UNIVERSIDAD NACIONAL DE LA PLATA

FACULTAD DE INFORMÁTICA

TESINA DE LICENCIATURA

Programa de Apoyo al Egreso para Alumnos con Práctica Profesional Supervisada

TÍTULO: Reingeniería de la Interfaz de usuario de Omnially: mejorando la experiencia de usuario y el acceso a la aplicación

AUTOR/A: Villanueva Jimena Magali

DIRECTOR/A ACADÉMICO: Harari Ivana

DIRECTOR/A PROFESIONAL: Martinez Fernando

CODIRECTOR/A ACADÉMICO:

CARRERA: Licenciatura en Informática

RESUMEN

Omnially es una aplicación web que permite a los usuarios crear y diseñar lienzos dinámicos, con el fin de proyectarlos en dispositivos físicos, como carteles digitales. En este trabajo, se llevó a cabo un rediseño integral del front-end de la aplicación, cuyo objetivo principal fue mejorar la interfaz, centrándose en las necesidades del usuario, logrando así una notable mejora en su experiencia. El proyecto abarcó desde la reorganización de las funciones clave para mejorar su comprensión y utilización hasta la centralización de integraciones dispersas, lo que facilitó el acceso y la gestión de las mismas. Para lograr esto, se llevaron a cabo estudios de usabilidad aplicando métodos y técnicas de diseño centrado en el usuario, con el fin de validar las decisiones y mejoras efectuadas. Asimismo, se implementaron nuevas vistas más intuitivas, optimizando la usabilidad, y se planificaron futuras mejoras para continuar elevando la calidad del servicio que ofrece Omnially a sus usuarios.

Palabras Claves

Reingeniería de Software, experiencia de usuario, aplicación web, gestor de lienzos dinámicos, aplicación cliente servidor, metodologías ágiles, desarrollo de Front-end.

Trabajos Realizados

Se realizó un rediseño integral de la aplicación web Omnially, centrado en mejorar la experiencia del usuario y optimizar la interfaz. Se abordaron diversas áreas claves, incluyendo la reorganización de funciones, la centralización de integraciones y la implementación de nuevas vistas. Las tareas se dividieron en fases para facilitar su gestión, comenzando por las más básicas y avanzando hacia las más complejas. Durante el proceso, se integraron nuevas herramientas y se superaron los desafíos que surgieron, al mismo tiempo que se llevaron a cabo estudios de usabilidad aplicando métodos y técnicas de diseño centrado en el usuario para validar las decisiones y mejoras efectuadas, asegurando así una mejora continua de la plataforma.

Conclusiones

El rediseño de la aplicación Omnially ha sido un proyecto desafiante que ha requerido abordar múltiples obstáculos. La transformación integral del front-end, tanto de sus vistas como de sus componentes, centrada en optimizar la experiencia del usuario, ha mejorado notablemente la usabilidad de la plataforma logrando una interfaz más intuitiva y funcional. Este proyecto no solo ha alcanzado con los objetivos propuestos, sino que también ha establecido las bases para futuras mejoras y actualizaciones, convirtiendo a Omnially como una herramienta más accesible y fácil de usar.

Trabajos Futuros

El trabajo realizado sirvió de base para futuros desarrollos dentro de la aplicación Omnially, estableciendo un ejemplo a seguir para el resto de las funciones y aplicaciones de la empresa. Se identificaron áreas clave donde el rediseño y la reingeniería pueden extenderse, incluyendo vistas y componentes como la barra de navegación principal y las vistas generales del sistema. Estas mejoras, junto con el enfoque aplicado, ofrecen una base sólida para futuros cambios y una guía para optimizar la cohesión visual y mejorar la experiencia del usuario.

Fecha de la presentación: NOVIEMBRE 2024

Facultad de Informática | UNIVERSIDAD NACIONAL DE LA PLATA

Índice General

Índice de Figuras

ANEXO 4 – Formato tesina PAE-PPS

1. Exposición de lo realizado en la PPS

Durante la Práctica Profesional Supervisada, se llevó a cabo un proceso integral de desarrollo de la aplicación web “Omnially”, con un enfoque en el rediseño del *frontend*. Este proceso se planificó en varias fases para mejorar la experiencia del usuario.
Omnially es una aplicación de gestión de lienzos que permite a los usuarios crear “*canvas*” o lienzos dinámicos en blanco y diseñar carteles personalizados con distintos *widgets*. Estos lienzos o “*canvas*” son publicados como carteles digitales, en diversos dispositivos físicos.
Debido a la alta cantidad de usuarios diarios, se recopilaron comentarios y *feedbacks* que revelaron la necesidad de un rediseño enfocado en mejorar la experiencia del usuario. A través de cuestionarios y entrevistas realizadas durante el proceso, se recolectaron quejas y observaciones de los usuarios del sistema, los cuales revelaron que la interfaz actual presenta dificultades de comprensión y uso. Además, existen problemas de encontrabilidad (*‘findability’*), lo que obliga a los usuarios a aprender constantemente cómo utilizar el sistema, impactando negativamente en la eficiencia del usuario y en la productividad general del sistema.
La planificación del rediseño, basada en etapas metodológicas y en un enfoque centrado en el usuario, busca abordar estos problemas al mejorar la encontrabilidad y la centralización de funciones. En base a los resultados obtenidos a partir de los cuestionarios y analizando las dificultades de uso que los usuarios experimentan diariamente, concluimos que el objetivo, entonces, es reducir los costos asociados al uso del sistema y hacer que la aplicación sea más fácil de usar y entender para todos los usuarios.
Omnially cuenta con seis integraciones distribuidas en la aplicación, lo que dificulta su gestión para el usuario. Este rediseño tiene como objetivo centralizar su acceso para mejorar la usabilidad.
La dispersión de estas integraciones no solo complica su gestión, sino que también dificulta la navegación y el aprovechamiento pleno de sus funcionalidades por parte de los usuarios.
Actualmente, las integraciones se encuentran distribuidas en tres sectores diferentes dentro de la aplicación: en la creación de una compañía dentro de un *canvas*, en la configuración de un *widget* de tipo *‘donorlist’*, y en la configuración de un *widget* de tipo ‘clima’. Esta organización fragmentada genera confusión, ya que los usuarios deben recordar en qué parte del sistema se encuentra cada integración. Esto no solo aumenta la carga cognitiva y el tiempo necesario para completar tareas, sino que también impide una gestión eficiente de los datos, lo que afecta negativamente la experiencia general de uso.

A continuación, se detallan las integraciones y sus usos específicos:

- **DonorPerfect y Raiser:** Estas integraciones son esenciales para las organizaciones que utilizan Omnially para gestionar campañas de donaciones. Proveen información detallada sobre las listas de donantes, permitiendo a las compañías mantener un registro actualizado y preciso de sus benefactores.
- **Vengo, Adomni y Adstash**: Estas tres integraciones son esenciales para la gestión y optimización de campañas publicitarias digitales. Facilitan la creación, despliegue y análisis de anuncios en diversas plataformas, permitiendo a las empresas maximizar el alcance y la efectividad de sus campañas publicitarias.
- **OpenWeatherMaps:** Esta integración proporciona información meteorológica en tiempo real, que es utilizada en los lienzos para mostrar el clima actual. Es especialmente útil para empresas que necesitan mostrar información climática relevante a sus clientes o empleados.

En resumen, la ubicación dispersa de las integraciones en la aplicación afecta negativamente la experiencia del usuario. Es por esto que centralizarlas en un solo lugar no solo mejorará la usabilidad, sino que también facilitará el acceso a sus funciones, optimizando el flujo de trabajo y aumentando la productividad general.

Luego de varias reuniones y análisis con el cliente (CEO de Omnially), se decidió centralizar la gestión de todas las integraciones en una nueva sección del menú principal llamada "*Manage -> Integrations*". Esta vista agrupa todas las integraciones en formato de tablero, donde los usuarios con rol de “Administrador” o “Super Administrador” pueden habilitar o deshabilitar integraciones de manera sencilla.
El proyecto se planificó en varias etapas, abordando el diseño de la UI, la configuración de cada integración y la adaptación de las funcionalidades existentes a la nueva estructura.

Etapa 1: *Integrations Page UI*
Etapa 2: *Integrations Settings Page UI Template*
Etapa 3: *Convert Blackbaud Raiser’s Edge NXT*
Etapa 4: *Convert Vengo*
Etapa 5: *Convert DonorPerfect*
Etapa 6: *Convert AdStash*
Etapa 7: *Convert AdOmni*
Etapa 8: *Convert OpenWeatherMaps*

El proceso de rediseño y centralización de integraciones estuvo bajo mi responsabilidad, donde las ocho etapas mencionadas conformaron un proyecto de reingeniería denominado '*Manage Integrations*', el cual reunió las tareas necesarias para llevar a cabo el objetivo principal. La idea era crear una nueva vista y sección en el menú principal, llamada ‘Integraciones’, donde se reúnen y muestran todas las integraciones del sistema con una apariencia de estilo *‘marketplace’*. Es decir, listando las integraciones del sistema en forma de tarjetas individuales que incluyen información clave, como el nombre, el proveedor y una breve descripción de su funcionalidad. De esta manera, los usuarios pueden explorar fácilmente las opciones disponibles y seleccionar las integraciones que mejor se adapten a sus necesidades, mejorando la usabilidad y accesibilidad del sistema. Una vez dentro de cada integración de la lista, se mostrará una imagen y una descripción de la integración, agregando una forma fácil para que puedan habilitar la integración para su cuenta.

El proceso de rediseño incluyó varias etapas clave que garantizaron la centralización de las integraciones y la mejora en la experiencia del usuario. En primer lugar, se propuso la creación de una nueva sección en el menú principal que reuniera todas las integraciones del sistema, permitiendo gestionar desde una sola vista todas las activaciones y configuraciones, lo que eliminó la necesidad de recordar su ubicación dispersa dentro del sistema. Este cambio mejoró la navegabilidad, facilitando la comprensión del sistema y reduciendo el tiempo y esfuerzo necesarios para completar tareas. La nueva interfaz centralizada incluyó mejoras en la organización de los menús y retroalimentación visual clara para guiar a los usuarios durante el proceso de configuración.

Durante este proceso de reingeniería, se implementaron diversas técnicas de diseño centrado en el usuario y metodologías colaborativas[1] para asegurar que las mejoras fueran efectivas y ajustadas a las necesidades del proyecto.

En la etapa de indagación, cuya organización y desarrollo también estuvo bajo mi responsabilidad, se aplicaron diversas técnicas de diseño centrado en el usuario para el relevamiento de necesidades y problemáticas[2]. Se realizó un análisis exhaustivo y de investigación del usuario a través de la recolección de quejas y opiniones, como entrevistas y encuestas, lo que me permitió identificar áreas clave a mejorar. Además, se llevó a cabo observación directa para evaluar la aceptación de los cambios propuestos. Para profundizar en la experiencia del usuario, apliqué la técnica de ‘recorrido cognitivo’ (*cognitive walkthrough*)[3], lo cual facilitó identificar problemas tanto desde mi propia experiencia como desde la perspectiva del desarrollador y la de usuario del sistema.
Para asegurar que el desarrollo cumpliera con las expectativas del cliente, se realizaron reuniones semanales donde se revisaron las etapas finalizadas y se planificaron las tareas siguientes. Además, dentro de ellas se implementó una etapa de validación que incluyó *focus groups* y la mirada de expertos, para garantizar que las mejoras propuestas fueran efectivas y respondieran a las

necesidades reales del proyecto. Estas reuniones fueron esenciales para revisar los cambios y ajustar el curso según las necesidades del proyecto.

Se utilizaron herramientas como Jira para la planificación, Slack para la comunicación diaria, y Figma para el prototipado visual, facilitando una colaboración constante con el cliente. Estas herramientas fueron propuestas por la empresa junto con el cliente, y estuvo a mi cargo el estudio, uso y la formación del resto del equipo en su implementación, asegurando una adopción efectiva y un flujo de trabajo optimizado.
Este enfoque integral permitió detectar errores del sistema, confirmar observaciones y evaluar los costos de ejecución de las actividades de mejora.

Etapa 1: Integrations Page UI
El objetivo fue crear la vista central de todas las integraciones (solo parte visual). Para esto se diseñó un tablero con tarjetas que muestran el nombre y el logo de cada integración, junto con una etiqueta que indica si la integración está activa. El diseño se desarrolló a través de *mockups*, teniendo en cuenta la escalabilidad futura para añadir más integraciones sin necesidad de rediseñar la interfaz.

A continuación, en la Figura 1 se muestra el prototipo del nuevo diseño esperado:

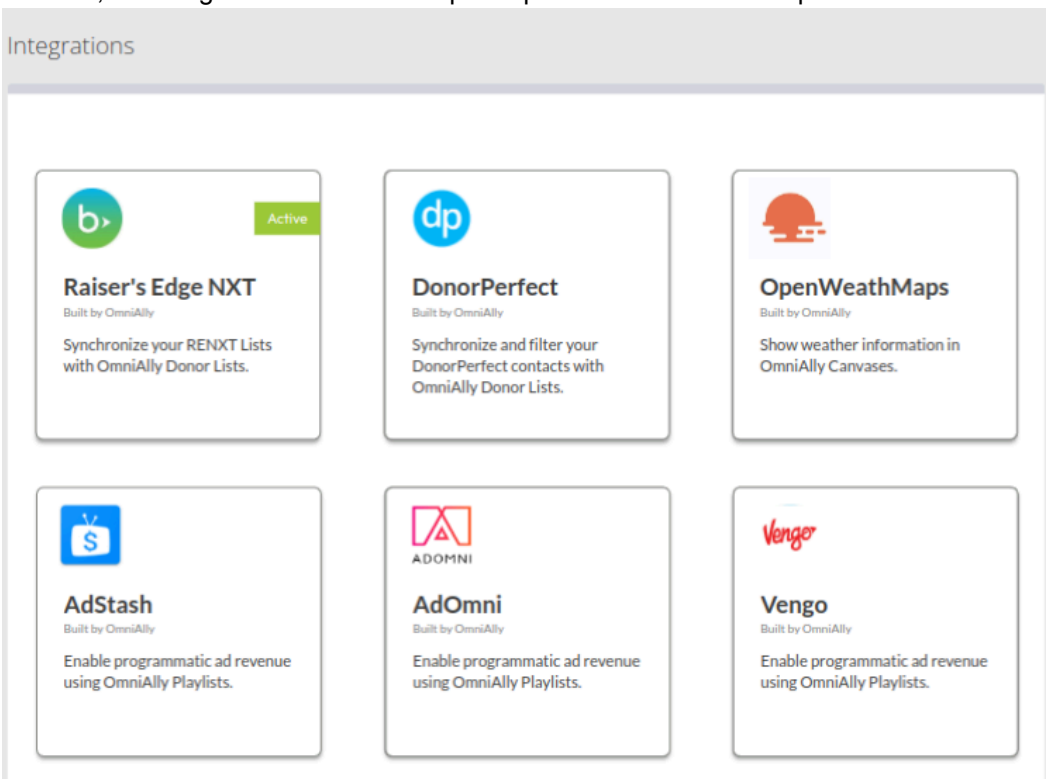

Figura 1. Prototipo del nuevo diseño esperado de la vista central de integraciones.

Etapa 2: Integration Settings Page UI Template
El objetivo fue diseñar la vista individual de configuración para cada integración. Para esto se desarrollaron *mockups* para representar tres nuevos estados posibles de una integración: activa, inactiva y próxima a caducar. La interfaz incluyó un encabezado con el nombre y el logo de la integración, información general y un panel de configuración.
Se consideraron los tres escenarios posibles y se diseñaron vistas específicas para cada uno de ellos.

- *Integración activa:* esto sucede cuando la integración fue activada por un usuario de la empresa. En este estado se muestra información importante sobre quien fue el que la activó, fecha de expiración del *link* (si corresponde) y un botón para desactivarla.

A continuación en la Figura 2, vemos el prototipo del nuevo diseño esperado para una integración en estado 'Activa'.

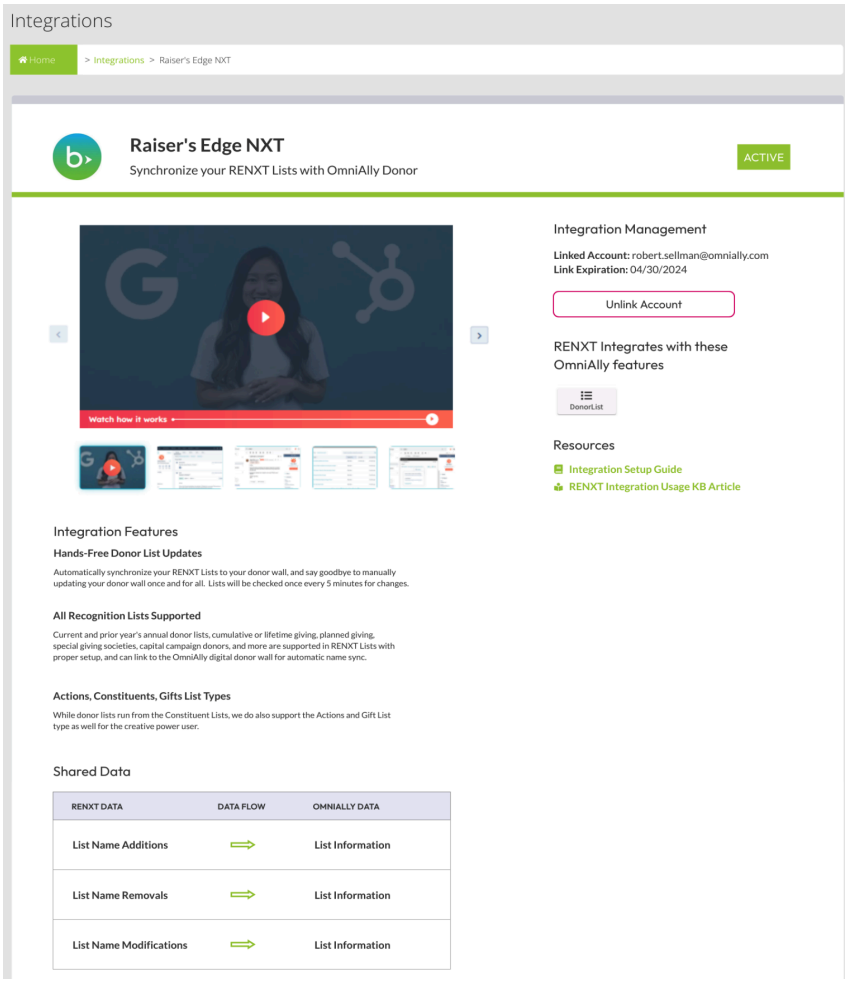

Figura 2. Prototipo del nuevo diseño de la vista general de una integración con estado 'Activa'.

- *Integración inactiva:* esto sucede cuando la integración nunca fue activada, o fue activada pero se desactivó, o en el caso de que el certificado OAuth se venció y no fue renovado. Para este estado se muestra un botón para activarla. Al presionarlo se abre una ventana con los campos necesarios para poder activarla.

A continuación en la Figura 3, observamos el prototipo del nuevo diseño esperado para una integración en estado ‘Inactiva’.

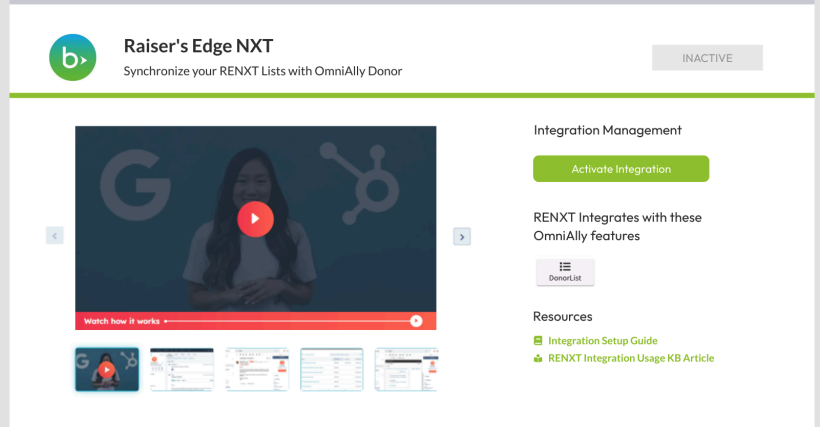

Figura 3. Prototipo del nuevo diseño de la vista general de una integración con estado ‘Inactiva’.

- *Integración en estado renew:* esto sucede cuando el certificado OAuth de la integración se caduca en 15 días. Para este estado se muestra información importante al igual que en el estado activa, junto a un cartel informando que está pronto a caducar el *link*. Además un botón para renovarla con la misma funcionalidad que el botón para activarla.

A continuación en la Figura 4, se muestra el prototipo del nuevo diseño esperado para una integración en estado ‘Renew’.

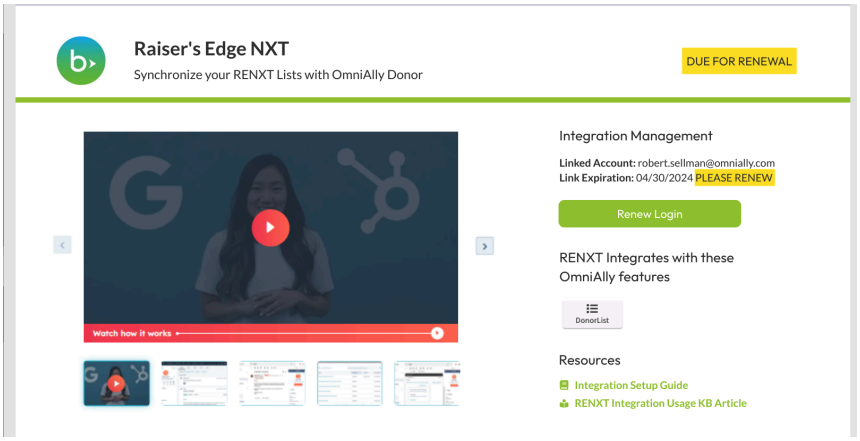

Figura 4. Prototipo del nuevo diseño de la vista general de una integración con estado ‘Renew’.

Etapa 3: Convert Blackbaud Raiser's Edge NXT

El objetivo fue adaptar la integración de Blackbaud existente dentro de los *canvas* para el nuevo sistema.

Para ello se modificó el comportamiento de la integración verificando el estado de la misma, estando activa o próxima a vencer antes de mostrar la información en el *widget* donde se utiliza. Además, a pedido del cliente, se añadió un sistema de notificaciones por correo electrónico para los administradores, avisando sobre el vencimiento de la autenticación 15, 5 y 1 día antes.

Etapa 4: Convert Vengo, Phase 5: Convert AdStash y Phase 6: Convert AdOmni
El objetivo fue adaptar las integraciones de tipo 'vast' (Vengo, Adomni y Adstash) para el nuevo sistema de gestión centralizada.
Para ello se diseñaron y desarrollaron las vistas correspondientes para cada integración, eliminando las formas antiguas de activación/desactivación y centralizando la configuración.
Previo al rediseño, estas integraciones eran configuradas mediante un checkbox ubicado dentro de la configuración de la compañía para la que se quería activar, donde al seleccionarlo se desplegaban distintos campos de tipo *inputs* para ingresar la información correspondiente de cada una.

Etapa 7: Convert DonorPerfect
El objetivo fue adaptar la integración existente dentro de los *canvas* para el nuevo sistema.
Para ello se mantuvo la consistencia de los componentes, asegurando que los *canvas* con *widgets* de tipo donor list con origen 'DonorPerfect' continúen funcionando correctamente. Se añadió la verificación del estado de la integración antes de permitir su uso.

Etapa 8: Convert OpenWeatherMaps
El objetivo fue adaptar la integración del *widget* de clima para el nuevo sistema.
Para ello se migró de AccuWeather a OpenWeatherMaps debido a restricciones de funcionalidad.
Esto implicó adaptaciones y negociaciones con el cliente para cumplir con los requisitos deseados.

2. Informe técnico

2.1. Resumen

A mediados del año 2023, se planteó la idea de llevar a cabo un proceso de reingeniería del software en pos de lograr centralizar y unificar un ambiente de trabajo muy importante para el sistema: las integraciones. Comenzó con mejoras visuales pretendiendo una mejor experiencia para el usuario, y luego fue derivando a cambios en permisos y funcionalidades ya implementadas, como así también cambios en las integraciones utilizadas.

El rediseño tuvo como objetivo centralizar las integraciones y mejorar la experiencia del usuario, optimizando la interfaz y el rendimiento de la aplicación. Encargándome desde la planificación hasta la implementación, asegurando que cada aspecto del proyecto cumpliera con los objetivos de usabilidad y funcionalidad establecidos.

Este proyecto de centralización y reingeniería se compuso de varias etapas, en orden según la prioridad estimada para cada una. Comenzando con tareas vinculadas directamente a crear una nueva vista más óptima para el usuario, y continuando con tareas relacionadas al funcionamiento y especificación de cada integración involucrada.

Durante cada semana, se llevaron a cabo bajo mi responsabilidad una serie de tareas estipuladas y asignadas, siguiendo las indicaciones del cliente y abordando los errores y correcciones que surgen reportados por el mismo. Debido a su menor complejidad, lo primero en ser implementado fue la nueva sección en el menú principal llamada 'Integrations' junto a su vista, reuniendo todas las integraciones del sistema. Esto permitió establecer pautas de diseño y desarrollo con el cliente, y sentar las bases para abordar etapas posteriores más complejas en el proyecto.
Luego, se optó por continuar con el diseño de la vista general de las integraciones, es decir la vista correspondiente al clickear sobre una integración en particular. Para esto se agregaron estados a las integraciones para indicar si se encontraba activa, inactiva o próxima a caducar. Esta etapa demandó un esfuerzo adicional ya que se tuvo que incluir el concepto de 'Integración' como una nueva tabla en la base de datos, junto a sus particularidades, para luego más adelante eliminar toda referencia a estas en otras tablas existentes en el sistema.

La siguiente etapa fue desarrollar los cambios necesarios en cada integración en particular, por lo que se desglosó en seis etapas más (conforme a la cantidad de integraciones existentes en el sistema). Esto introdujo desafíos más complejos ya que debía implementarse cambios en funcionalidades ya implementadas hace tiempo, sin descuidar lo realizado y dejando el sistema consistente en todo momento.
En esta etapa se diseñó la vista de cada integración teniendo en cuenta los campos necesarios para la activación de cada una de ellas en el formulario, como así también se eliminaron los viejos accesos de la activación previa existente.

El rediseño de Omnially representó un proyecto compuesto por desafíos diversos y constantes. En base al proceso de testing de las mejoras realizadas y la recolección de *feedback*, se puede concluir que este trabajo no solo cumple con las expectativas, sino que ha mejorado la experiencia del usuario y establece una base sólida para futuras innovaciones en otras funcionalidades de la plataforma.

2.2. Palabras Clave

Reingeniería de *Software*, experiencia de usuario, aplicación web, gestor de lienzos dinámicos, aplicación cliente servidor, metodologías ágiles, desarrollo de *Front-end*.

2.3. Marco Conceptual

Omnially es una startup de Portland, Oregón, que ofrece soluciones de señalización digital, incluyendo software y hardware, para entornos como hospitales, aeropuertos y hoteles en EE.UU. En colaboración con Sinaptia, han desarrollado una plataforma de gestión de medios y un dispositivo de señalización digital utilizando tecnologías como Ruby on Rails, PostgreSQL y React. Esta plataforma permite la gestión y actualización remota de contenido en sus dispositivos, ofreciendo comunicación bidireccional y soporte integral a sus clientes.

La aplicación web de Omnially ofrece a sus usuarios la capacidad de crear y personalizar lienzos dinámicos llamados *'canvas'* mediante diversos *widgets*. El propósito principal es proyectar estos lienzos como carteles digitales en dispositivos físicos, como pantallas. Esta funcionalidad está disponible para empresas que han contratado el servicio ofrecido por Omnially, siendo los usuarios los administradores de dichas organizaciones. Dentro de la plataforma, cada organización puede crear usuarios y asignarles distintos roles (administradores, súper administradores, usuarios regulares o usuarios espectadores) para restringir las acciones que pueden realizar. Los usuarios de una misma compañía pueden colaborar en la creación y modificación de los carteles, lo que agrega un elemento de interactividad a la aplicación.

En la actualidad, la aplicación cuenta con 250 usuarios pertenecientes a 100 compañías diferentes, incluyendo hospitales, cafeterías, fundaciones, hoteles, entre otros.

Uno de los aspectos más destacados de Omnially es su sistema de reconocimiento de donantes[4], denominado "*donorlist*" o "muro de donantes" (*donor wall*). Estos muros digitales son herramientas utilizadas por las organizaciones sin fines de lucro para reconocer públicamente a las personas y entidades que han contribuido económicamente a su causa. Tradicionalmente, el reconocimiento de donantes se hacía mediante paneles de vidrio o placas de metal, pero estos métodos están siendo reemplazados por muros digitales debido a sus numerosas ventajas.

Omnially se destaca por ofrecer soluciones innovadoras en el ámbito de la señalización digital y el reconocimiento de donantes. Su capacidad para integrar tecnologías avanzadas y proporcionar una plataforma flexible y personalizable para sus usuarios la posiciona como una herramienta valiosa

para diversas organizaciones que buscan mejorar la comunicación visual y el reconocimiento de sus contribuyentes.

Omnially ofrece una variedad de características avanzadas para sus sistemas digitales de reconocimiento de donantes, como:

- **Listas de Donantes actualizables:** Permiten modificar fácilmente los nombres y detalles de los donantes.
- **Soporte para texto e imágenes:** Integración con software como DonorPerfect y Raiser's Edge NXT.
- **Gestión remota:** Capacidad de gestionar el sistema desde cualquier lugar a través de un panel en la nube.
- **Operatividad sin conexión:** Funcionan incluso sin conexión a internet.
- **Diseños personalizados:** Ofrecen servicios de diseño personalizado que se adaptan a la marca y a necesidades específicas de cada organización.
- **Funciones interactivas:** Incluyen herramientas interactivas que facilitan la participación del público y la actualización del contenido.
- **Gestión y actualización fácil:** Herramientas que facilitan la actualización y gestión del compromiso con los donantes desde cualquier lugar a través de un panel en la nube.
- **Contenido multimedia:** Permiten la inclusión de videos, imágenes y otros contenidos interactivos, mejorando la presentación y el atractivo del muro de donantes.

La creación de un *canvas* consiste en incluir distintos tipos de *widgets* como de texto, imágenes, de clima, de *donor lists*, de hora, *pop ups*, de búsqueda, o *playlists*, entre otros.

Un *widget* de tipo '*donor list*' o lista de donantes, tiene distintos tipos de orígenes para acceder a la información de los donantes:
- a través de una integración llamada Raiser's Edge NXT[5],
- a través de una tabla cargada en el sistema,
- desde otra integración llamada Donor Perfect[6],
- o a través de un google *Spreadsheet* (hoja de cálculo de google).

Así mismo, sucede algo similar con el *widget* de clima. Utiliza una integración relacionada al clima para acceder a la información externa (OpenWeatherMaps[7]).

Además existen otras tres integraciones en el sistema utilizadas para anuncios y publicidad de estos carteles digitales presentados en dispositivos físicos, estas son AdStash[8], AdOmni[9] y Vengo[10], lo cual hace que el sistema cuente en total con seis integraciones distribuidas con distintas funcionalidades.

Uno de los problemas detectados fue que al momento de activar o desactivar una integración dentro de la aplicación, los usuarios debían seguir una serie de pasos complejos y no intuitivos, lo que generaba frustración, abandono de tareas, y un tiempo excesivo dedicado al aprendizaje de su uso.

Este problema se manifestaba claramente en el proceso de activación y desactivación de integraciones. Los usuarios debían seguir los siguientes pasos para activar una integración, lo que demostraba lo engorroso del proceso:
1. Navegar por la barra de navegación principal.
2. Ir a la sección de "compañías".
3. Configurar una empresa.
4. Activar un checkbox de la integración deseada y proporcionar los datos correspondientes.

A continuación, en la Figura 5, una imagen del proceso de activación del diseño anterior:

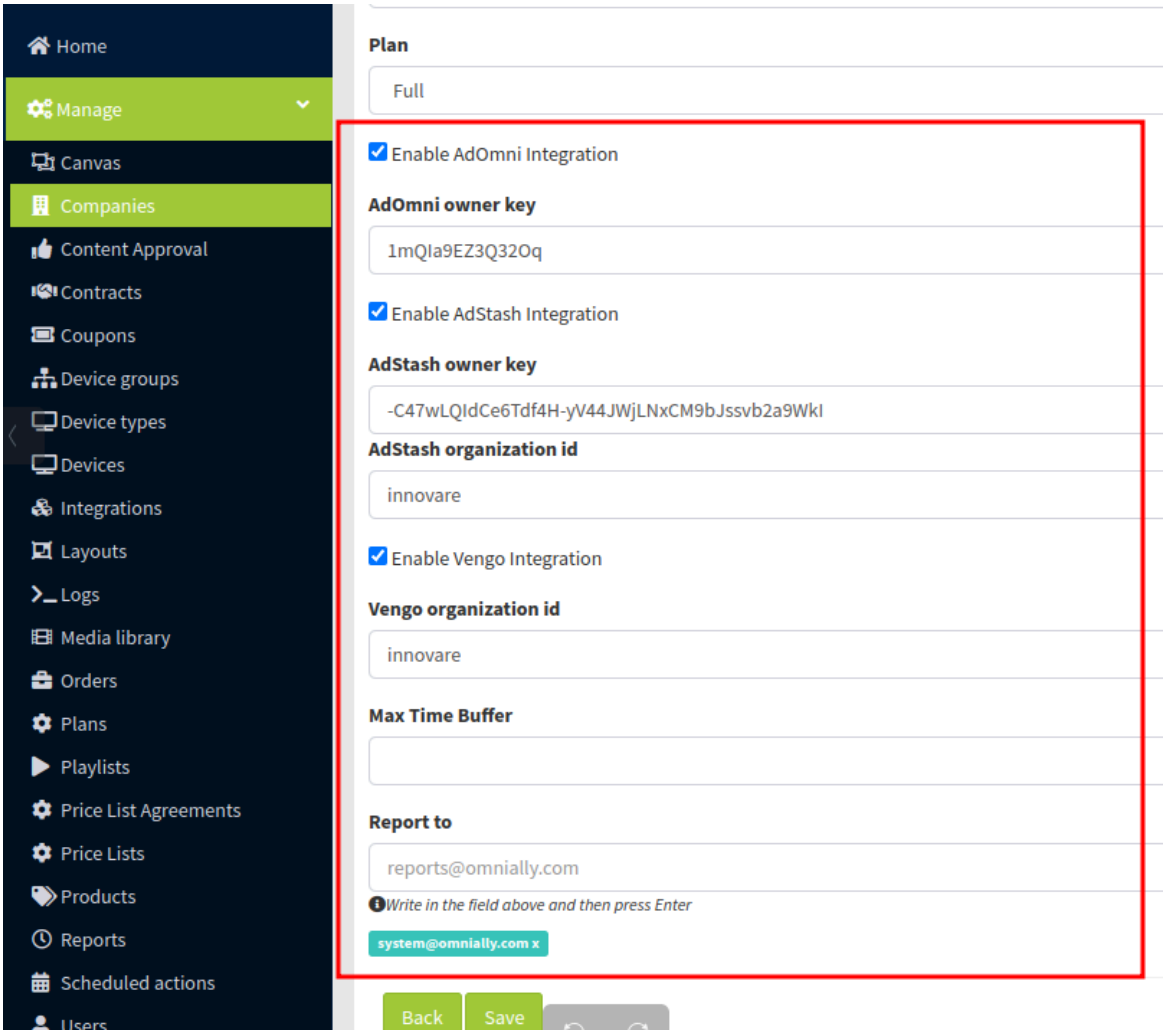

Figura 5. Diseño anterior de los pasos para activar una integración dentro de una compañía.

Estos pasos eran necesarios sólo para las integraciones de tipo vast pero además, existían otros tipos de integraciones en el sistema que también requerían activación por parte del usuario, y se encontraban ubicadas en diferentes secciones y requieren otra serie de pasos y datos diferentes.

A continuación se detallan los pasos que eran necesarios para integrar las demás integraciones del sistema:

- Para activar integración Blackbaud y DonorPerfect:
    1. Navegar por la barra de navegación principal.
    2. Ir a la sección de "*Canvas*" (crear uno nuevo o usar uno existente).
    3. Crear un *widget* de tipo "*Donorlist*".
    4. Seleccionar el origen de datos:
        a. Raiser: Se abrirá una ventana emergente para iniciar sesión.
        b. DonorPerfect: Se desplegará un campo de tipo *input* para ingresar la *API key*.

- Para activar integración OpenWeatherMap:
    1. Navegar por la barra de navegación principal.

2. Ir a la sección de “*Canvas*” (crear uno nuevo o usar uno existente).
3. Crear un *widget* de tipo “Clima”.
4. Ingresar la *API key*.

Este proceso resultaba problemático tanto para los usuarios como para el equipo de desarrollo, ya que recordar los pasos necesarios y la ubicación de cada integración era complicado. A menudo, los usuarios debían consultar documentación o pedir ayuda a colegas, lo que ralentizaba el proceso y aumentaba la insatisfacción.

En conclusión, dado que la interfaz de usuario no facilitaba el proceso, y en base a un relevamiento realizado mediante opiniones de los usuarios, surgió la necesidad de rediseñar visualmente la aplicación utilizando métodos de diseño centrado en el usuario.
Para abordar estos problemas, se implementó un rediseño del sistema que centralizó todas las integraciones, permitiendo su gestión desde un único lugar. El rediseño se centró en mejorar la usabilidad mediante una interfaz unificada que simplificó la activación y configuración de las integraciones. Además, se mejoraron la retroalimentación visual y la navegación, facilitando la comprensión y el uso del sistema, lo que redujo significativamente el tiempo necesario para realizar estas tareas y aumentó la productividad de los usuarios.

Los aspectos clave del rediseño incluyen:

- Una interfaz centralizada y simplificada para gestionar todas las integraciones.
- La capacidad de activar y desactivar integraciones con los datos necesarios desde una sola vista y ubicación.
- Mejoras en la navegabilidad[11] y la organización de los menús.
- Retroalimentación visual clara para guiar al usuario a través del proceso.

Como se detalla en la sección correspondiente al trabajo realizado, este rediseño no solo optimizó el flujo de trabajo y redujo el tiempo y esfuerzo requeridos por los usuarios, sino que también mejoró significativamente la interacción con el sistema, contribuyendo a la finalización de tareas que previamente eran complicadas o requerían mucho tiempo.

Como parte del equipo de desarrollo *full stack*, estuvo a mi cargo llevar a cabo todas las tareas tanto en el *frontend* como en el *backend*, cumpliendo con el objetivo del proyecto en su totalidad. Me ocupé del rediseño de las vistas e interfaces de usuario, siguiendo los lineamientos establecidos, así como de la implementación y gestión de la lógica del servidor y la base de datos.

Por otro lado, el cliente cumplió el rol de Product Manager donde definió la visión general del producto y emitió revisiones y *feedbacks* sobre el estado de las tareas, garantizando que el producto final cumpla con las expectativas.
Junto al cliente se analizó y se propuso por un diseño de estilo *‘marketplace’*, donde se puedan visualizar fácilmente todas las integraciones del sistema mediante tarjetas o *‘cards’* que contengan el nombre de la integración, el logo, el estado en el que se encuentra, junto a una breve descripción de cada una.
Las reuniones semanales con el cliente se llevaron a cabo al final de cada semana, momento en donde se revisaban las tareas completadas y se ajustaban los objetivos y prioridades para la semana siguiente, realizando ajustes para asegurar que el desarrollo avance conforme a las expectativas previstas. En algunas ocasiones, se aprovechaban estos encuentros para la realización de *focus groups*, en los cuales se convocaba a integrantes de otras áreas para presentar las propuestas de diseño y mejoras, analizar su nivel de aceptación y facilitar la toma de decisiones para su implementación.
Durante el proceso de reingeniería y centralización, como se mencionó previamente, se aplicaron diversas metodologías y procesos para garantizar un enfoque estructurado y eficiente. Entre estas metodologías, se destaca la adopción de un enfoque ágil, en particular de la metodología *Scrum*[12], que facilitó a la organización y colaboración con el cliente, permitiendo hacer ajustes rápidos en respuesta a los cambios en los requisitos y sugerencias del mismo.

Por otro lado, se utilizaron otras herramientas durante el proceso de desarrollo para mantener la conversación con el cliente durante el día a día, y así garantizar que el rediseño se implemente de manera eficiente.
Para una comunicación diaria escrita y directa con el cliente, se utilizó la herramienta Slack[13] para discutir problemas e inquietudes. Esta plataforma permitió mantener una conversación fluida de manera ágil, consultando y obteniendo respuestas en tiempo real.
A su vez se optó por la herramienta Figma[14] para diseñar y proporcionar prototipos interactivos, lo que permitió visualizar los cambios propuestos de manera clara y efectiva.
Para la gestión de tareas, se utilizó la herramienta Jira, que proporciona un sistema de *tickets* que organiza el proyecto en diversas tareas asignadas a cada miembro del equipo. En esta plataforma, se cargan los *tickets* o tareas que se llevarán a cabo durante la semana, considerando la capacidad de trabajo de cada integrante, lo que garantiza una distribución efectiva de las responsabilidades y un seguimiento preciso del progreso.
El tablero de Jira se estructuró en ocho columnas (*Blocked, To Do, In Progress, Product Review, In Review, QA, Ready for Release, Done*), cada una representando un estado específico en el flujo de trabajo. Se estableció una correspondencia clara entre las columnas y las tareas asignadas a cada miembro del equipo, permitiendo a todos comprender fácilmente cuándo les corresponde una tarea en particular.

El *feedback* constante y los comentarios del cliente fueron fundamentales para garantizar que los cambios realizados mejoraran la experiencia del usuario y resolvieran los problemas identificados.

2.4. Trabajo Realizado

Para llevar adelante el proyecto, elaboré un análisis exhaustivo de las ventajas y desventajas de distintos diseños propuestos. Además, como mencioné en otras secciones del informe, llevé a cabo estudios de usabilidad utilizando métodos y técnicas de diseño centrado en el usuario, con el objetivo de validar las decisiones, cambios y mejoras implementados.
A través del uso del recorrido cognitivo, pude experimentar personalmente los problemas reportados por los usuarios y validar las observaciones detectadas. Estos errores, relacionados con la usabilidad del sistema, fueron confirmados y corregidos durante el proceso de rediseño. Los *focus groups* realizados permitieron validar estas mejoras y proponer nuevas alternativas.
Se prestó especial atención a las dificultades que enfrentan los usuarios al activar o desactivar una integración, buscando transformar estas barreras en facilidades. Para ello, se consideraron varios aspectos clave: mejorar la organización del menú principal, centralizar las áreas donde se encontraban distribuidas las integraciones, simplificar los pasos en el proceso de configuración, y optimizar la experiencia del usuario enfocándose en la redistribución eficiente de las integraciones dentro del sistema.

Algunas de las decisiones que surgen a raíz del análisis fueron:

- **Agregar una sección en el menú principal que identifique la nueva vista llamada ‘Integrations’.** El objetivo principal es brindar una ubicación directa y lógica para el usuario.

- **Crear una vista nueva que reúna y liste todas las integraciones del sistema.** Para esto se decidió un diseño en forma de tablero. Cada integración estará representada por una tarjeta o ‘card’ donde se mostrará información destacada. Esta disposición permite presentar una mayor información sobre cada integración e identificar las integraciones que se encuentran activas en ese momento para esa compañía.

- **Crear una vista general para mostrar cada integración.** Para esto conviene crear una especie de ‘plantilla’ que permita presentar la información de todas las integraciones, respetando un formato mientras lo que varía es el contenido. Esto nos permite reutilizar componentes lo que ayuda a acelerar el proceso de rediseño. Además, esta disposición de la información ayuda a mantener una coherencia entre todas las integraciones, ubicando datos

importantes en mismos sectores, logrando así que le sea más reconocible al usuario y que se encuentre familiarizado con la organización de la información.

- **Reducir la cantidad de pasos en el proceso de configurar una integración.** En esta tarea es muy importante disminuir el abandono del sitio sin éxito. Esto se logra mediante la consolidación de la información en un único lugar del sistema: la vista de configuración de cada integración. En esta etapa es importante que el usuario sepa en qué estado se encuentra la integración y qué pasos puede hacer o qué cambios puede realizar sobre ella, identificando los datos necesarios para cada caso en particular.

- **Eliminar antiguos accesos.** Al reunir todas las integraciones en una única vista, es fundamental eliminar la posibilidad de configurarlas desde donde se configuraban anteriormente.

Para comenzar con el rediseño del *front-end* se utilizó una librería destacada por su capacidad para crear interfaces de usuario web y nativas. React Js [16] brinda un enfoque moderno y la capacidad de desarrollar aplicaciones web dinámicas y de alto rendimiento. La naturaleza modular de React Js y la posibilidad de crear componentes reutilizables permiten crear una estructura de código limpia y un desarrollo eficiente.
Desde la perspectiva del desarrollo, el uso de React Js ofrece una experiencia coherente y adaptable. Su principal ventaja radica en la libertad para crear componentes que luego pueden reutilizarse en diversas secciones del proyecto, facilitando la colaboración entre distintos desarrolladores.

Con el fin de ilustrar cómo se organiza el proyecto con React JS, a continuación en la Figura 6, se muestra tanto la organización requerida como los nuevos archivos necesarios para el rediseño:

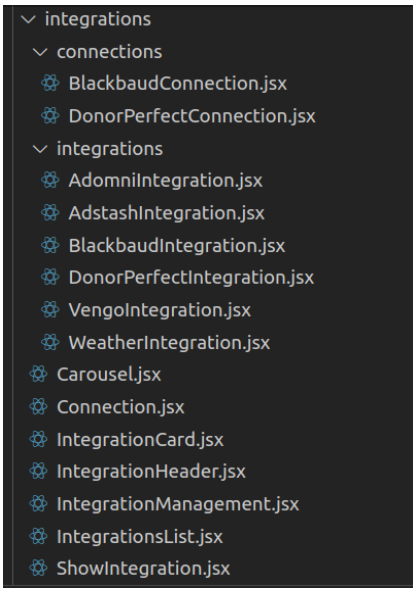

Figura 6. Nuevos archivos para lograr el diseño nuevo.

La estructura contiene en total quince archivos, de los cuales doce fueron agregados y tres fueron modificados. Se crearon archivos específicos para la conexión de cada integración, como también componentes para integrar la visualización, dividiendo un componente grande en varios más pequeños para su reutilización.

Descripción de la estructura:

Integraciones específicas.
`'integrations/'`: contiene los componentes que visualizan y detallan la información sobre cada integración específica. Es la vista general para cada caso.
Conexiones específicas.
`'connections/'`: contiene los componentes que gestionan la configuración de las conexiones para cada integración específica, ya que cada una de ellas requiere distintos datos.
Componentes generales.
Contiene componentes reutilizables que se utilizan en varias vistas de integraciones, como el encabezado (*IntegrationHeader*), la administración de la integración (*IntegrationManagement*) y la lista de integraciones (*IntegrationsList*).
El componente *'IntegrationsList'* muestra la lista de integraciones en forma de tablero. Simplemente hicimos una iteración por las integraciones del sistema, indicando datos importantes y el estado en el que se encuentra a través de una etiqueta en la tarjeta donde se muestra. En esta iteración se utiliza el componente *'IntegrationCard'* que contiene el estilo general de la tarjeta, visualizando el nombre de la integración, descripción y logo, junto a la etiqueta que indica si se encuentra activa.

En tanto en el archivo *'ShowIntegration'* se utiliza como plantilla para todas las integraciones. Está compuesto por otros componentes que contienen a los elementos que tienen en común en todas las vistas de las integraciones.

Estos componentes son:
- *'IntegrationHeader'* : es el encabezado de la vista. En él se muestra el nombre de la integración, el logo y un breve detalle junto a una etiqueta que indica el estado de la misma.
- *'..nombre..Integration'* : Cada archivo de integración, como '*AdomniIntegration'*, '*AdstashIntegration'*, etc., contiene la información detallada de cada integración. Estos componentes manejan la visualización y el detalle específico de cada integración particular.
- *'IntegrationManagement'* : es el componente de configuración. En él se muestra información sobre la última sesión iniciada, como la fecha de *linkeo*, el usuario que lo hizo y la fecha de expiración de la misma. Además, si el usuario posee los permisos necesarios, se muestra un botón que permite activar o desactivar la integración, dependiendo del estado en el que se encuentre la misma.
  Dentro de este componente, se hace el llamado a otro *'..nombre..Connection'*. Cada archivo de conexión, como '*BlackbaudConnection'*, *'DonorPerfectConnection'*, etc., contiene los datos requeridos para la configuración específica de cada conexión.

Nota: para las integraciones de tipo 'vast' (vengo, adomni y adstash) se llama al componente 'connection.erb'.

Estos archivos contienen los botones necesarios para cada caso: activar, desactivar o renovar.
En el caso de activar o renovar, se abre un *popup* con los datos necesarios para la conexión.
Mientras que en el caso de desactivar, este muestra el botón correspondiente y al momento de desactivar la integración, cambia el estado de la misma y se muestra la vista del estado inactiva.

Para la activación de cada integración, se requieren distintos datos, por lo tanto se abrirá un *modal* que dependerá del tipo de integración que se desee activar:

- Blackbaud: se abre una ventana emergente para la autorización OAuth. Esto lo logramos mediante un script en js solicitando el pedido a la url indicada enviando los datos correspondientes.

  A continuación en la Figura 7, se muestra una imagen de la ventana emergente para iniciar sesión en una cuenta Blackbaud.

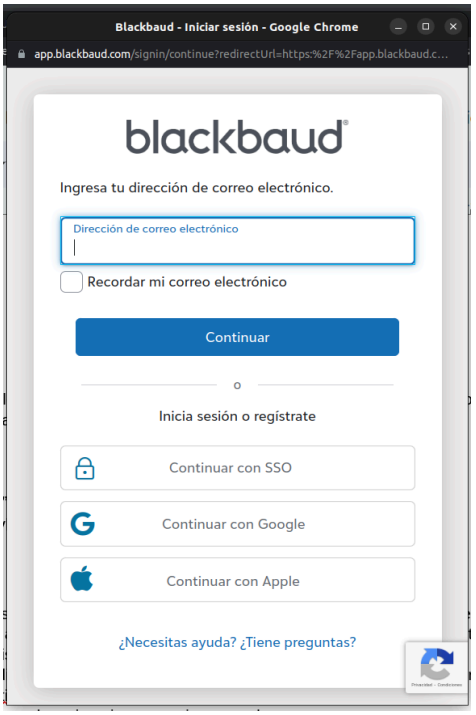

Figura 7. Ventana emergente para iniciar sesión con cuenta de 'Blackbaud'.

- Weather y DonorPerfect: *modal* con un campo de tipo *input* donde se ingresa la *api_key*.
A continuación en la Figura 8, vemos el nuevo diseño del modal para activar una integración en de tipo 'DonorPerfect' o una de clima.

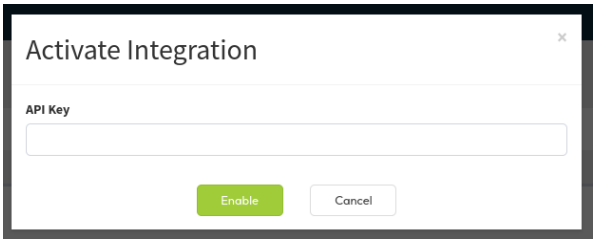

Figura 8. Modal con los campos requeridos para activar una integración de tipo 'DonorPerfect' o de clima.

- Vast: *modal* o *popup* con campos de tipo *input*.

  - Para la integración Vengo se solicitan los datos organization_id, max_time_buffer, un *checkbox* que indica si se activa el *overage* y un mail de reporte.

A continuación en la Figura 9, vemos el nuevo diseño del modal para activar una integración en de tipo ‘Vengo’.

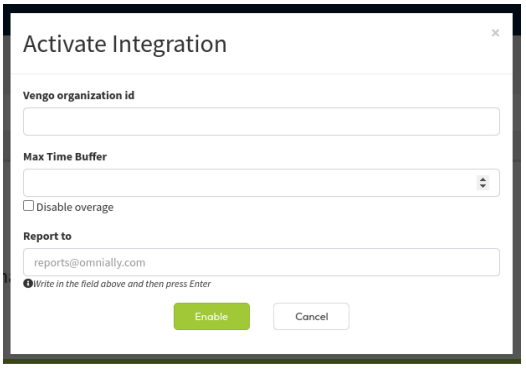

Figura 9. Modal con los campos requeridos para activar una integración de tipo ‘Vengo’.

- Para la integración Adomni se solicitan los datos owner_key, max_time_buffer y mail de reporte.
  A continuación en la Figura 10, vemos el nuevo diseño del modal para activar una integración en de tipo ‘Adomni’.

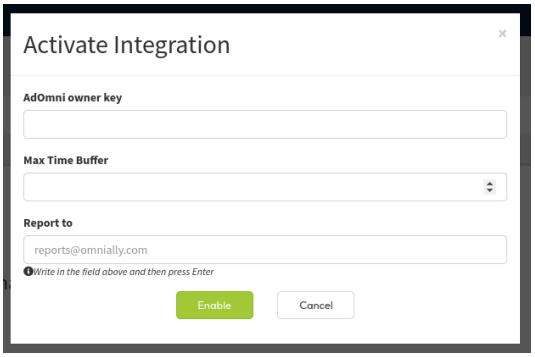

Figura 10. Modal con los campos requeridos para activar una integración de tipo ‘Adomni’.

- Para la integración Adstash se solicitan los datos organization_id, owner_key, max_time_buffer y mail de reporte.
  A continuación en la Figura 11, vemos el nuevo diseño del modal para activar una integración en de tipo ‘Adstash’.

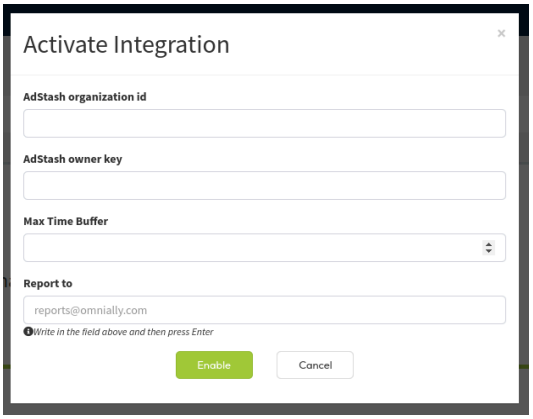

Figura 11. Modal con los campos requeridos para activar una integración de tipo ‘Adstash’.

En todos los casos, el formulario del *popup* junto a los datos proporcionados por el usuario son enviados al *backend* verificando y guardando la conexión en la base de datos.
A diferencia del diseño anterior, se logró concentrar toda la información necesaria para la configuración de la integración en una sola pantalla. En la versión previa, era necesario navegar por varias secciones del sistema, abarcando diferentes pantallas y áreas. De este modo, al agrupar todos los pasos de configuración en una única pantalla, se eliminó la necesidad de seguir el extenso proceso de múltiples pasos que requería la versión anterior.

Para mejorar la gestión y el rendimiento de las integraciones en el sistema, se tuvo que realizar bajo mi cargo varias modificaciones importantes en la base de datos. Un gran cambio en el sistema fue la implementación de un estado para las integraciones. Para esto, cree una tabla nueva en la base de datos llamada *'Integrations'* junto a su modelo y controlador. Esta nueva tabla almacena las seis integraciones existentes en el sistema y almacena información sobre cada una de ellas, incluyendo el estado, la fecha de caducidad y las *API keys*. Además, fue necesario agregar la tabla intermedia *'company_integrations'* que vincula cada integración con una compañía específica (es decir, la compañía del usuario que activa la integración). Esta tabla almacena información indicando si la integración se encuentra activa o no para dicha compañía.

Por otro lado, fue necesario implementar varias modificaciones sobre la tabla donde antes se guardaba esta información: la tabla *'companies'*.
Anteriormente, la información de las integraciones de tipo "vast" (Vengo, Adomni y Adstash) se almacenaba dentro de esta table como un campo más, por ejemplo, el atributo 'adomni_sync'. Esto resultaba ineficiente, ya que implicaba recorrer cada compañía para verificar sus integraciones, lo que hacía las búsquedas muy lentas.
En cambio, la nueva estructura mejora significativamente la eficiencia de las búsquedas y la gestión de las integraciones, evitando la necesidad de recorrer todas las compañías para verificar sus integraciones.
Con este nuevo enfoque se consiguió un manejo más eficiente, sencillo y claro del estado e información de cada integración.

Para lograr estos cambios, estuvo bajo mi responsabilidad realizar varias migraciones a la base de datos propias para el proceso. Algunas de las más importantes fueron:

Clasificación de integraciones.
Cree una migración para clasificar las integraciones. Agregue el atributo 'type' que indica el tipo de integración que corresponde. Para ello clasifique las integraciones en tres tipos:

- '*donor*': incluye a las integraciones utilizadas en los *donorlist* (Blackbaud y Donor Perfect).
- '*weather*': incluye a la integración utilizada en el *widget* de clima (OpenWeather).
- '*vast*': incluye a las integraciones utilizadas para publicidad en los dispositivos (Adomni, Adstash y Vengo).

Este tipo de clasificación no estaba en el diseño anterior, por lo que su incorporacion ha permitido optimizar las búsquedas de las integraciones.

Estado de la integración.
Realice una migración para agregar el estado de la integración. Implemente un sistema enumerado para indicar el estado de cada integración. Para esto fue necesario agregar un estado a las integraciones a través de un elemento enumerable en la clase CompanyIntegration: 0 (desactivado), 1 (activada), 2 (renovado). Este nuevo estado no estaba presente en la versión anterior, lo que ha facilitado una nueva forma de categorizar y localizar las integraciones del sistema.

Eliminación de tablas obsoletas.
Implemente una migración para eliminar las tablas que ya no eran necesarias en el nuevo diseño. Para eso elimine la tabla 'component_blackbaud' y las referencias a datos de las integraciones en la tabla de *'companies'*.

Previo al rediseño existía una tabla que guardaba la información de sesión para las integraciones de tipo ‘blackbaud’ llamada ‘component_blackbauds’, lo cual fue eliminada y cree otra tabla llamada ‘account_integration_tokens’ más intuitiva para gestionar la información de sesión de este tipo de integraciones, guardando ahora más información sobre la sesión. Esto facilitó la asociación de la sesión o *token* con la integración activa para una compañía específica.

A continuación, en la Figura 12 se muestra una imagen que visualiza la estructura de la base de datos sobre las tablas mencionadas previo al rediseño:

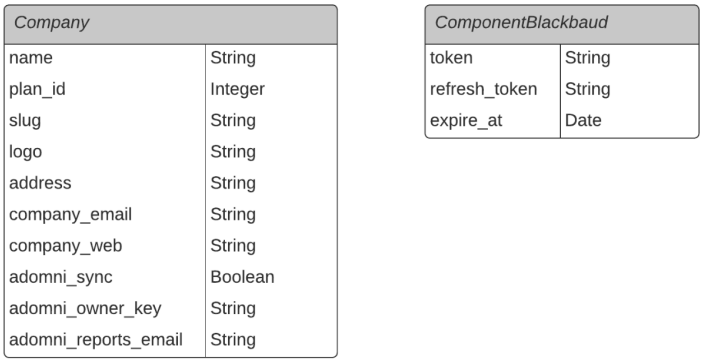

Figura 12. Parte de la estructura de la base de datos antes del rediseño.

Y por otro lado, en la Figura 13, tenemos una imagen que visualiza la estructura de la base de datos sobre dichas tablas después del rediseño:

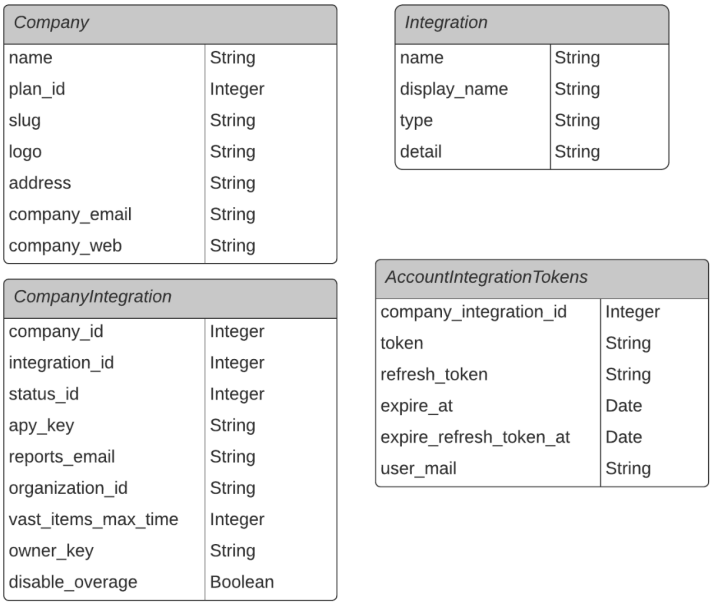

Figura 13. Parte de la estructura de la base de datos después del rediseño.

Este cambio estructural permite una administración más eficiente y escalable de las integraciones, mejorando tanto el rendimiento como la usabilidad del sistema.

A raíz del cambio de la estructura de la base de datos, también se debieron modificar los métodos de creación y activación de los componentes anteriormente mencionados.

En el antiguo enfoque, la información de la sesión para las integraciones de tipo Blackbaud se guardaba en la tabla ‘component_blackbauds’ y para activar una integración de este tipo, se creaba un componente del mismo (ComponentBlackbauds).
A su vez, la búsqueda y gestión de las sesiones dependían del atributo ‘component_uuid’, por lo que implicaba un vínculo directo con el *widget* y el *canvas* donde era activada la integración. Esto hacía que la administración de las sesiones fuera más compleja y menos eficiente.
Además, los datos necesarios para la integración, como por ejemplo la *api_key*, se almacenaban directamente en la tabla de *‘companies’*. Esta estructura resultaba ineficiente, ya que la tabla contiene campos no utilizados en todos los casos, complicando la gestión y el acceso a los datos.
En cambio, en el nuevo enfoque, se introdujo la tabla ‘account_integration_tokens’ para gestionar solo la información de la sesión de las integraciones. Ahora al activar una integración de tipo Blackbaud, se crea un componente de tipo AccountIntegrationTokens. Así, la sesión o *token* se asocia directamente con la integración activa para una compañía específica. Esto se gestiona a través de la tabla company_integrations, mejorando la eficiencia y claridad de la relación.

A continuación en la Figura 14 vemos cómo se creaba y guardaba anteriormente una sesión de *token* para las integraciones de tipo *blackbaud*:

```
    if response.success?
      response_data = JSON.parse(response.body)
      component = ComponentBlackbaud.new(
        component_uuid: data["widget"],
        token: response_data["access_token"],
        refresh_token: response_data["refresh_token"],
        expire_at: Time.now + response_data["expires_in"]
      )
      if component.save
        redis.call "PUBLISH", "dashboard:browsers_update",
          Oj.dump(
            component_id: data["widget"],
            type: "component_change"
          )
      end
      if response.status == 200
        flash[:success] = "Successfully authenticated with blackbaud."
      else
        flash[:error] = "Failed to authenticate with blackbaud."
      end
    end
    render "blackbaud/new"
  end
```

Figura 14. Implementación para crear y guardar una sesión de Blackbaud antes del rediseño.

Y en la Figura 15 observamos cómo se crea y guarda una sesion de tipo *token* de una integración *blackbaud* ahora:

```
  if response.success?
    integration = Integration.blackbaud
    response_data = JSON.parse(response.body)
    company_integration = CompanyIntegration.first(
      company_id: ghost_or_current.company_id,
      integration_id: integration.id
    )

    if company_integration.blank?
      company_integration = CompanyIntegration.new(
        company_id: ghost_or_current.company_id,
        integration_id: integration.id,
        api_key: "blackbaud",
        enabled: true,
        report_emails: "",
        status_id: 1
      )
    end
    company_integration.enable!

    if company_integration.save
      ::AccountIntegrationToken.create(
        company_integration_id: company_integration.id,
        token: response_data["access_token"],
        refresh_token: response_data["refresh_token"],
        expire_refresh_token_at: Time.now + response_data
        ["refresh_token_expires_in"],
        expire_at: Time.now + response_data["expires_in"],
        user_email: response_data["email"]
      )
    end
    if response.status == 200
      flash[:success] = "Successfully authenticated with blackbaud."
    else
      flash[:error] = "Failed to authenticate with blackbaud."
    end
  end
  render "blackbaud/new"
end
```

Figura 15. Implementación para crear y guardar una sesión de Blackbaud después del rediseño.

Podemos ver en la Figura 16 cómo en el diseño anterior se buscaba el componente Blackbaud por 'component_uuid' dependiendo completamente del *widget* y del *canvas* donde se activaba la integración y no la compañía del usuario que la activo.

```
  on delete, "blackbaud_connection" do
    destroy = ComponentBlackbaud.where(component_uuid: req.params["component_uuid"]).destroy
    exit_code = destroy == 1 ?
      "The connection was successfully removed" :
      "There is no connection for the given component"
    res.write exit_code.to_json
  end
end
```

Figura 16. Implementación para eliminar el token de una sesión de Blackbaud antes del rediseño.

A su vez, como podemos ver en la Figura 17, la eliminación de una sesión de token ahora se realiza solo para la compañía que activó la integración. Gracias a la nueva clase AccountIntegrationToken, asociada a company_integration, es posible acceder y gestionar fácilmente la relación utilizando el método *account_integration_token* de un company_integration de la forma: *company_integration.account_integration_token*. Esto asegura que la integración se guarda correctamente y la sesión se elimina de manera eficiente.

```
on delete, param("company_integration_id"), "blackbaud_connection"
do |company_integration_id|
  company_integration = CompanyIntegration.first(id:
  company_integration_id)
  token_integration = company_integration.account_integration_token
  token_integration&.destroy
  company_integration.disable!
  company_integration.save
  res.write ""
end
end
```

Figura 17. Implementación para eliminar el token de una sesión de Blackbaud después del rediseño.

Como podemos ver en la Figura 17, la nueva estructura permite búsquedas y gestión de integraciones más rápidas y eficientes, al centralizar la información relevante en tablas específicas. La asociación directa de *tokens* de sesión con integraciones activas simplifica la administración y elimina la dependencia de identificadores de componentes específicos. Con la eliminación de la tabla component_blackbauds y la integración de account_integration_tokens, el mantenimiento de la base de datos se vuelve más sencillo y menos propenso a errores.
Estos cambios aseguran que la información de la integración y la sesión se gestionen de manera más eficiente y ordenada, mejorando el rendimiento general del sistema.

Otro de los grandes cambios que lleve a cabo fue la verificación previa a utilizar la integración. Antes del rediseño, los datos de acceso se solicitaban en el momento de querer utilizar la integración. Por ejemplo para solicitar la lista de nombres de donantes de un *donorlist*, se solicitaban los datos a través de *inputs* dentro de la configuración del *widget* y si los datos eran correctos, se accedía a la información. Ahora, cuando el usuario quiere usar la integración, se verifica el estado en el que se encuentra la integración. Si no está activa para la compañía del usuario que la quiere usar, se muestra un mensaje informativo que indica que debe activarla previamente, brindando un *link* directo para que la configure. En cambio, si al momento de usar la integración se encuentra activa, los datos se brindan correctamente. En el caso de estar activa al momento de crear el *widget* y posteriormente se desactiva, se muestra dentro del *widget* un mensaje indicando que la integración está desactivada.

Este cambio facilita y mejora muchos aspectos, uno de ellos es que al verificar que la integración se encuentre activa, si el usuario por ejemplo desea crear muchos *widgets* que utilizan esa integración, una vez activada no tendrá que configurarla nuevamente. En cambio si lo comparamos con el diseño anterior, el usuario debía repetir el proceso de configuración por cada *widget* creado. Lo cual es repetitivo y puede provocar que el usuario se canse y abandone la tarea.

En la Figura 18 podemos ver cómo se verifica si la integración está activa o no con el estado *'connected'*, y en caso negativo muestra un cartel informando la situación al usuario.

```
return (
  <div>
    {
      connected &&
      <BlackbaudListSelectionSettings
        companyIntegration={companyIntegration}
        listType={listType}
        list={list}
        onListTypeChange={({ list_type }) => localOnChange({ list_type })} //
        eslint-disable-line
        onListChange={({ list }) => localOnChange({ list })}
      />
    }
    {
      !connected &&
      <p className='editor_list-control_empty'>
        Integration inactive. Please go to <a href="/integrations"> Manage
        Integrations</a> to activate
      </p>
    }
  </div>
)
```

Figura 18. Implementación en React para comprobar el estado de la integración para mostrar la información correspondiente.

Para esta funcionalidad se utiliza el hook que proporciona React llamado *useState*. Este permite agregar un estado local a un componente funcional. Es decir, proporciona una forma de declarar una variable de estado y una función para actualizarla, permitiendo manipular el estado de un componente sin tener que escribir una clase.
También se hizo uso de otros *hooks* de React a lo largo del proceso como *useSelector(), useEffect()* y *useRef()*.
Por otro lado, durante el proceso de rediseño, se presentó un desafío significativo: la *API* de clima utilizada previamente no cumplía con los nuevos requerimientos del cliente. Anteriormente, se utilizaba AccuWeather para este tipo de *widget*, ya que proporcionaba información gratuita útil para nuestras funcionalidades. Sin embargo, con el tiempo, estas funciones dejaron de ser gratuitas y, como resultado, dejaron de funcionar adecuadamente.
Dado que esta funcionalidad no era muy utilizada, se decidió cambiar de *API* para cumplir con los requisitos y mejorar la funcionalidad del *widget*. Opté por OpenWeatherMaps como la nueva *API*, a pesar de que esta tampoco era completamente gratuita. Para esto tuve que adaptar y desarrollar algunas soluciones para obtener los datos necesarios que el cliente deseaba, lo que implicó un intercambio de ideas y negociaciones, dado que el tema excedía el marco del *ticket* de diseño que estaba desarrollando.

Uno de los cambios más importantes fue dentro de la búsqueda de la información del clima. En la configuración del *widget* del clima, al ingresar el nombre de una ciudad, se solicitaba información correspondiente, lo que exigió una modificación completa en la solicitud.

Previamente se solicitaba la petición a la *api* de accuweather de la siguiente forma:

```
fetchJson(`https://api.accuweather.com/locations/v1/cities/search?apikey=${co
nfig.apikey}&q=${input}`)
```

Ahora, con los nuevos cambios de *api*, se solicita la información de la siguiente forma:

```
fetchJson(`https://api.openweathermap.org/geo/1.0/direct?q=${input}&limit=10&
appid=${apiKey}`)
```

Lo mismo sucede con la solicitud de la información al mostrar el *widget* de clima. El cambio a la *url* fue:

```
const forecastUrl =
`https://api.openweathermap.org/data/2.5/${typeGetData}?q=${config.city},${co
nfig.country}&appid=${apiKey}&units=imperial`
```

Además, fue necesario crear una equivalencia entre las imágenes o iconos que se mostraban en el *widget* relacionado al clima. Para lograr esto, implemente un método que convierte cada icono de OpenWeather a su correspondiente en AccuWeather. Por ejemplo, el icono '01d' de OpenWeather, que representa un día soleado, fue mapeado para corresponder al icono '01' en AccuWeather. Este proceso asegura que cada condición climática se representará de manera coherente entre ambos servicios. En la Figura 19, se puede ver el proceso que parsea los iconos de cada *api*, en el cual la columna izquierda hace referencia a los iconos de openWeatherMap y la columna derecha a los iconos de AccuWeather:

```
export const iconsWeatherEquivalences = {
  "01d": '01',
  "01n": '33',
  "02d": '03',
  "02n": '34',
  "03d": '07',
  "03n": '07',
  "04d": '08',
  "04n": '08',
  "09d": '12',
  "09n": '12',
  "10d": '14',
  "10n": '39',
  "11d": '15',
  "11n": '15',
  "13d": '22',
  "13n": '22',
  "50d": '11',
  "50n": '11'
}
```

Figura 19. Equivalencias de los iconos de la *api* de AccuWeather a los iconos de la *api* de OpenWeatherMaps.

Además, creé un nuevo archivo llamado ‘utils.js’, en el cual agrupé métodos de utilidad, incluyendo el previamente mencionado y otros diseñados para el manejo de la información del *widget* del clima. Estos métodos se encargan tanto de realizar las solicitudes como de *parsear* los resultados y presentar la información de acuerdo con las especificaciones del cliente.

Otros cambios y consideraciones técnicas que surgieron durante el proceso de rediseño fueron:

Compatibilidad y Consistencia:
Dado que la funcionalidad de las integraciones ya existía en el sistema, fue necesario mantener la consistencia de los componentes. Se realizaron pruebas exhaustivas para asegurar que los *canvas* y *widgets* existentes siguieran funcionando correctamente tras los cambios.
Además se añadieron mensajes de error y enlaces para reconfigurar integraciones desactivadas o caducadas, ya que antes el componente se mostraba en blanco sin ningún mensaje de error, haciendo que el usuario no supiera qué estaba pasando ni tuviera información sobre el problema. Ahora, con el mensaje de error, el usuario es informado de la situación y sabe que debe activar la integración para que la información se muestre correctamente.

Cambios en la Configuración de Integraciones:
Se modificaron los permisos en la configuración de las integraciones. Antes, cada usuario tenía el poder de modificar si una integración estaba activa o no. Ahora, este acceso ha cambiado a nivel de empresa. Esto significa que en lugar de que cada usuario individual active o desactive una integración, se registra la empresa a la que pertenece el usuario y la activación o desactivación se aplica a todos los usuarios de esa empresa. Esta modificación asegura que la configuración de las integraciones sea coherente para toda la empresa, facilitando la gestión y mejorando la uniformidad en el uso de las integraciones.
Además este cambio se aplica únicamente a la misma empresa y no se hereda a empresas hijas ni empresas padre, y es válido solo para los administradores de la misma empresa.

Testeos:
Integraciones como Vengo y DonorPerfect, siendo las más utilizadas, requirieron muchas pruebas y chequeos constantes para asegurar que todas las empresas y dispositivos que las usaban en ese momento no sufrieran ningún error en su funcionamiento.

Como resultado del proceso de rediseño realizado, se logró alcanzar importantes avances y mejoras tanto en la optimización del sistema como en la experiencia general del usuario. Además, a lo largo de este proceso, se realizaron pruebas de usabilidad con la participación del cliente y el CEO de la empresa, lo que permitió evaluar y validar las mejoras de forma continua a medida que se iban implementando.

Por último y como parte final del proceso, desarrolle y documente un manual de usuario a modo de ayuda para el cliente y para los mismos miembros del equipo técnico.
Para esta etapa se utilizó la herramienta Confluence[17] para documentar cada paso, bajo ciertas pautas de organización y escritura del documento sugeridas por el cliente. Esta herramienta permite la colaboración en tiempo real con los demás miembros del equipo, permitiendo realizar *feedback* y discutir detalles a través de comentarios y correcciones en el mismo.
Cuenta con un control de versiones, por lo que se podía retroceder a una versión antigua en caso de grandes modificaciones.
Además está integrado con la herramienta Jira, por lo que se utilizaron enlaces directos a *tickets* para una mejor comprensión de cada paso.

2.5. Líneas de posibles Trabajos Futuros

Las mejoras implementadas en la aplicación Omnially no solo han optimizado la experiencia actual, sino que también han establecido una base sólida para futuras actualizaciones y desarrollos. Estas mejoras iniciales son fundamentales, ya que proporcionan un marco de referencia que facilitará la implementación de cambios más amplios en el futuro. Al abordar áreas clave de la aplicación, se han identificado estrategias y metodologías que podrán aplicarse de manera consistente a otras funciones y aplicaciones dentro de la empresa.

A partir de lo desarrollado,una línea de posibles trabajos futuros podrían incluir:

- **Reorganización del *'Navbar'***: La barra de navegación principal podría ser rediseñada para centralizar y destacar las funciones clave, mejorando la accesibilidad y usabilidad del sistema. Este rediseño se basaría en las mejoras ya realizadas, que han demostrado ser efectivas en la simplificación de la navegación.

- **Optimización de las vistas generales**: Se proyecta una reorganización más profunda de las listas de elementos, como *canvas*, compañías y usuarios, con el objetivo de hacer el sistema más intuitivo y fácil de usar. Las optimizaciones actuales han mostrado que una mejor organización puede reducir significativamente el tiempo de aprendizaje, lo cual guiará futuras iteraciones.

- **Actualización del diseño visual**: A partir de las mejoras ya implementadas en aspectos como la página de inicio, la barra de navegación y la paleta de colores del sistema, se prevé continuar refinando el diseño visual para lograr una estética más cohesiva. Esto no solo mejorará la experiencia del usuario, sino que también establecerá un estándar visual que podría aplicarse en otras partes de la aplicación y en diferentes productos de la empresa.

Referencias Bibliográficas

[1] Norman, D. (2013). *The Design of Everyday Things*. https://dl.icdst.org/pdfs/files4/4bb8d08a9b309df7d86e62ec4056ceef.pdf

[2] Unger, R., & Chandler, C. (n.d.). *A Project Guide to UX Design*. (s. f.). Google Books. https://books.google.com.ar/books?id=dF7li-90OYQC&printsec=frontcover&redir_esc=y#v=onepage&q&f=false

[2] Nielsen, J., & Mack, R. L. (1994). *Usability Inspection Methods*. *Amazon.com: Books*. https://www.amazon.com/Usability-Inspection-Methods-Jakob-Nielsen/dp/0471018775

[3] Flaherty, K. (n.d.). *Evaluate Interface Learnability with Cognitive Walkthroughs*. Nielsen Norman Group. https://www.nngroup.com/articles/cognitive-walkthroughs/

[4] Omnially. (n.d.). *Digital Donor Recognition Walls: Everything You Need to Know.* https://www.omnially.com/blog/digital-donor-recognition-walls-what-to-know

[5] Blackbaud. (n.d.). *Raiser's Edge NXT Integration: Advanced Donor Management Solution.* https://www.blackbaud.com/products/blackbaud-raisers-edge-nxt

[6] DonorPerfect. (n.d.). *DonorPerfect Integration: Donor Management Software.* https://www.donorperfect.com/

[7] OpenWeatherMap.org. (n.d.). *Weather API: Tools for Real-Time Weather Data.* https://openweathermap.org/api

[8] AdStash. (n.d.). *AdStash Integration: Monetization Solution for Digital Signage.* https://www.adstash.com/

[9] AdOmni. (n.d.). *AdOmni Integration: Advertising Platform for Digital Signage Campaigns.* https://www.adomni.com/

[10] Vengo. (n.d.). *Vengo Integration: Digital Distribution Platform for Interactive Screens.* https://vengolabs.app/

[11] W3C Initiative. (n.d.). *Navigation design*. Web Accessibility Initiative (WAI) https://www.w3.org/WAI/curricula/designer-modules/navigation-design/

[12] Scrum.org. (2020). *The 2020 Scrum Guide*. https://scrumguides.org/scrum-guide.html

[13] Slack. (n.d.). *Communication and Collaboration Tools for Teams.* https://slack.com/intl/es-ar/

[14] Figma. (n.d.). *Guide to Developer Handoff in Figma*. https://www.figma.com/best-practices/guide-to-developer-handoff/

[15] Atlassian. (n.d.). Introduction to Jira: Getting Started Guide.
https://www.atlassian.com/software/jira/guides/getting-started/introduction

[16] Fedosejev, A. (2015). *React.js Essentials: A fast-paced guide to designing and building scalable and maintainable web apps with React Js*.
https://legacy.reactjs.org/docs/getting-started.html

[17] Atlassian. (n.d.). *Confluence Guide*.
https://www.atlassian.com/software/confluence/resources/guides/how-to/whiteboards-planning-prioritization

[17] Atlassian. (n.d.). *Tutorial: Using Confluence Cloud and Jira together*
https://www.atlassian.com/software/confluence/resources/guides/extend-functionality/confluence-jira