proceso de desarrollo, el uso constante de prototipos de la aplicación, impactó positivamente el diseño de ésta, ya que permitió realizar modificaciones y correcciones de manera temprana. Este desarrollo también permitió la transmisión de los conocimientos y aprendizajes obtenidos por el alumno memorista, hacia el resto del equipo de Synaptic. Por otra parte, la iniciativa fue recibida de manera positiva por el cliente, viendo rápidamente el valor de la idea y plataforma desarrollada, incorporándolo a la discusión de la problemática planteada en este trabajo de manera natural, por lo que se espera que esto signifique en una mejora continua con el paso del tiempo.

A pesar de que la plataforma fue recibida de manera positiva, se presentaron claros aspectos que podrían mejorar y desarrollo futuro basándose en el trabajo realizado por el alumno memorista. Principalmente, la plataforma podría recibir ayuda en el aspecto del diseño gráfico, potenciando una visual más empresarial y acabada, debido a que es un canal de comunicación formal con clientes, potenciando el uso y la experiencia de usuario del trabajo realizado. Por otra parte, se manifestaron variadas ideas para hacer crecer la plataforma en el futuro. Las principales ideas recibidas fueron:

1. Incorporar la estimación de las tareas a realizar en las tarjetas de Trello, con el fin de mejorar el proceso de estimación de manera progresiva, comparando la estimación en horas de las tareas, contra el tiempo real tomado para realizarla.

2. Integrar más gráficos con información resumida o procesada, para mejorar la toma de decisiones en base a la información del trabajo realizado en los proyectos.

3. Integrar estadísticas asociadas a los release de los proyectos para evaluar la calidad del software entregado en cada uno de estos.

Finalmente, entre las cosas que podrían mejorarse en el futuro para la plataforma desarrollada, está el encontrar una mejor estrategia de sincronización de la información

de los tableros con el sistema, puesto que la API de Trello está en constante crecimiento y en un estado relativamente inmaduro. Por lo tanto, algunas de las funcionalidades que ofrece la API podrían optimizarse para realizar menos consultas a ésta.

Bibliografía

1. Coordinador.cl. (2018). *Coordinador Eléctrico Nacional*. Disponible en: https://www.coordinador.cl/ [Última visita 19 de Junio 2018].
2. Trello.com. (2018). *Trello*. Disponible en: https://trello.com/ [Última visita 19 de Junio 2018].
3. Slack. (2018). *Donde tu trabajo fluye*. Disponible en: https://slack.com/intl/es [Última visita 19 de Junio 2018].
4. Everhour.com. (2018). *Everhour — More Transparent Time Tracking*. Disponible en: https://everhour.com/ [Última visita 19 de Junio 2018].
5. Atlassian. (2018). *Jira | Software de seguimiento de proyectos e incidencias | Atlassian*. Disponible en: https://es.atlassian.com/software/jira [Última visita 19 de Junio 2018].
6. Ruby on Rails. (2018). *Ruby on Rails*. Disponible en: http://rubyonrails.org/ [Última visita 19 de Junio 2018].
7. Ruby-lang.org. (2018). *Lenguaje de Programación Ruby*. Disponible en: https://www.ruby-lang.org/es/ [Última visita 19 de Junio 2018].
8. MuleSoft. (2018). *What is REST API Design?*. Disponible en: https://www.mulesoft.com/resources/api/what-is-rest-api-design/ [Última visita 19 de Junio 2018].
9. Djangoproject.com. (2018). *The Web framework for perfectionists with deadlines | Django*. Disponible en: https://www.djangoproject.com/ [Última visita 19 de Junio 2018].
10. Reactjs.org. (2018). *React - A JavaScript library for building user interfaces*. Disponible en: https://reactjs.org/ [Última visita 19 de Junio 2018].
11. Koppers, T. (2018). *webpack module bundler*. Webpack.github.io. Disponible en: https://webpack.github.io/ [Última visita 19 de Junio 2018].
12. Grommet.io. (2018). *Grommet*. Disponible en: http://grommet.io/ [Última visita 19 de Junio 2018].
13. Vuejs.org. (2018). *Vue.js*. Disponible en: https://vuejs.org/ [Última visita 19 de Junio 2018].
14. Vuestic.epicmax.co. (2018). *Vuestic Admin*. Disponible en: http://vuestic.epicmax.co/ [Última visita 19 de Junio 2018].
15. Trello.com. (2018). *Get Connected with Trello Power-Ups*. Disponible en: https://trello.com/power-ups [Última visita 19 de Junio 2018].
16. Heroku.com. (2018). *Cloud Application Platform | Heroku*. Disponible en: https://www.heroku.com/ [Última visita 19 de Junio 2018].

17. Zeit.co. (2018). *Now – Realtime Global Deployments*. Disponible en: https://zeit.co/now [Última visita 19 de Junio 2018].

18. Docker. (2018). *Docker*. Disponible en: https://www.docker.com/ [Última visita 19 de Junio 2018].

19. Redis.io. (2018). *Redis*. Disponible en: https://redis.io/ [Última visita 19 de Junio 2018].

20. GitHub. (2018). *resque/resque*. Disponible en: https://github.com/resque/resque [Última visita 19 de Junio 2018].

21. Docs.aws.amazon.com. (2018). *Working with Amazon S3 Buckets - Amazon Simple Storage Service*. Disponible en: https://docs.aws.amazon.com/AmazonS3/latest/dev/UsingBucket.html [Última visita 19 de Junio 2018].

22. Facebook.github.io. (2018). *Flux | Application Architecture for Building User Interfaces*. Disponible en: https://facebook.github.io/flux/ [Última visita 19 de Junio 2018].

23. Semaphoreci.com. (2018). *Continuous Integration & Delivery - Semaphore*. Disponible en: https://semaphoreci.com/ [Última visita 19 de Junio 2018].

24. Redux.js.org. (2018). *Read Me · Redux*. [online] Available at: https://redux.js.org/ [Última visita 19 de Junio 2018].

25. Alonso, A. (2018). *Developer Training #1: Aprendiendo Redux*. adrianalonso.es. Disponible en: https://adrianalonso.es/formacion/developer-training-1-aprendiendo-redux/ [Última visita 19 de Junio 2018].

26. MongoDB. (2018). *Fully Managed MongoDB, hosted on AWS, Azure, and GCP*. Disponible en: https://www.mongodb.com/cloud/atlas/ [Última visita 19 de Junio 2018].

27. Jwt.io. (2018). *JWT.IO*. Disponible en: https://jwt.io/ [Última visita 19 de Junio 2018].

28. ReactRouterWebsite. (2018). *React Router: Declarative Routing for React*. Disponible en: https://reacttraining.com/react-router/ [Última visita 19 de Junio 2018].

Anexos

Anexo A - Resultados encuesta sobre la plataforma

A continuación se presenta la tabla de resultados sobre la encuesta que se realizó acerca de la plataforma.

Tabla 2: Resultados encuesta realizada

N ° Encuestado	¿Qué le pareció la usabilidad del sistema?	¿La plataforma le parece útil?	¿La plataforma le permite llevar un mejor seguimiento a los proyectos?	¿Qué aspectos positivos encontró en la plataforma?	¿Qué aspectos negativos encontró en la plataforma?	¿Qué puede mejorar en la plataforma?
1	Alta	Sí	Sí	Entrega un buen resumen de gestión para contrastar lo que se ha ejecutado.	Ninguno	Puede mejorar en aspectos comparativos respecto de la estimación inicial v/s lo ejecutado, mejorar la visualización, mejorar el análisis de registro de errores por release de manera de poder evaluar la calidad del software.
2	Alta	Sí	Sí	Reporteria	Actual diseño de los gráficos en reporteria: no se diferencia qué gráfico es de qué proyecto	Actual diseño de los gráficos en reporteria: no se diferencia qué gráfico es de qué proyecto

3	Alta	Sí	Sí	Permite ver de forma clara y centralizada el avance los diferentes proyectos, así como sacar información del rendimiento de los equipos	Mejorar diseño.	Diseño. Se podrían agregar más gráficos que permitan analizar otros aspectos de los proyectos
4	Alta	Sí	Sí	Resumen general de todos los tableros de la organización	UX	UX, mejora de filtros y diseño en general
5	Alta	Sí	Sí	Buena usabilidad y visibilidad de las tareas y horas de los proyectos	Algunos errores que impiden ver el total de horas en algunos reportes	Mejoras de Diseño
6	Alta	Sí	Sí	Poder tener seguimiento de cada proyecto, con sus tareas y indicadores de evolución.	Compatibilidad en los navegadores, colores al crear un nuevo requerimiento.	*Mejorar en el aspecto Mobile *Distribución de la información a nivel de estilo.
7	Alta	Sí	Sí	El gráfico de distribución de horas es bien útil. También la vista de tareas semanal.	Lo plano del diseño	Agregar color de fondo a las etiquetas, para distinguirlas rápidamente en el listado
8	Alta	Sí	Sí	La información se presenta de manera detallada y también agregada, permitiendo una mirada rápida de la situación de cada proyecto.	Algunas pantallas (reportes de un proyecto y detalle de un proyecto) presentan prácticamente la misma información, lo que produce una sensación de desorientación/error obligando a double-checkear.	No hay tooltips o etiquetas que ayuden a entender la navegación (o significado de cada cosa)

9	Alta	Sí	Sí	La rapidez de la aplicación, las visualizaciones, el concepto medido		No tiene adaptación a Firefox para ciertos errores gráficos. El gráfico de distribución de tareas

Anexo B - Diagramas de estructura de componentes

A continuación se muestra y explica brevemente la estructura de los componentes de la solución. En particular la Figura 27 muestra el contenedor de la lista de proyectos, el cual está compuesto por el componente de Header, un componente de tabla y un selector. El contenedor maneja las organizaciones obtenidas para el filtro (Select), un componente de tabla para listar los proyectos y el componente de Header de la vista.

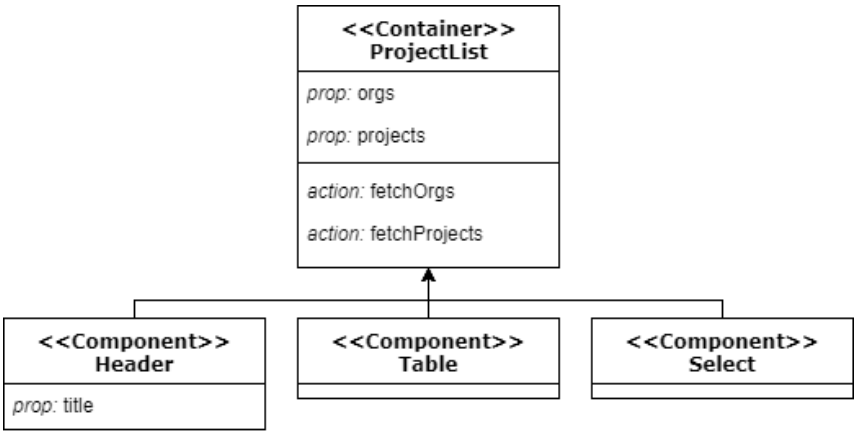

Figura 27: Contenedor de lista de proyectos

Luego está el contenedor de resumen, el cual particularmente utiliza el componente de resumen de tareas, como se puede apreciar en la Figura 28. El contenedor maneja las últimas tareas obtenidas para listarlas en el formato adecuado explicado en el documento, pasándoselas a su componente hijo: TaskSummary.

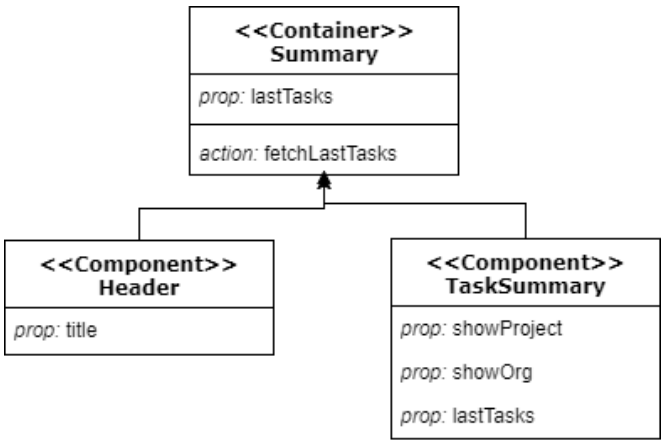

Figura 28: Contenedor de resumen

El componente de resumen de tareas (Figura 29) recibe las tareas de su padre y pasa cada tarea a un componente llamado TaskSummaryItem que le da el formato especial a cada fila de la tabla. Además utiliza un componente de tabla para generar la tabla que contiene el resumen.

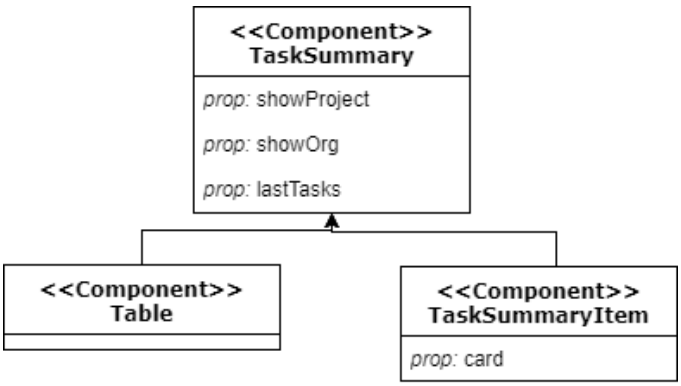

Figura 29: Componente de resumen de tareas

Por otra parte se tiene el contenedor de detalle de proyecto, el cual maneja la información del proyecto en particular, las etiquetas del tablero de Trello al que pertenece, las últimas tareas del proyecto y otros parámetros. En la Figura 30 se puede ver que el contenedor tiene muchos componentes hijo, dentro de los cuales hay dos importantes: ProjectSummary, que genera un gráfico de la distribución de tareas del proyecto y RequirementModal, el cual genera el modal necesario para el registro de nuevos requerimientos en el proyecto.

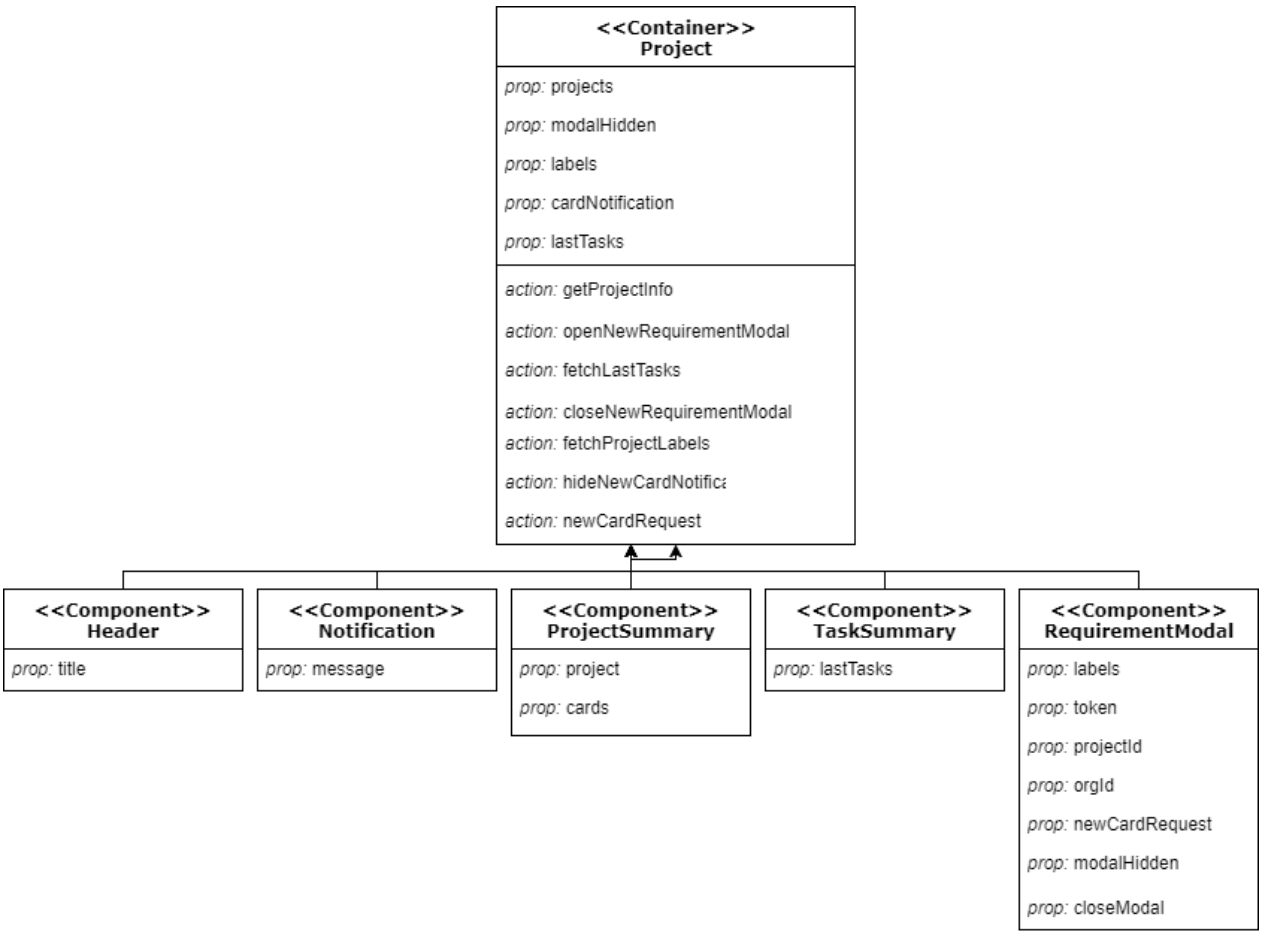

Figura 30: Contenedor detalle de proyecto

Reports es el contenedor que maneja la información de los proyectos para generar la vista y los documentos de reportes. Necesita las organizaciones para realizar un filtro y los proyectos para pasárselos a ReportTable que genera la vista de reporte de los proyectos seleccionados y a ProjectSummaryList, que genera un gráfico para cada proyecto utilizando ProjectSummary, como se aprecia en la Figura 31.

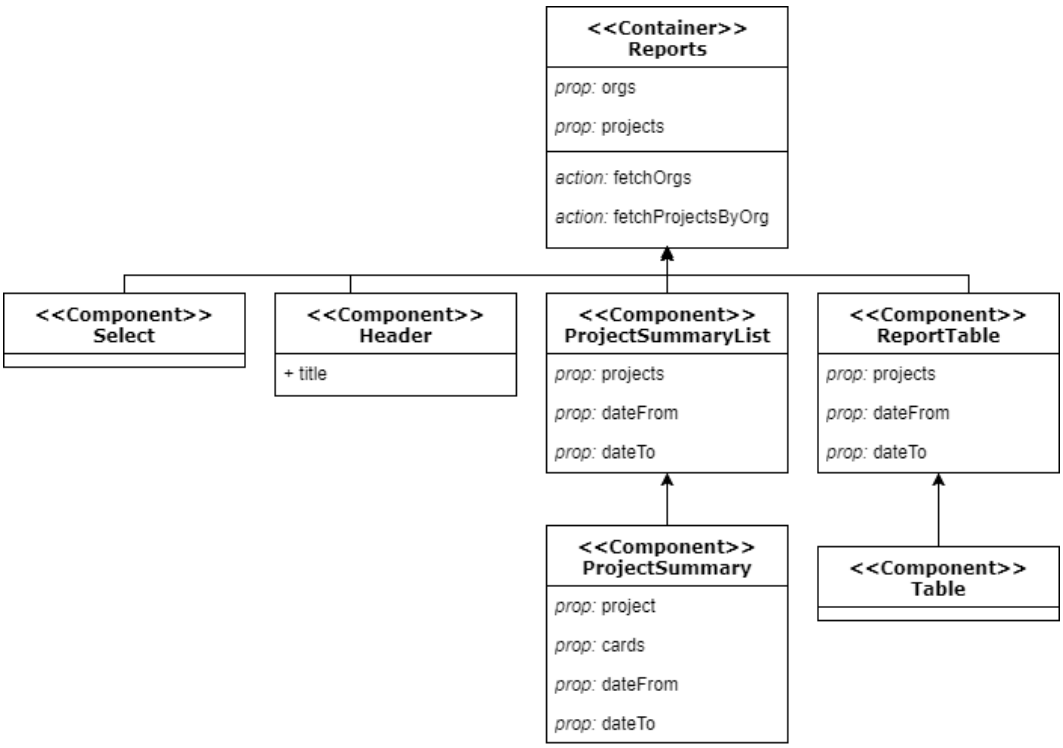

Figura 31: Contenedor de reportes

El contenedor de Dashboard maneja las organizaciones para poder filtrar, el listado de proyectos de la organización seleccionada y la distribución horaria de los proyectos en el periodo anual. Utilizando un componente de pestañas (Tabs), separa la vista en el Dashboard de horas y de tareas, de los cuales son responsables HoursDashboard y TasksDashboard (ver Figura 32). Estos componentes tienen un filtro por organización y se encargan de generar los gráficos explicados y mostrados en el documento.

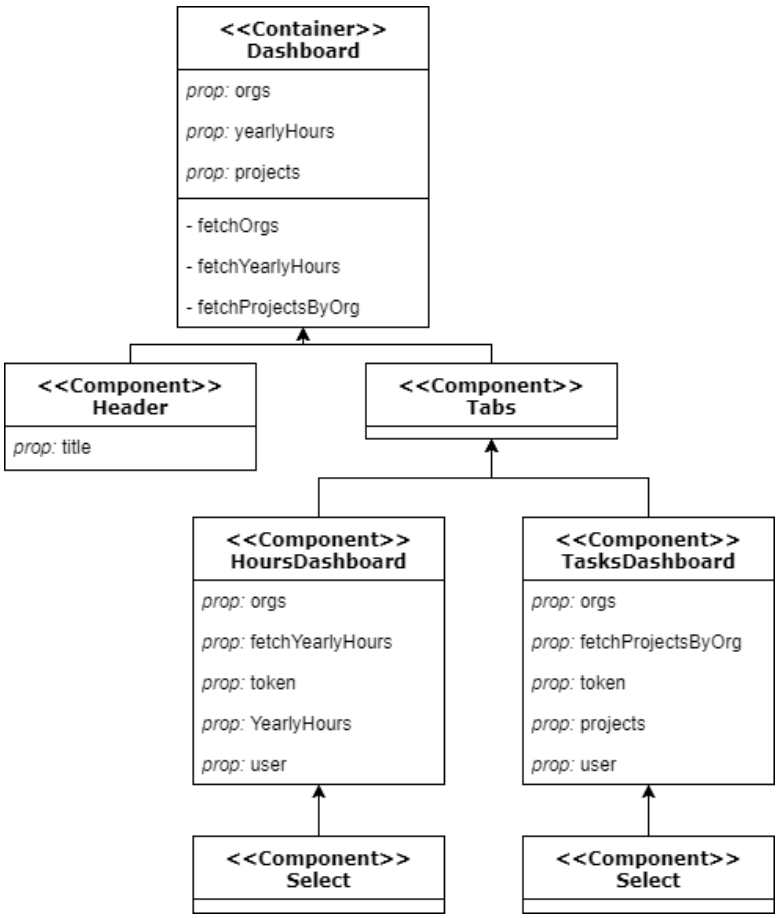

Figura 32: Contenedor de dashboard

Finalmente tenemos el contenedor del administrador, el cual maneja toda la información necesaria para la creación de organizaciones, proyectos y usuarios en el sistema. Utilizando pestañas, separa la vista en tres: OrgAdmin, ProjectAdmin y UserAdmin, los cuales mediante una tabla presentan la información actualmente disponible en el sistema. Cada uno de los componentes principales del administrador cuenta con un componente que genera su modal respectivo para mostrar el formulario necesario para crear o editar la información que le corresponde, como se puede apreciar en la Figura 33.

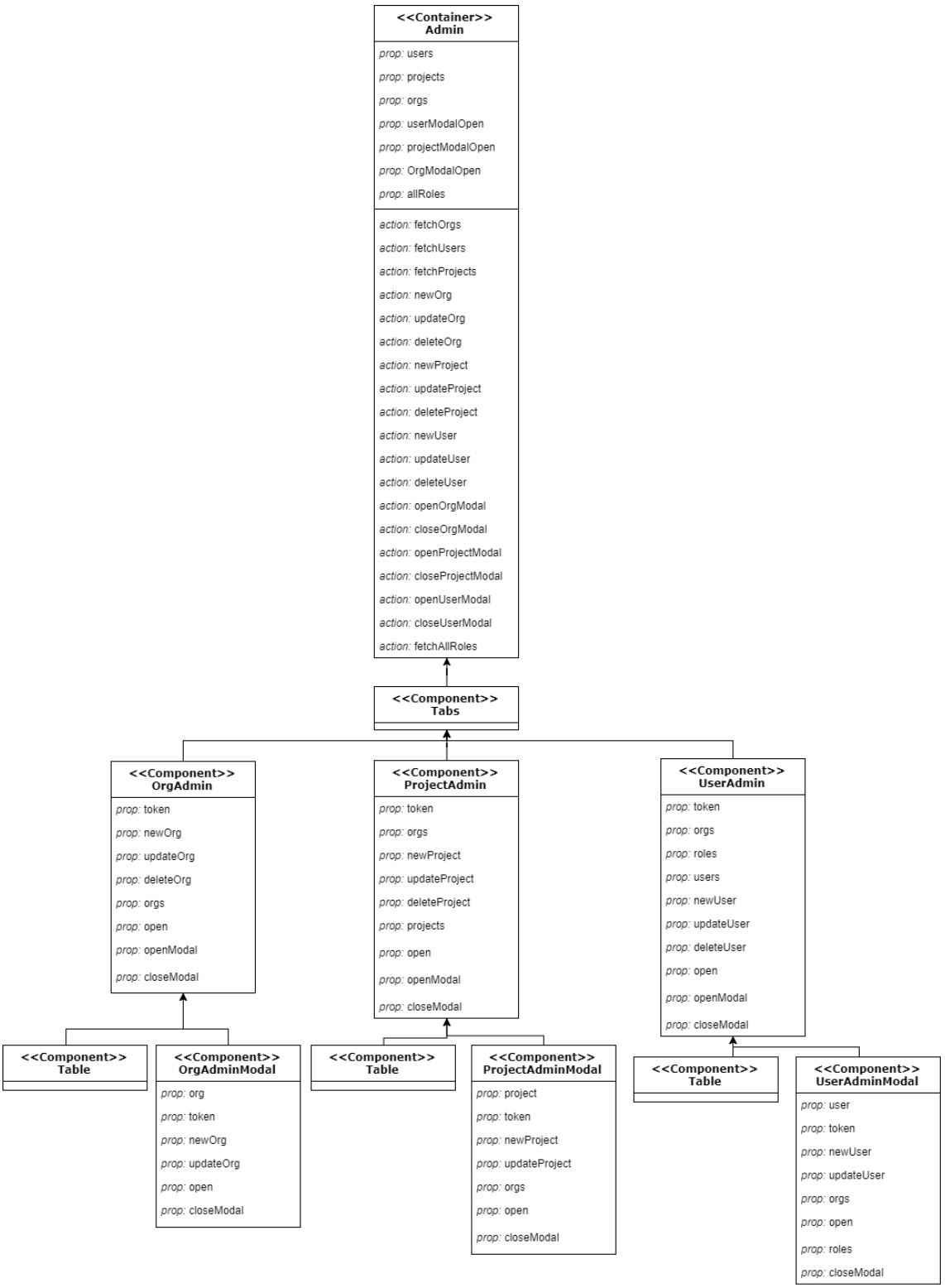

Figura 33: Contenedor de administrador