



**Eloy Lozano Barrios**



**CPR Liceo La Paz**

*Desarrollo de Aplicaciones Web*

Jesús Ángel Pérez Roca

“*Technology alone is not enough – It’s technology married with liberal arts that yields us the result that makes our hearts sing*.”

* Steve Jobs

# Resumen

Deskify es una plataforma web fundamentada en un sistema de ticketing destinado a la gestión de incidencias en el ámbito empresarial. El propósito de este proyecto es proveer una solución centralizada para el seguimiento y la resolución de los tickets de soporte, con el fin de optimizar la comunicación entre los usuarios finales y los agentes técnicos.

La plataforma se sustenta en tecnologías vanguardistas como Svelte, Astro y Spring Boot, garantizando robustez, escalabilidad y una experiencia de usuario sobresaliente. Además, incorpora una arquitectura basada en microservicios, lo que permite una integración eficiente con sistemas externos y futuras ampliaciones.

Deskify emerge como resultado de la experiencia como técnico de soporte, con la intención de abordar las deficiencias identificadas en otras herramientas de helpdesk. Su diseño y funcionalidades están orientados a simplificar los procesos, reducir los tiempos de respuesta y ofrecer una experiencia más fluida y eficiente tanto para los usuarios como para los agentes técnicos. A continuación, se presentan las principales características de este software:

* Gestión de incidencias: Creación, actualización y seguimiento de los tickets.
* Roles y permisos: Diferenciación de accesos para administradores, supervisores, managers, agentes y usuarios.
* Dashboard: Estadísticas en tiempo real sobre el estado de los tickets y el rendimiento del equipo.
* API Rest: Exposición de servicios para la integración con otras plataformas.
* Diseño intuitivo: Interfaz minimalista y adaptable para garantizar la accesibilidad desde cualquier dispositivo.

Algunos de los objetivos que promete cumplir Deskify son:

1. La mejora de la efectividad en la gestión de soporte técnico.
2. Reducir al máximo los tiempos de respuesta y resolución de tickets.
3. Proveer a las empresas de una herramienta que fomente la colaboración y la productividad.
4. Mejora en la satisfacción del cliente al agilizar la atención.

Por último, se detallan todas las tecnologías empleadas en el desarrollo de esta aplicación, organizadas según las diferentes partes del sistema:

* Back-End: Spring Boot, JPA, Hibernate, ModelMapper, Lombok, Rest, Swagger, Thymeleaf, JWT.
* Front-End: Svelte, SvelteKit, Tailwind, Astro.
* Base de datos: MySQL, Amazon Bucket S3.

# Abstract

Deskify is a web-based platform that utilises a ticketing system for the management of incidents at the enterprise level. The objective of this project is to provide a centralised solution for the tracking and resolution of support tickets, thereby enhancing communication between end users and technical agents. This platform is founded on cutting-edge technologies such as React and Spring Boot, ensuring robustness, scalability and an exceptional user experience. Additionally, it employs a microservices-based architecture, enabling efficient integration with external systems and facilitating future extensions.

The impetus for Deskify's development stemmed from the author's personal experience as a support technician, driven by the recognition of deficiencies in existing helpdesk tools. The design and functionalities of Deskify are meticulously crafted to streamline processes, reduce response times, and provide a more intuitive and efficient experience for both end users and technical agents. The software's primary features are as follows:

* Incident management: creation, update and tracking of tickets.
* Roles and permissions: Access differentiation for admins, supervisors, managers, agents and users.
* Dashboard: Real-time statistics on ticket status and team performance.
* API Rest: Exposure of services for integration with other platforms.
* Intuitive design: Minimalist and adaptable interface to ensure accessibility from any device.

The following objectives are proposed to be accomplished by Deskify:

1. Improving the effectiveness of technical support management.
2. The reduction of response and ticket resolution times to the maximum.
3. The provision of a tool that fosters collaboration and productivity for companies.
4. Improved customer satisfaction through the streamlining of customer service.

Finally, the technologies employed in the development of this application are listed below, organized according to the different parts of the system:

1. Back-End: Spring Boot, JPA, Hibernate, ModelMapper, Lombok, Rest, Swagger, Thymeleaf, JWT.

* Front-End: Svelte, SvelteKit, Tailwind, Astro.

1. Database: MySQL, Amazon Bucket S3.

# Palabras Clave

Astro: Framework moderno para construir sitios web rápidos y optimizados, que permite usar múltiples tecnologías de frontend en un mismo proyecto.

Brainstorming: Herramienta de trabajo grupal que facilita el surgimiento de nuevas ideas sobre un tema o problema determinado.

Dashboard: Panel visual que muestra datos e información relevante en tiempo real, facilitando el seguimiento y análisis de métricas clave.

HelpDesk: Servicio de soporte técnico que atiende y resuelve problemas de los usuarios finales dentro de una organización.

JavaMail: API Java que facilita el envío y recepción de correo electrónico desde código Java a través de protocolos SMTP, POP3, y IMAP.

JPA (Java Persistence API): Especificación que facilita la interacción con bases de datos relacionales mediante objetos Java.

JWT (JSON Web Tokens)**:** Tecnología utilizada para la autenticación y autorización segura en aplicaciones web, garantizando la integridad de las sesiones de usuario.

Lombok: Biblioteca de Java que simplifica el código repetitivo generando automáticamente getters, setters, constructores y otros métodos comunes.

Microservicios: Arquitectura que descompone una aplicación en servicios pequeños e independientes para mejorar su escalabilidad y mantenimiento.

MoodBoard: Collage visual que agrupa imágenes, colores y tipografías para transmitir la inspiración estética o el estilo de un proyecto.

ModelMapper: Herramienta que simplifica el mapeo de datos entre objetos de dominio y DTOs (Data Transfer Objects).

Must: forma básica y más directa de decir que algo es obligatorio. Significa que no tienes más remedio que hacer (o no hacer) una acción.

MySQL: Sistema de gestión de bases de datos relacional que almacena toda la información de los tickets, usuarios y operaciones del sistema.

Sistema de Ticketing: Plataforma usada para gestionar incidencias, solicitudes o tareas mediante la asignación de tickets, facilitando la comunicación entre usuarios y agentes técnicos.

Spring Boot: Framework de Java para el desarrollo rápido de aplicaciones web robustas y escalables con soporte para microservicios.

Svelte: Framework de JavaScript que convierte los componentes en código optimizado durante la compilación, ofreciendo alto rendimiento y una sintaxis sencilla.

Tailwind: Framework de CSS basado en clases utilitarias que permite diseñar interfaces modernas directamente desde el HTML de forma rápida y personalizable.

# Agradecimientos

En primer lugar, es preciso expresar el más sincero agradecimiento a Mónica, mi profesora de diseño, quien ha ejercido un rol de suma importancia en el desarrollo de este proyecto desde sus inicios. Su constante apoyo y motivación, así como sus valiosas contribuciones en términos de diseño y sugerencias para la creación del mismo, han sido elementos fundamentales para la conformación de Deskify.

Asimismo, es preciso reconocer la contribución de los demás docentes, quienes, con su paciencia y dedicación, facilitaron la resolución de las dudas surgidas durante el proceso de creación. Su orientación y conocimiento resultaron fundamentales para superar los obstáculos encontrados en el desarrollo del proyecto.

De manera muy especial, quiero agradecer a mi pareja, por estar siempre a mi lado desde el primer día. Gracias por aguantar todos mis quebraderos de cabeza, por soportar mis frustraciones cuando las cosas no salían como esperaba, y por motivarme cada día a seguir adelante. Tu apoyo incondicional ha sido una parte fundamental de este camino.

Por último, es necesario expresar un agradecimiento a los compañeros Dani y Miguel, cuya ayuda, consejos y observaciones durante el desarrollo del proyecto condujeron a mejoras significativas que garantizaron el correcto funcionamiento de la plataforma. Su colaboración resultó un apoyo significativo para alcanzar este objetivo.

A todos ustedes, gracias por formar parte de este viaje.

# Contenido

[Resumen 3](#_Toc197252108)

[Abstract 4](#_Toc197252109)

[Palabras Clave 5](#_Toc197252110)

[Agradecimientos 7](#_Toc197252111)

[Contenido 8](#_Toc197252112)

[Introducción 10](#_Toc197252113)

[Motivación y objetivos 11](#_Toc197252114)

[Estado del arte 12](#_Toc197252115)

[Caso de estudio 13](#_Toc197252116)

[Implementación de tickets por correo electrónico 13](#_Toc197252117)

[Diagramas 15](#_Toc197252118)

[Base de datos 15](#_Toc197252119)

[Entidad-Relación 16](#_Toc197252120)

[Clases 17](#_Toc197252121)

[Casos de Uso 19](#_Toc197252122)

[Diseño 21](#_Toc197252123)

[Guía de estilos 22](#_Toc197252124)

[MoodBoard 24](#_Toc197252125)

[Wireframes 25](#_Toc197252126)

[Página Web 25](#_Toc197252127)

[Aplicación Web 27](#_Toc197252128)

[Panel de Administrador 29](#_Toc197252129)

[Mockups 30](#_Toc197252130)

[Errores 30](#_Toc197252131)

[Página Web 32](#_Toc197252132)

[Aplicación Web 33](#_Toc197252133)

[Panel de administrador 36](#_Toc197252134)

[Mapa de navegación 37](#_Toc197252135)

[Componentes UI 38](#_Toc197252136)

[Desarrollo del Proyecto 40](#_Toc197252137)

[Backend - API con Spring 40](#_Toc197252138)

[Frontend - Landing Page con Astro 40](#_Toc197252139)

[Frontend - Aplicación con Svelte 41](#_Toc197252140)

[Gestión de tickets y sistema de roles 42](#_Toc197252141)

[Pruebas y validación 42](#_Toc197252142)

[Manual del Administrador 43](#_Toc197252143)

[Manual del Usuario 44](#_Toc197252144)

[Viabilidad Tecno-Económica 45](#_Toc197252145)

[Viabilidad Técnica 45](#_Toc197252146)

[Viabilidad Económica 46](#_Toc197252147)

[Trabajo a futuro 48](#_Toc197252148)

[Conclusiones 49](#_Toc197252149)

[Biblioteca de Recursos 50](#_Toc197252150)

# Introducción

El desarrollo de Deskify surge como una respuesta directa a los desafíos que me he enfrentado a durante mi experiencia como técnico de soporte. Durante este período, trabajé con alguna plataforma de ticketing, enfrentándome a las limitaciones de herramientas que, aunque cumplían su función, carecían de personalización y una experiencia de usuario intuitiva. Estos mismos defectos me hicieron reflexionar sobre una posible solución, la creación de mi propio software. Una simple mejora en el diseño puede marcar la diferencia tanto para el técnico como para el usuario final.

La elección de este proyecto no fue casual, Deskify combina mi pasión por el desarrollo junto con mi conocimiento sobre el soporte técnico. Además, el desarrollo representa un desafío técnico que me permite demostrar mis conocimientos adquiridos durante este curso. Quise crear algo que no solo cumpliera con los requisitos académicos, sino también algo que me ayudase a descubrir nuevas tecnologías y formas de trabajo a la hora de la creación de un proyecto real.

En resumen, Deskify no es solo un proyecto académico; es una demostración de mi compromiso con el desarrollo y la mejora continua en el ámbito de las tecnologías de la información. Con este trabajo, aspiro a demostrar mi capacidad de aprendizaje y adaptación a los retos tecnológicos.

# Motivación y objetivos

El principal objetivo de este programa es proporcionar una solución integral que simplifique la gestión de incidencias y mejore la comunicación entre los diferentes roles involucrados en el proceso de soporte técnico. La presente plataforma es la creación de un entorno en el que la resolución de problemas sea más eficiente, lo que se traduciría en una reducción de los tiempos de respuesta y un aumento de la satisfacción de los usuarios finales.

Se identificó que las herramientas tradicionales presentan limitaciones, siendo más notable su rigidez y falta de adaptación a las necesidades específicas de cada organización. Esta constatación motivó la concepción de Deskify, fundamentada en una filosofía de personalización y escalabilidad. El presente estudio tiene como objetivo demostrar que la plataforma puede adaptarse con facilidad a diversos sectores y equipos de trabajo, ofreciendo funcionalidades modulables que permitan a las empresas configurarla según sus requisitos particulares.

En un contexto de creciente interconectividad, la accesibilidad y la usabilidad intuitiva se convierten en pilares fundamentales. En este sentido, Deskify incorpora una interfaz contemporánea y minimalista, simplificando la gestión tanto para los técnicos como para los usuarios no técnicos. Este enfoque tiene como objetivo principal reducir la curva de aprendizaje, permitiendo que cualquier individuo pueda utilizar la herramienta de manera eficiente desde el primer momento.

En suma, Deskify constituye un proyecto concebido con la finalidad de generar un impacto positivo y tangible en el ámbito del soporte técnico. El propósito de este estudio es demostrar que esta plataforma no solo constituye una solución innovadora, sino que también sirve como un ejemplo tangible de cómo la tecnología puede ser una herramienta para transformar procesos, mejorar la productividad y, en última instancia, facilitar el trabajo de las personas.

# Estado del arte

Deskify no constituye una novedad en el ámbito de las aplicaciones de ticketing, pues en la actualidad existen numerosas alternativas disponibles en el mercado, tales como *Zendesk*, *Jira Service Management*, *FreshDesk* o *Salesforce Service Cloud*. No obstante, este proyecto ha emergido con el fin de consolidarse en el mercado mediante la explotación de las debilidades de sus competidoras y la optimización de sus fortalezas, para desarrollar un software y un diseño singulares para el usuario.

A diferencia de estas herramientas, Deskify apuesta por una experiencia de usuario más accesible y personalizable, enfocándose en una interfaz minimalista y moderna, sin sacrificar la funcionalidad avanzada que exigen las organizaciones actuales. Además, incorpora flujos de trabajo intuitivos que agilizan la gestión de tickets y fomenta la colaboración eficiente entre los diferentes roles del sistema.

A continuación, se presentan las mejoras de Deskify con respecto a las alternativas existentes

* Personalización Extrema: Mientras que muchas soluciones existentes ofrecen opciones limitadas de personalización, Deskify permite a las empresas ajustar flujos de trabajo, roles, categorías y diseños según sus necesidades específicas.
* Interfaz Intuitiva: Diseñada con un enfoque moderno y minimalista, reduce la complejidad de uso, minimizando la curva de aprendizaje tanto para técnicos como para usuarios finales. Combinando lo mejor de las 4, dividiendo por páginas las diferentes funcionalidades, al contrario de *Salesforce Service Cloud*.
* Accesibilidad Multicanal Simplificada: Aunque herramientas como *Zendesk* y *Freshdesk* son multicanal, Deskify se centra en una integración más simplificada y eficiente, priorizando los canales más relevantes para cada organización.
* Roles y Permisos Flexibles: Ofrece una estructura de roles completamente adaptable, lo que facilita la asignación de responsabilidades y el seguimiento del rendimiento, algo que puede ser limitado en otras plataformas.
* Optimización de Recursos: Al estar diseñado desde cero con tecnologías modernas, - Deskify garantiza un rendimiento más eficiente y una integración fluida con herramientas externas, reduciendo tiempos de carga y mejorando la experiencia general.
* Filosofía de Mejora Continua: Deskify incorpora un enfoque ágil para adaptarse rápidamente a las nuevas tendencias y necesidades del mercado, algo que en ocasiones es más lento en plataformas consolidadas debido a su estructura compleja.

# Caso de estudio

## Implementación de tickets por correo electrónico

**1. Introducción al problema**

En el ámbito organizacional, se observa una tendencia entre los usuarios a recurrir al soporte técnico a través del correo electrónico en lugar de iniciar sesión en una plataforma específica. No obstante, múltiples herramientas de helpdesk no gestionan adecuadamente estos correos electrónicos, lo que deriva en duplicidades, pérdida de información o carencia de integración fluida. Deskify emerge como una solución a esta limitación, permitiendo que los usuarios envíen tickets directamente por correo electrónico. Estos tickets son procesados automáticamente en el sistema, garantizando la trazabilidad y el orden de los mismos.

**2. Objetivo clave**

La funcionalidad en cuestión tiene como propósito facilitar que los correos electrónicos remitidos a una dirección determinada (como, por ejemplo, soporte@deskify.com) sean procesados de manera automática y convertidos en tickets dentro del sistema de Deskify. Este proceso comprende la extracción de la información relevante del mensaje, como el asunto, el contenido y los archivos adjuntos, y su asignación a un técnico o a una cola de soporte correspondiente.

**3. Diseño de la solución**

* Recepción del correo**:** Deskify configurará un servicio que se conectará a una bandeja de entrada específica (usando protocolos como IMAP o SMTP).
* Procesamiento del contenido**:** El sistema analizará el correo para extraer:
  + **Asunto:** Será el título del ticket.
  + **Cuerpo del mensaje:** Conformará la descripción del problema.
  + **Remitente:** Para identificar o al usuario en la base de datos si está registrado.
* Creación del ticket**:** Una vez procesada la información, Deskify generará un ticket con los datos del correo, asignándolo a la categoría o técnico adecuado basado en palabras clave o reglas predefinidas.

**4. Implementación técnica**

* Conexión con el servidor de correo**:** Usar una librería como *JavaMail* para conectarse al servidor y leer los correos entrantes.
* Sincronización con la base de datos**:** Guardar el ticket y vincularlo al usuario remitente.
* Notificaciones**:** Enviar una respuesta automática al usuario confirmando que su ticket ha sido recibido.

**5. Ejemplo práctico**

Un cliente envía un correo a soporte@deskify.com con el asunto: "No puedo acceder a mi cuenta" y describe su problema en el cuerpo del mensaje.

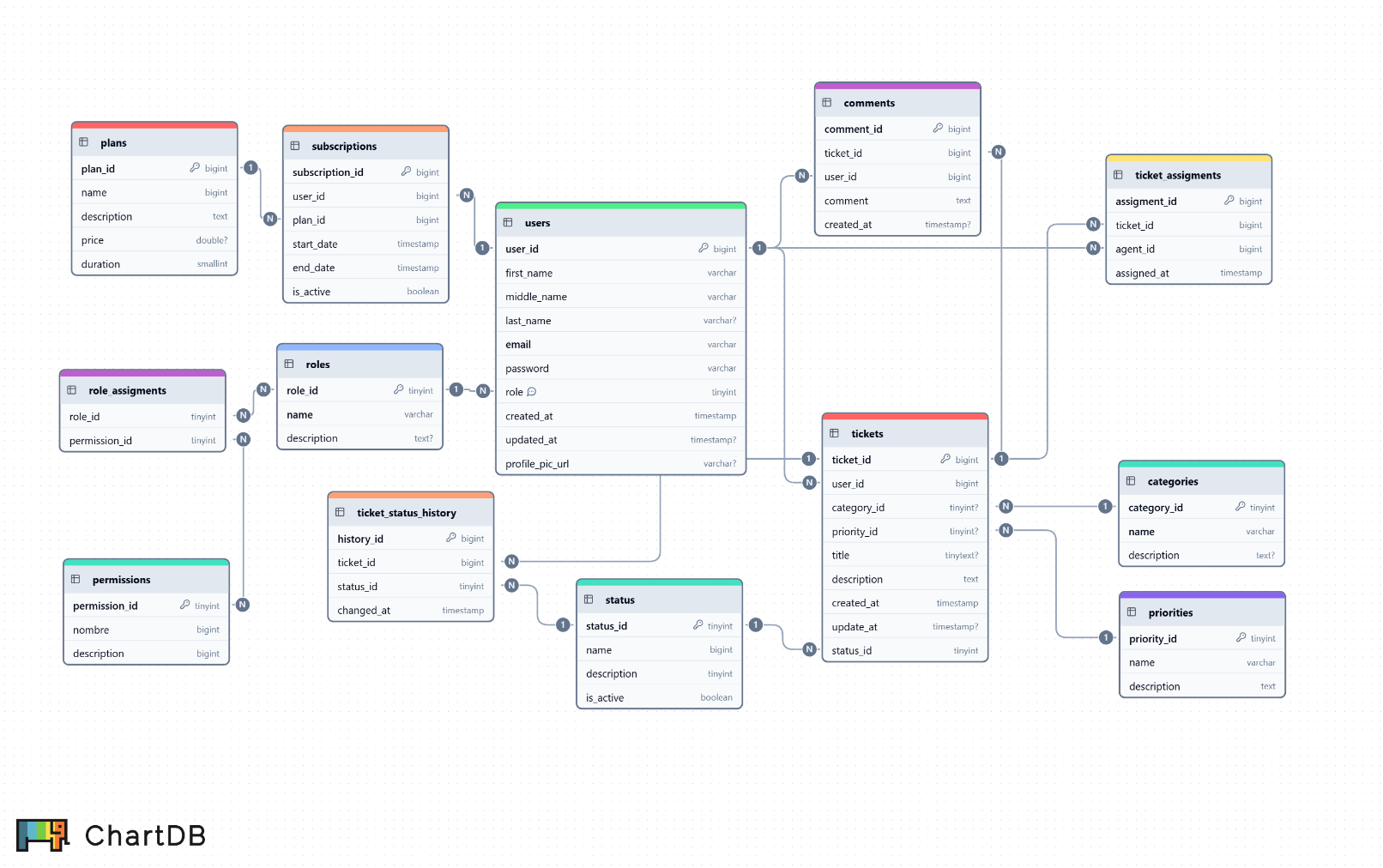
* Deskify recibe el correo y genera un ticket con el asunto "No puedo acceder a mi cuenta".
* La descripción del problema se agrega al cuerpo del ticket.
* El cliente recibe una notificación automática con el número del ticket y un enlace para realizar el seguimiento.

**6. Resultados esperados**

* Mejora en la experiencia del usuario**:** Los clientes pueden enviar tickets sin necesidad de iniciar sesión en la plataforma.
* Mayor eficiencia**:** Reducción del tiempo de creación de tickets por parte de los técnicos.
* Centralización de datos**:** Todos los tickets generados por correo electrónico quedan integrados con el resto del sistema, facilitando su seguimiento.

# Diagramas

## Base de datos

En primer lugar, se presenta el diagrama de la base de datos. Este diagrama representa la estructura de la base de datos diseñada para Deskify. El objetivo de esta base de datos es organizar la información relacionada con los usuarios, los roles que desempeñan dentro del sistema, los tickets de soporte generados, su estado y evolución, así como los planes de suscripción asociados a cada usuario.

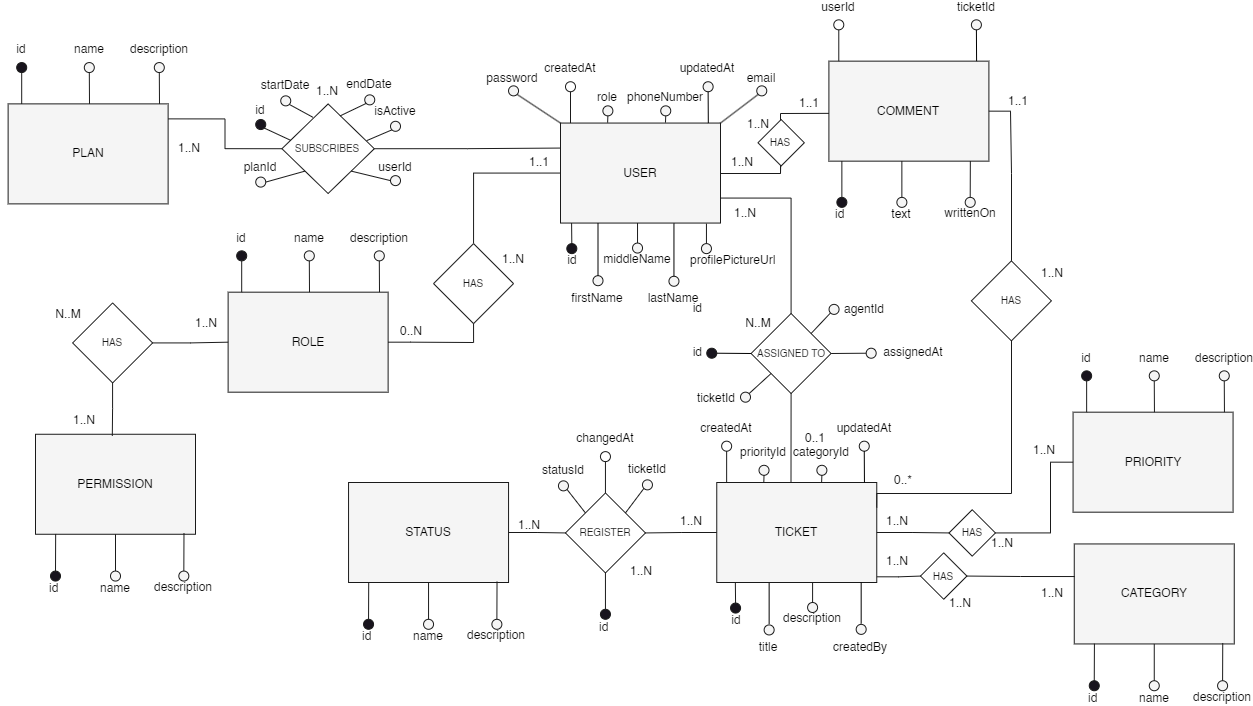
Para comprender en profundidad la estructura de la base de datos de Deskify, es necesario analizar cómo se organizan las distintas entidades que la componen y las relaciones que existen entre ellas. Esta base de datos ha sido diseñada siguiendo principios de normalización y claridad estructural, con el objetivo de garantizar un almacenamiento eficiente, coherente y fácilmente escalable.

A continuación, se detallan las principales tablas del sistema, su propósito dentro del modelo de datos y cómo interactúan entre sí para dar soporte a las funcionalidades clave de la plataforma:

* Usuario: Esta tabla almacena la información básica de los usuarios registrados, como nombre, correo electrónico, contraseña, rol, foto y fechas de creación y actualización.
* Rol: Define los diferentes tipos de roles en la plataforma (admin, supervisor, manager, agente, usuario). Un usuario puede tener un único rol asignado.
* Ticket: Representa las solicitudes de soporte realizadas por los usuarios. Cada ticket tiene un título, descripción, fecha de creación, estado, prioridad y está asociado a un usuario.
* Estado: Tabla que define los posibles estados de un ticket (abierto, en proceso, cerrado, esperando respuesta, etc.).
* Suscripción: Permite llevar un historial de los planes que ha tenido un usuario. Cada entrada incluye la fecha de inicio, fecha de finalización, y el tipo de plan.
* Plan: Define los diferentes tipos de planes disponibles (gratuito, profesional, empresarial, etc.) con sus respectivas características.

Con el objetivo de garantizar la escalabilidad del sistema y facilitar la gestión de posibles cambios futuros, se optó por no utilizar enumeraciones en campos como estado, prioridad, categoría, planes, roles y permisos. En su lugar, se implementaron tablas específicas para cada uno de estos elementos. Esta decisión permite una mayor flexibilidad a la hora de añadir, modificar o eliminar valores sin necesidad de modificar el código fuente, lo que favorece el mantenimiento y evolución del sistema a largo plazo.

## Entidad-Relación



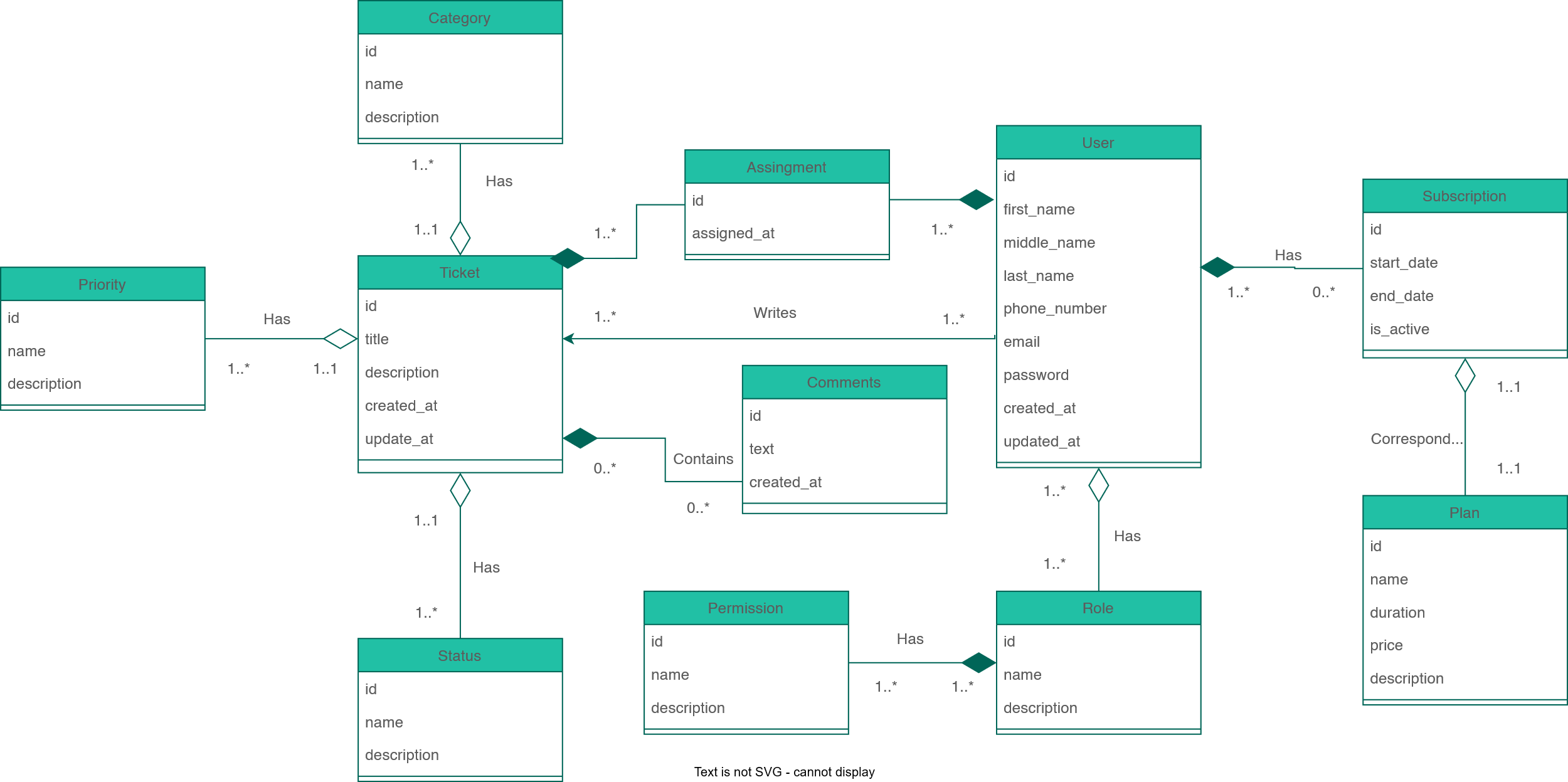
El diagrama entidad-relación de Deskify representa de forma clara y estructurada el modelo lógico de datos de la aplicación, reflejando cómo se organiza y relaciona la información dentro del sistema. Este diagrama ha sido diseñado siguiendo criterios de normalización, escalabilidad y separación de responsabilidades, con el objetivo de construir una base de datos robusta y fácilmente mantenible.

Entre las principales entidades se encuentran:

* Usuario: Entidad central del sistema. Cada usuario posee atributos como nombre, correo, contraseña, foto de perfil y fecha de registro. Está relacionada con otras entidades clave como Rol, Suscripción y Tickets.
* Rol: Define el tipo de acceso o privilegios que tiene un usuario dentro del sistema. Un usuario puede tener un único rol asignado, entre los que se incluyen *Admin*, *Supervisor*, *Manager*, *Agente* y *Cliente*.
* Ticket: Es la representación de una incidencia o solicitud de soporte generada por un usuario. Incluye información como título, descripción, fecha de creación, estado y prioridad. Cada ticket puede ser visualizado por varios agentes, pero únicamente tiene un creador y puede estar asignado a un único agente a través de la entidad intermedia Assignment.
* Assignment (Asignación): Entidad intermedia que permite vincular un ticket con el agente asignado, añadiendo información relevante como la fecha de asignación o el historial de responsables, si se desea extender en el futuro.
* Estado, Categoría y Prioridad: Estas entidades permiten una gestión flexible de los tickets, definiendo su situación actual (abierto, en proceso, cerrado…), su nivel de urgencia y su categoría. Se optó por externalizar estos valores en tablas independientes en lugar de utilizar enumeraciones, facilitando su gestión dinámica sin necesidad de modificar el código fuente.
* Suscripción: Permite llevar un control histórico de los planes contratados por cada usuario, incluyendo fecha de inicio, fin y tipo de plan.
* Plan: Define las características de cada nivel de servicio (gratuito, profesional, empresarial…), lo cual permite personalizar la experiencia de usuario según su nivel de suscripción.

Este modelo facilita la evolución de la aplicación, permitiendo añadir nuevos roles, estados, prioridades o planes sin alterar la estructura existente. Además, garantiza la integridad referencial mediante relaciones bien definidas y asegura un acceso coherente y eficiente a los datos de la plataforma.

## Clases

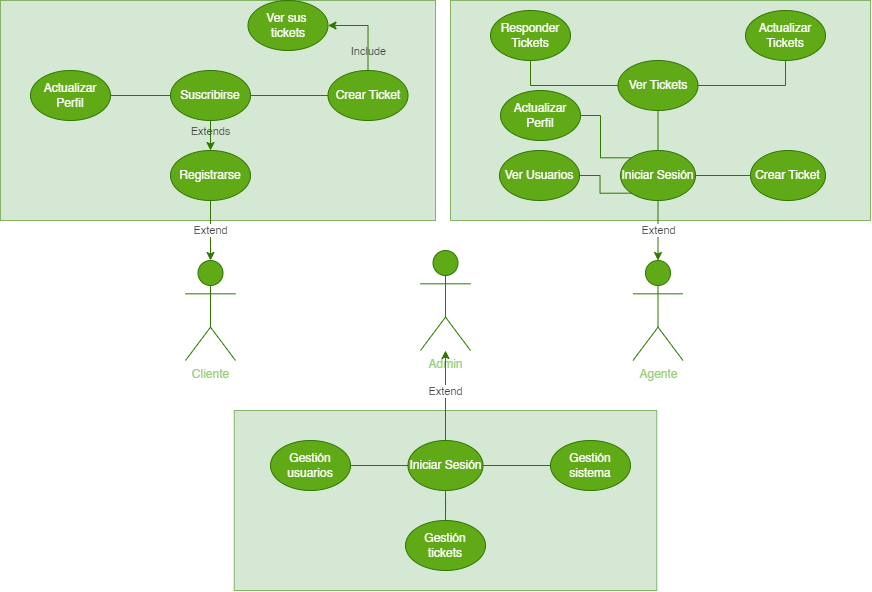


El diagrama de clases de Deskify ha sido diseñado para representar la estructura orientada a objetos del sistema, permitiendo una gestión clara de los datos y facilitando la interacción entre las distintas entidades. A continuación, se detallan las principales clases que componen la aplicación, su propósito y las relaciones entre ellas.

1. Usuario: La clase User almacena la información personal de los usuarios registrados en la plataforma, como su nombre, apellido, correo electrónico, número de teléfono, foto de perfil, entre otros. Además, gestiona la relación con los roles y suscripciones. Cada usuario está asociado a un único rol y puede tener una o varias suscripciones a planes.
2. Rol: La clase Role define los distintos tipos de roles disponibles en el sistema, tales como admin, supervisor, manager, agente o usuario. Cada rol tiene permisos específicos asociados, y los usuarios pueden tener un único rol asignado.
3. Permiso: La clase Permission gestiona los permisos asociados a cada rol. Los permisos son esenciales para el control de acceso y la administración de las funcionalidades dentro de la plataforma.
4. Asignación: La clase Assignment establece una relación entre un Ticket y un Usuario (en este caso, el agente que está trabajando en el ticket). Esta clase permite gestionar las asignaciones de tickets a lo largo de su ciclo de vida, reflejando los momentos en que un ticket es asignado a un usuario.
5. Ticket: La clase Ticket representa las solicitudes de soporte que los usuarios crean dentro del sistema. Un ticket tiene atributos como el título, la descripción, la fecha de creación, y la fecha de actualización. Además, cada ticket tiene un estado y una prioridad asignados, y puede estar relacionado con una categoría específica. El ticket también puede ser asociado a un usuario que lo haya creado y a un agente que lo esté gestionando.
6. Estado: La clase Status define los distintos estados que un ticket puede tener, como "Abierto", "En Proceso", "Cerrado", etc. Un ticket solo puede tener un único estado en un momento dado, pero múltiples tickets pueden compartir el mismo estado.
7. Prioridad: La clase Priority establece los niveles de prioridad que un ticket puede tener. Las prioridades ayudan a los agentes a determinar la urgencia de un ticket y gestionarlo en consecuencia.
8. Categoría: La clase Category organiza los tickets en categorías específicas, como "Problemas de cuenta", "Problemas técnicos", etc. Esta clasificación permite una gestión más eficiente de los tickets, asignándolos a las áreas correspondientes.
9. Plan: La clase Plan define los distintos planes de suscripción que los usuarios pueden adquirir, como "Gratuito", "Profesional", "Empresarial", etc. Cada plan tiene un precio y una duración específica.
10. Suscripción: La clase Subscription gestiona las suscripciones de los usuarios a los planes. Almacena las fechas de inicio y finalización de cada suscripción y vincula a un usuario con un plan específico.

Relaciones entre las Clases:

* Usuario y Rol: Un usuario tiene un único rol (relación uno a uno). Esta relación es fundamental para el control de acceso en el sistema.
* Usuario y Suscripción: Un usuario puede tener varias suscripciones a lo largo del tiempo, lo que implica una relación uno a muchos entre User y Subscription.
* Usuario y Ticket**:** Un usuario puede crear múltiples tickets, pero también puede ser asignado a un ticket a través de la clase intermedia Assignment. La relación entre User y Ticket es indirecta y se gestiona mediante la clase Assignment**.**
* Ticket y Estado: Un ticket tiene un único estado en cualquier momento, pero varios tickets pueden compartir el mismo estado. Es una relación de muchos a uno entre Ticket y Status.
* Ticket y Prioridad: Un ticket tiene un único nivel de prioridad, pero un nivel de prioridad puede estar asociado a múltiples tickets. La relación es de muchos a uno entre Ticket y Priority.
* Ticket y Categoría: Un ticket puede estar asociado a una categoría específica, y una categoría puede contener varios tickets. La relación es de muchos a uno entre Ticket y Category.
* Suscripción y Plan: Cada suscripción está vinculada a un plan específico, y un plan puede estar asociado a múltiples suscripciones. La relación es de uno a muchos entre Plan y Subscription.



## Casos de Uso

El diagrama de casos de uso de **Deskify** representa gráficamente las funcionalidades principales del sistema y cómo interactúan los distintos actores con la plataforma. Se identifican tres principales que reflejan los diferentes perfiles de usuario que pueden operar dentro del sistema: Cliente, Agente y Administrador. Cada uno de ellos accede a funcionalidades específicas, en función de su rol, con el fin de garantizar una experiencia personalizada y acorde a sus responsabilidades.

Cliente**:** Es el usuario final que necesita soporte técnico y por tanto inicia la mayoría de las solicitudes dentro del sistema.

* Para acceder a Deskify, primero debe registrarse en la plataforma.
* Una vez registrado, debe suscribirse a un plan, ya que solo los usuarios con una suscripción activa pueden hacer uso de los servicios.
* Tras completar la suscripción, podrá crear tickets de incidencia cuando necesite asistencia.
* También podrá consultar el estado de sus tickets, así como visualizar el histórico completo de sus solicitudes anteriores.
* Además, tiene la opción de editar su perfil personal, modificando datos como su nombre, teléfono, contraseña o foto de perfil.

Agente**:** Es el encargado de dar soporte técnico y resolver las incidencias creadas por los clientes.

* Para comenzar a trabajar, debe iniciar sesión en la plataforma.
* Una vez dentro, puede visualizar y gestionar los tickets asignados, incluyendo actualizaciones de estado, respuesta a comentarios y resolución.
* Puede crear nuevos tickets en nombre de los usuarios si es necesario.
* Tiene acceso a un módulo para consultar el perfil de los usuarios con el fin de comprender mejor el contexto de cada solicitud.
* Asimismo, puede modificar su propio perfil como cualquier otro usuario.

Administrador: Es el actor con mayores privilegios en la plataforma y su función es garantizar el correcto funcionamiento del sistema en su conjunto.

* Al igual que el resto de usuarios, debe autenticarse mediante inicio de sesión.
* Una vez dentro, dispone de acceso completo para gestionar usuarios, lo que incluye la creación, modificación, desactivación o eliminación de cuentas.
* También puede gestionar el sistema, configurando parámetros clave como los planes de suscripción, permisos, roles o categorías de ticket.
* Tiene facultades para supervisar todos los tickets del sistema, sin limitación de acceso, asegurándose de que se atiendan correctamente y en tiempo.

# Diseño

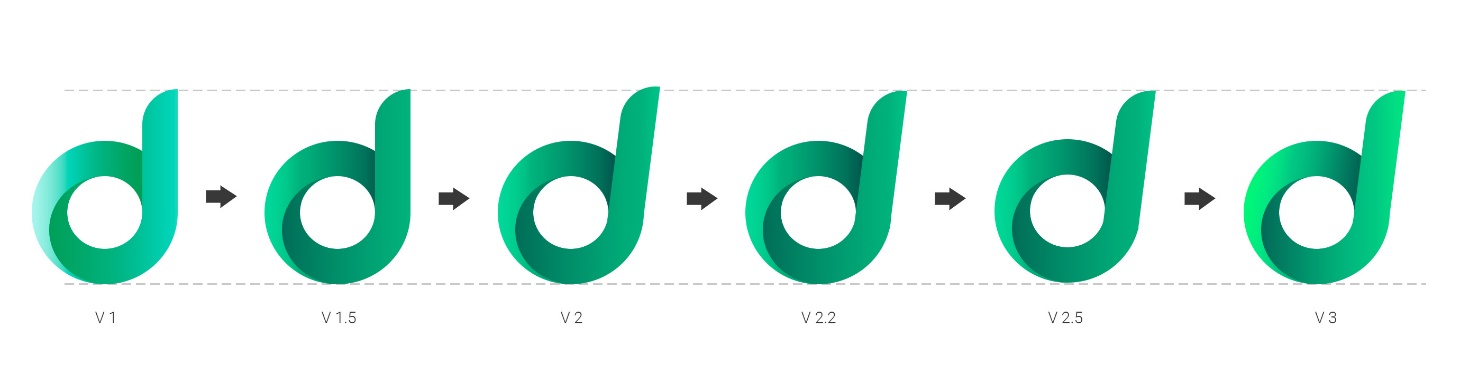
El proyecto se inició con la concepción del nombre de la marca, un elemento fundamental para definir la identidad del software. Esta tarea resultó desafiante, ya que se requería un nombre que fuera contemporáneo, accesible y cautivador, capaz de transmitir profesionalismo y facilidad de uso. Tras una exhaustiva sesión de *brainstorming*, se obtuvieron tres opciones finalistas: Deskify, EasyDesk y PriorityDesk. Se observó una tendencia hacia términos relacionados con escritorios y soporte, lo que refleja el enfoque en la gestión de tickets e incidencias.

Tras una exhaustiva deliberación, se concluyó que Deskify sobresalía como la opción más adecuada. La elección de este nombre no fue arbitraria, sino que se fundamentó en una serie de consideraciones semánticas y de usabilidad. En primer lugar, el término «desk» evoca los conceptos de soporte y organización, mientras que el sufijo moderno «-ify» sugiere transformación y dinamismo. Este nombre representa de manera precisa la visión del proyecto, que es la de transformar la experiencia del *helpdesk* en algo ágil, intuitivo y eficiente. Además, su brevedad y sonoridad lo hacen fácil de recordar y atractivo para los usuarios.

Una vez seleccionado el nombre, el próximo paso del proceso de desarrollo de la marca consistió en la creación de un logotipo que reflejara su esencia y cumpliera con los requisitos clave de simplicidad, versatilidad y adaptabilidad a colores vibrantes y tonos monocromáticos planos. Tras evaluar diversas opciones, se tomó la decisión de centrar el logotipo en la inicial de «deskify», convirtiéndola en un símbolo distintivo y memorable.

Posteriormente, se procedió a la definición de los colores que conformarían la identidad visual de la marca. Tomando en consideración el enfoque tecnológico del proyecto, se determinó que los tonos más apropiados serían el verde y el azul, ya que ambos colores suelen estar asociados con placas de circuitos y tecnología avanzada, lo que refuerza el carácter innovador de Deskify.

Durante el proceso de diseño del logotipo, mientras se examinaban bocetos y se evaluaban propuestas, emergió un elemento que se convirtió en un aspecto esencial de la composición: el logotipo debía exhibir una ligera inclinación. Este detalle no solo contribuiría a generar dinamismo, sino que también simbolizaría velocidad y eficacia, valores fundamentales de Deskify en su misión de resolver incidencias de manera rápida y efectiva.

A continuación, se presenta un gráfico que ilustra la evolución del logotipo desde sus concepciones iniciales hasta su versión actual, mostrando cómo cada iteración fue afinando la identidad visual de Deskify para alinearla con los principios mencionados.

## Guía de estilos

Tras establecer el logotipo de la marca, el siguiente paso consistió en definir su identidad visual a través de una guía de estilos. Para ello, se tomó como punto de partida la paleta de colores original, identificando y documentando los códigos de color que conforman la base estética de la marca.

Tal y como se muestra en la guía, se pueden distinguir claramente las diferentes secciones que la componen. En primer lugar, se presenta el menú de navegación de la web, diseñado con un efecto de cristal translúcido que aporta un estilo minimalista y moderno a la interfaz.

A continuación, se expone la paleta de colores definida para el sitio separada por tonalidades y acompañada de una breve explicación sobre la elección de los tonos utilizados. A la derecha, se encuentra el logotipo principal previamente mencionado, junto con sus variantes y tamaños establecidos. Justo debajo, se muestra nuevamente el logotipo, esta vez acompañado del nombre de la marca, también con sus diferentes versiones y dimensiones correspondientes.

Otro apartado destacable es el de tipografía y tamaños de fuente, donde se especifica la fuente principal utilizada en la web, así como los distintos estilos tipográficos que se aplican. Se incluyen los encabezados desde H1 hasta H5, seis niveles de peso tipográfico para ofrecer versatilidad en el diseño, y una muestra representativa de los párrafos tal y como se visualizarán en la página. Además, se incorpora una demostración del elemento <hr>, utilizado como separador visual a lo largo del sitio.

También se incluye el apartado correspondiente a los botones, en el que se detallan los cuatro tipos de botones presentes en la web o aplicación. Cada uno de ellos se presenta con una demostración visual de su comportamiento, tanto en su estado normal como al pasar el cursor por encima (hover). Además, se especifican sus características particulares, como el estilo, color, tamaño y sombra.

Por último, se encuentra la sección dedicada a los formularios, un elemento fundamental en la interacción del usuario con la plataforma. En esta área se incluyen tanto el formulario de registro como el de inicio de sesión, diseñados para ser intuitivos, accesibles y coherentes con la identidad visual del proyecto.

El formulario de registro requiere la cumplimentación de los siguientes campos: nombre, correo electrónico y contraseña. Se ha puesto especial atención en la validación de datos, por lo que también se muestra una demostración visual del comportamiento del formulario ante errores, como puede ser el caso de campos incompletos, indicando al usuario de forma clara qué debe corregir.

## MoodBoard

El *moodboard* presentado a continuación sirve como punto de partida visual e inspiración para la creación de la identidad estética de la plataforma. Junto con la guía de estilos, establece las bases conceptuales sobre las que se construirá la estructura y el diseño de la futura web. A través de una cuidadosa selección de imágenes, colores, tipografías y elementos visuales, se busca transmitir la esencia minimalista, moderna y funcional de Deskify, garantizando una experiencia coherente, atractiva y centrada en el usuario desde sus primeras fases de desarrollo.

El *moodboard* de Deskify ha sido diseñado para transmitir de manera coherente la identidad y el estilo que se desean transmitir a través de la plataforma. En la parte superior se encuentran tres degradados principales que se han utilizado de forma recurrente a lo largo del proyecto y actúan como un elemento distintivo de la marca. Estos degradados no solo aportan dinamismo visual, sino que también refuerzan la identidad moderna y tecnológica de la página web.

La base del *moodboard* está compuesta por una serie de seis imágenes dispuestas verticalmente, cada una acompañada de un degradado que armoniza con la paleta cromática del proyecto, la cual se muestra claramente a la derecha mediante seis tonos principales. Esta combinación visual refleja la intención de crear una estética limpia, coherente y unificada. En el centro, se sitúa el logotipo de Deskify, actuando como sello de identidad y punto focal del *moodboard*, consolidando así la conexión entre la inspiración visual y la marca.

## Wireframes

Los *wireframes* presentados a continuación constituyen la primera aproximación visual a la estructura y distribución de contenidos de *Deskify*. Su principal objetivo es definir de forma clara y ordenada la jerarquía de la información, así como la disposición de los elementos clave en la interfaz, sin entrar aún en detalles de diseño visual.

Debido a la cantidad de *wireframes* desarrollados para este proyecto, y con el fin de mantener una presentación clara y organizada, se ha optado por separarlos en dos secciones diferenciadas: una correspondiente a la versión web y otra a la versión aplicación.

### Página Web

Como se puede observar, estos *wireframes* corresponden a la *Landing Page* del proyecto, la cual será desarrollada utilizando el *framework* Astro, conocido por su eficiencia y enfoque en el rendimiento. Esta página tendrá como objetivo principal presentar Deskify al usuario final de manera atractiva, clara y funcional, destacando sus características, ventajas y valor diferencial.

A través de estos *wireframes* se define la estructura inicial de la *landing*, incluyendo secciones como la presentación del producto, testimonios, llamada a la acción y otros elementos estratégicos que guiarán al usuario de forma intuitiva. Este diseño actúa como una guía esencial para la posterior implementación técnica, asegurando coherencia visual y experiencia de usuario desde el primer contacto con la plataforma.

La página está claramente organizada en cinco secciones principales, que se encuentran definidas en el menú de navegación. La página principal sigue un *Hero Layout*, en el que se incluirá un título de bienvenida acompañado de una breve descripción de la marca. Junto a ello, se destacará un botón de llamada a la acción (*CTA*) que invitará al usuario a registrarse como cliente. Todo esto estará complementado con un fondo visualmente atractivo que reflejará los colores y valores esenciales de la compañía.

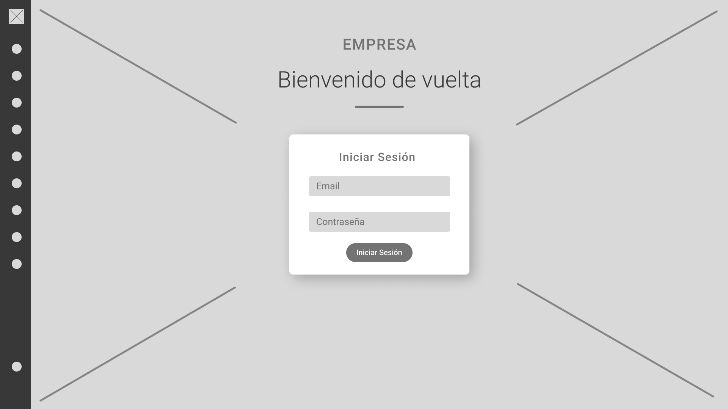
A continuación, se encuentra la página *About* o Sobre nosotros, que inicia con un *Hero Layout*, replicando el mismo efecto visual utilizado en la página principal. Luego, la página sigue un *Z Layout*, en el que las imágenes y textos descriptivos se distribuyen de manera estratégica para guiar al usuario a través de los valores y principios fundamentales de la marca, ofreciendo una explicación detallada de su misión y visión. Terminando con un pequeño apartado en el que posteriormente estará dedicado al usuario.

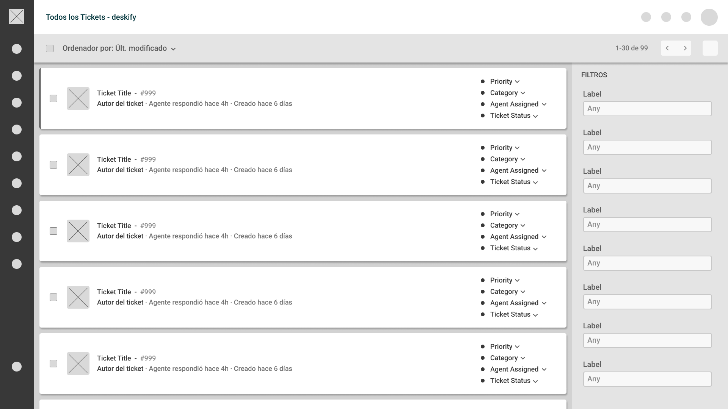
En tercer lugar, se encuentra la página destinada a los planes de suscripción, la cual adoptará una estructura de *Grid of Cards*. Esta estará formada por tres tarjetas que representan los distintos planes disponibles, permitiendo que el usuario los distinga con claridad. Cada tarjeta incluirá una descripción detallada de las ventajas de cada plan, junto con los precios correspondientes.

Por otro lado, encontramos la página dedicada a la plataforma, que sigue una estructura similar a la de la página *About*, pero en este caso utiliza un *F Layout*. Esta página se compone de una descripción detallada del funcionamiento de la aplicación, acompañada de una imagen que ilustra cómo interactúa el usuario con la plataforma.

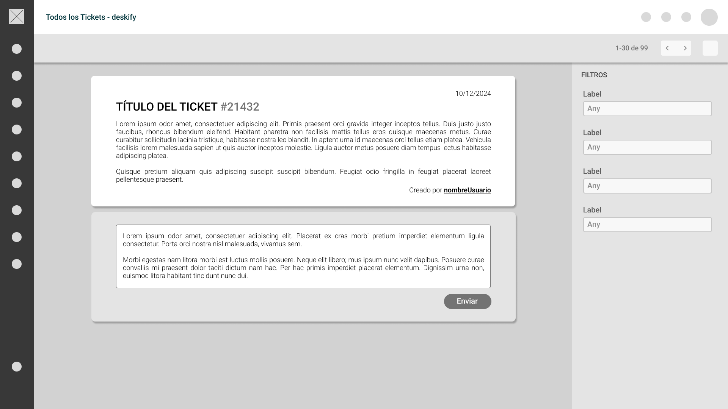
En último lugar, se encuentra la página dedicada a las opiniones de los clientes que utilizan la plataforma como su principal servicio. La estructura sigue un *Grid of Cards*, con seis tarjetas que presentan las opiniones, junto con el nombre del cliente y la valoración de nuestro servicio. Al final de la página, se incluye un texto descriptivo que destaca la importancia de los testimonios y, para motivar a posibles clientes, un botón de *CTA* (Call to Action) invitando a ponerse en contacto o probar el servicio.

### Aplicación Web

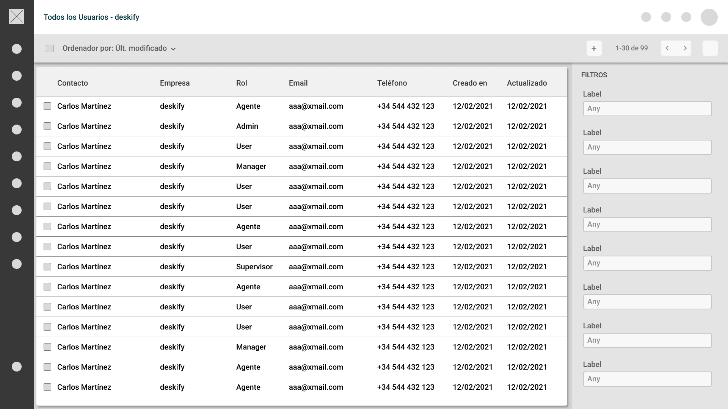
En primer lugar, al acceder al enlace que dirige a la aplicación, el usuario se encuentra con un formulario de inicio de sesión, necesario para ingresar en la plataforma. Es importante destacar que, para poder acceder, el usuario debe estar previamente registrado y contar con una suscripción activa, ya que la aplicación está destinada exclusivamente a clientes del servicio.



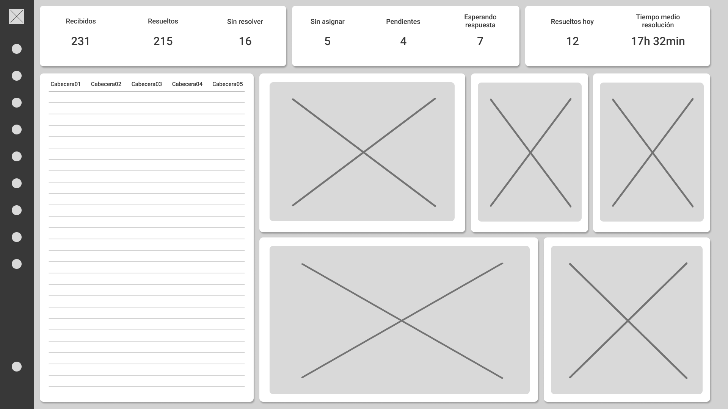
Una vez iniciada la sesión, el usuario será redirigido al listado de tickets. En esta sección, los agentes podrán realizar un seguimiento de cada caso, filtrando por estado, categoría o nivel de prioridad. Además, se mostrará qué agente tiene asignado cada ticket, junto con una breve descripción del problema.

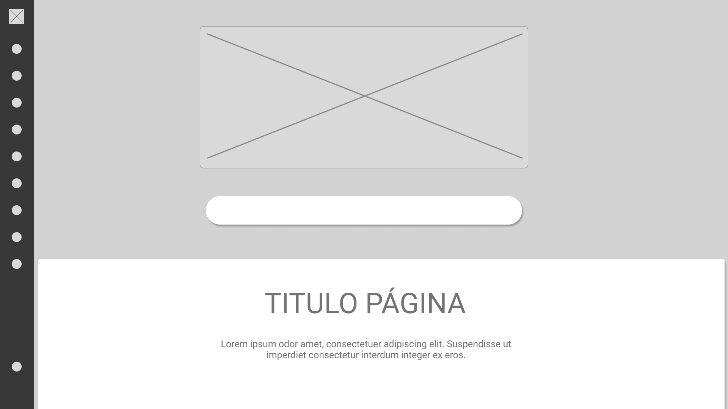
Cuando el agente accede a una incidencia, se le redirige a una nueva vista del ticket. En esta página, podrá añadir comentarios, así como modificar la prioridad, la categoría y el estado del ticket a través del menú situado en el lateral derecho. Todas estas acciones quedan registradas de forma automática y, al final de la página, se muestra un historial completo del ticket, permitiendo un seguimiento claro y organizado de la incidencia.

 Esta será la vista destinada al cliente de la empresa, quien, al presentar una incidencia, podrá comunicarse directamente con el técnico asignado a través de un sistema de mensajes. En esta vista, el menú lateral desaparece, ya que el cliente no cuenta con el rol de agente y, por tanto, no tiene acceso a funcionalidades avanzadas de gestión.

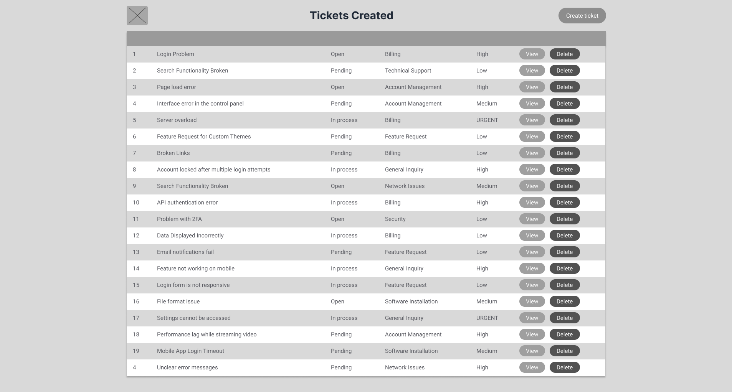
 En el apartado de usuarios, aquellos con los permisos adecuados podrán acceder a la información detallada de todos los clientes y agentes registrados en la plataforma. Para facilitar la navegación y gestión de estos datos, se incluye un menú lateral que permite filtrar la información por diferentes criterios, agilizando así la búsqueda y mejorando la experiencia de uso.

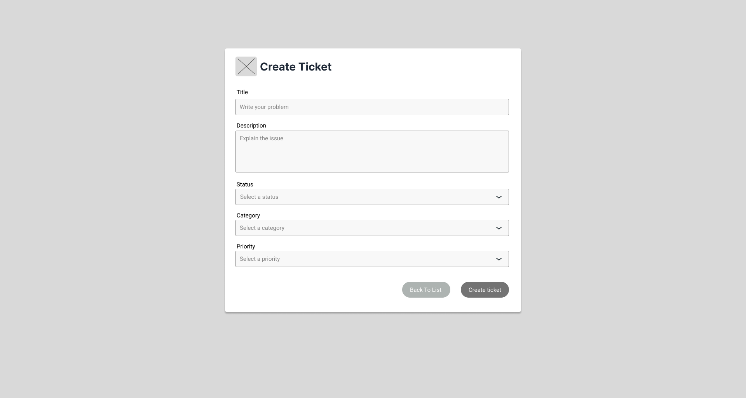
Al hacer clic sobre un usuario, se accede a una página individual dedicada exclusivamente a su perfil. En ella, se muestra toda su información personal. A su lado, se presenta un panel detallado con estadísticas relacionadas con su actividad dentro de la plataforma, ofreciendo una visión general del rendimiento y uso del sistema por parte del usuario.

Dentro del menú lateral de navegación, también se incluirá un enlace al *Dashboard*, el cual funcionará como un panel de control con estadísticas globales de la aplicación. Este apartado ofrecerá una visión general del rendimiento del sistema, incluyendo métricas como el número total de tickets creados y resueltos, el tiempo medio de respuesta, así como un ranking de los agentes con mayor número de incidencias solucionadas.

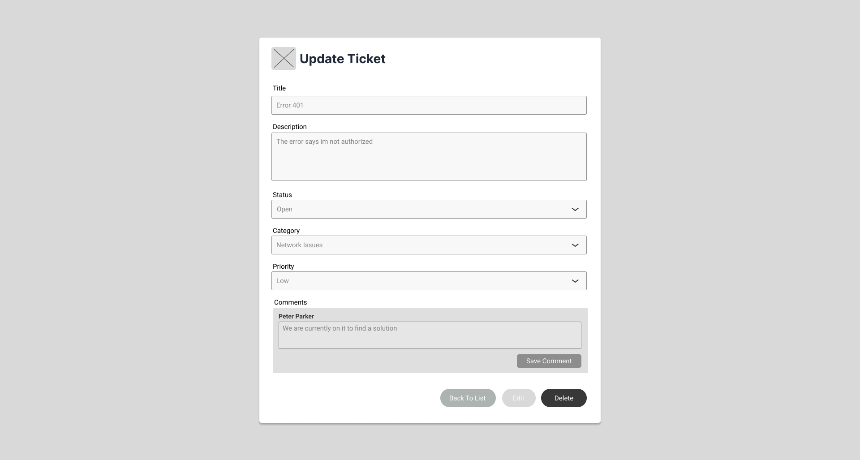
Por último, se encuentra la sección de *FAQs*, esta ofrece soporte inmediato a los clientes que necesiten resolver dudas comunes sobre el funcionamiento de la aplicación. En este apartado se recogen las incidencias más comunes junto con sus respectivas soluciones, redactadas de forma clara y accesible. Además, podrán contactar con el servicio de asistencia si no se resuelve el problema.

### Panel de Administrador

Además, se ha implementado un diseño para el panel de administrador. Como su propio nombre indica, este será un apartado desde el que los trabajadores con rol de administrador podrán gestionar los tickets, como editarlos o eliminarlos. Esta función no será accesible desde la propia aplicación. Por motivos de seguridad, esta función no está disponible.



El propio administrador podrá crear tickets desde este panel. Al hacer clic en el botón «Crear», se despliega esta página, donde podrá modificar el estado del ticket antes de crearlo. De este modo, los agentes podrán filtrarlo en la plataforma.



Por último, también podrán actualizar el contenido completo del tique. Incluso podrán cambiar los comentarios si tienen algún error o dato incorrecto. Para ello, deberán hacer clic en «editar» para activar los campos y, a continuación, guardarán los cambios. Esto quedará registrado en la base de datos.

## Mockups

Para comenzar esta sección, es importante señalar que los mockups presentados corresponden a una versión inicial de cómo se visualizará la web en su versión final. Esto significa que pueden estar sujetos a ligeros cambios a medida que avance el desarrollo del proyecto. Además, se incorpora un nuevo apartado dedicado a la gestión visual de errores, los cuales se mostrarán en pantalla según el tipo de fallo ocurrido, ya sea del lado del cliente o del servidor. Esta incorporación busca mejorar la experiencia del usuario y ofrecer una comunicación clara en caso de incidencias.

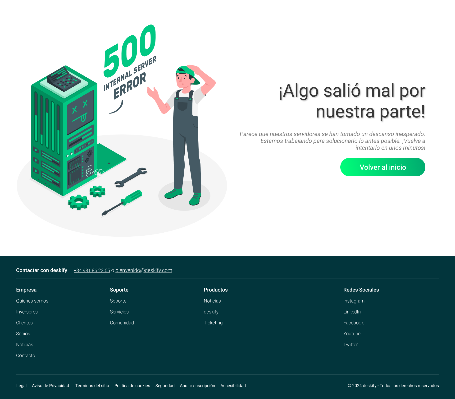
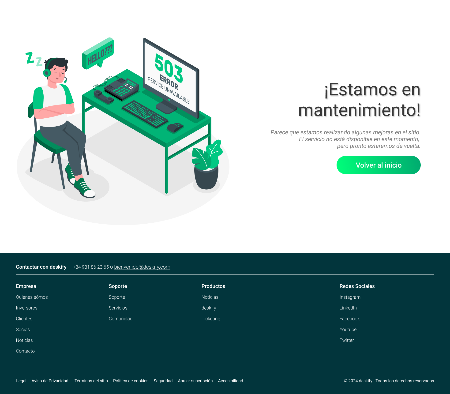
A continuación, se presentan las páginas personalizadas diseñadas para gestionar los distintos errores que pueden surgir durante la navegación en la plataforma. Cada una de ellas está compuesta por una imagen ilustrativa que representa visualmente el tipo de error, un título claro y una descripción que ayuda al usuario a entender qué ha ocurrido. Además, se incluye un botón que permite regresar fácilmente al inicio o volver a la página anterior, ofreciendo así una experiencia más amigable y guiada, incluso cuando algo no sale como se esperaba.

### Errores

#### Servidor

En este apartado se muestran las pantallas personalizadas dedicadas a los errores del lado del servidor más habituales: 500 (Error interno del servidor), 503 (Servicio no disponible) y 504 (Tiempo de espera agotado de la puerta de enlace). Cada una ha sido diseñada con una estética coherente con el resto del proyecto, ofreciendo una experiencia clara y comprensible al usuario incluso en situaciones donde el sistema no responde como debería.

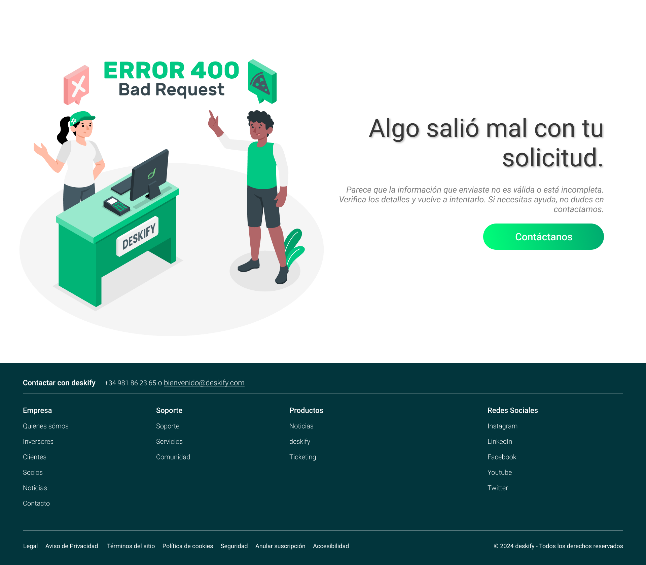
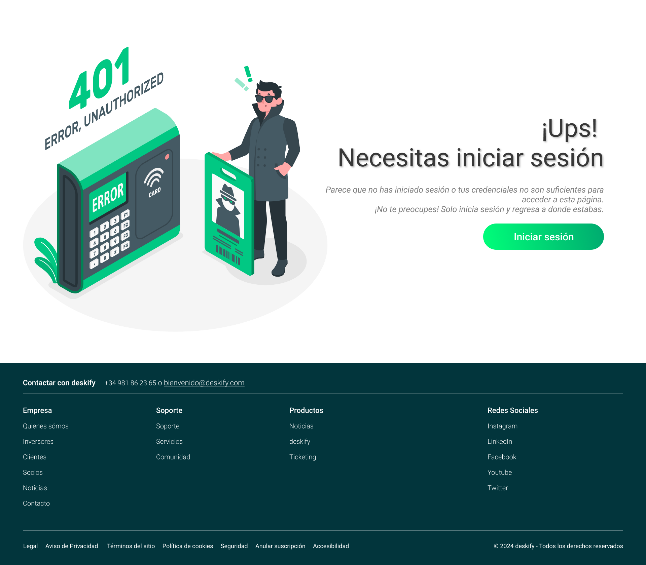
Estas páginas no solo informan con un lenguaje cercano y accesible, sino que también incorporan elementos visuales y un botón de navegación que facilita al usuario volver al inicio o retroceder. Aunque estos errores suelen deberse a problemas temporales, comunicar de forma adecuada lo que ocurre genera confianza y mejora la percepción de la plataforma durante incidencias técnicas.

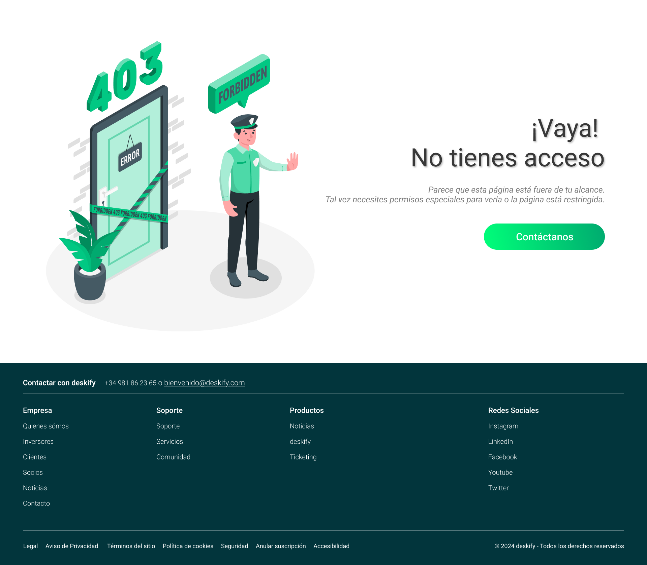
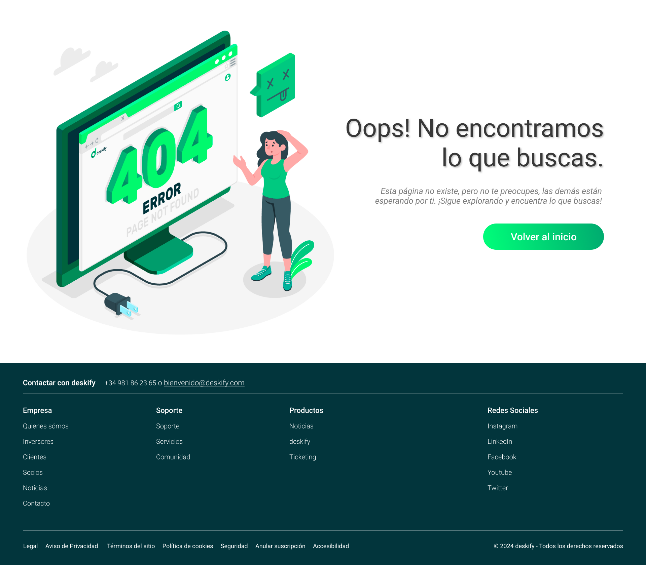


#### Cliente

Los errores de cliente más comunes como 400 (Petición incorrecta), 401 (No autorizado), 403 (Prohibido) y 404 (No encontrado) se han representado mediante pantallas personalizadas que combinan claridad y diseño amigable. Cada una incluye una ilustración explicativa, un título cercano y una descripción que informa al usuario de forma comprensible sobre lo que ha ocurrido.

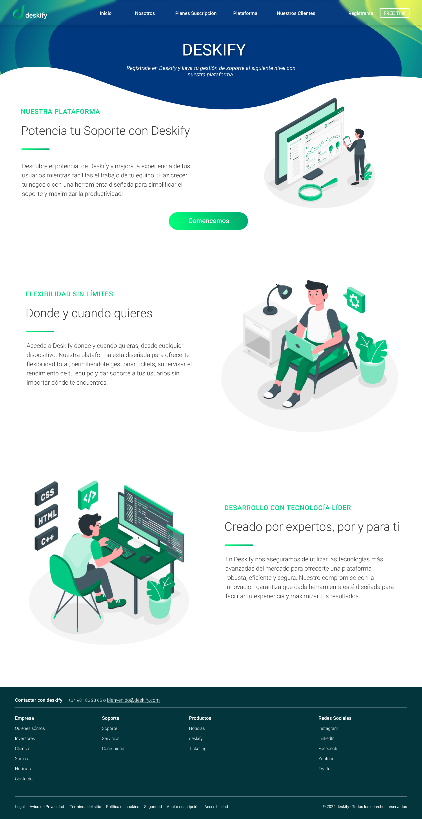
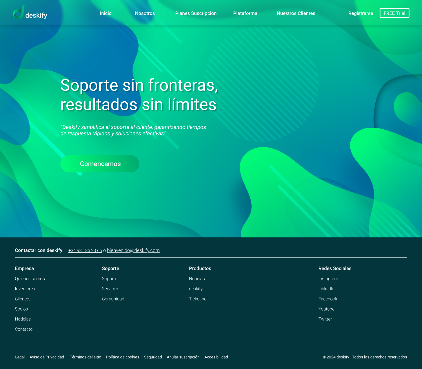
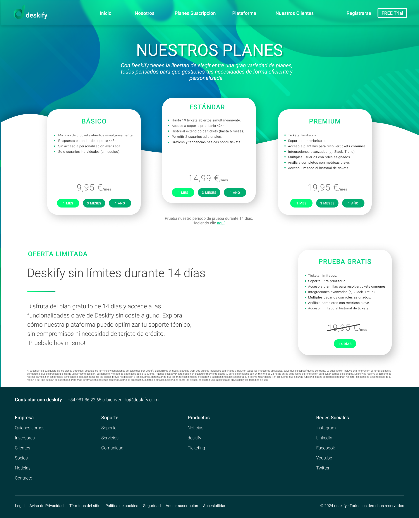
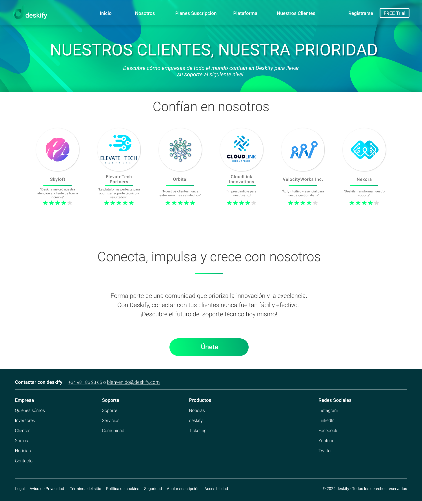
Estas páginas no solo buscan reducir la confusión, sino también ofrecer una experiencia coherente y accesible, incluso cuando algo sale mal. Para facilitar la navegación, todas ellas incorporan un botón que permite volver al inicio o regresar a la página anterior sin complicaciones.

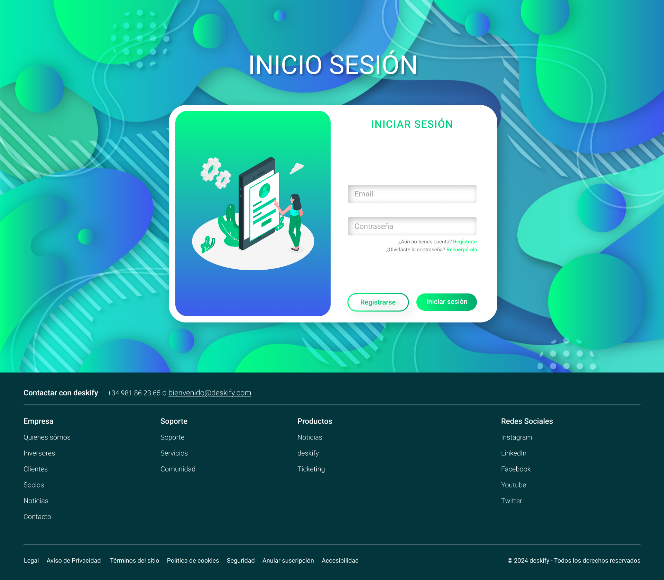
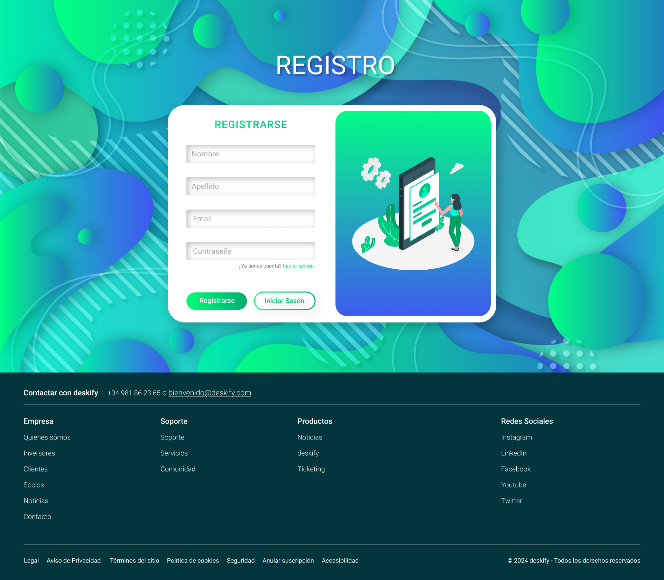




### Página Web

Los mockups presentados justo debajo son una versión más avanzada y visual del diseño de la plataforma web, desarrollados a partir de los *wireframes* definidos previamente. En esta etapa, se han aplicado los elementos visuales propuestos en el *moodboard* y la guía de estilos, incluyendo la paleta de colores, los degradados y la identidad visual de la marca. Gracias a esto, se puede apreciar cómo las estructuras planteadas en los diseños preliminares cobran vida con un diseño coherente, atractivo y fiel a la esencia de Deskify.





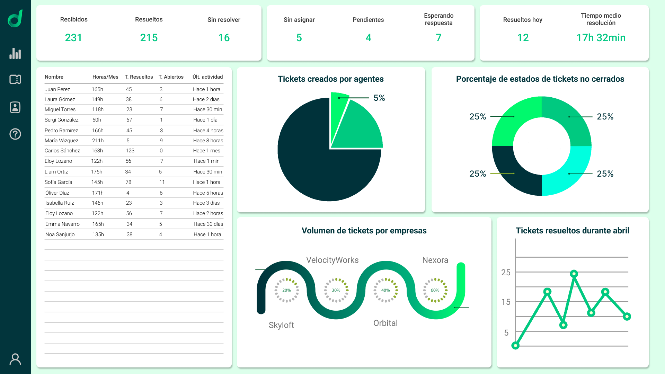
También se diseñaron los mockups correspondientes para los formularios de inicio de sesión y registro. Se optó por unificarlos en una única página para ofrecer una experiencia más cómoda e intuitiva al usuario. De esta forma, al seleccionar entre iniciar sesión o registrarse, la imagen se desplaza dinámicamente para dar paso al formulario correspondiente, manteniendo así una navegación fluida y una estética coherente.

### Aplicación Web

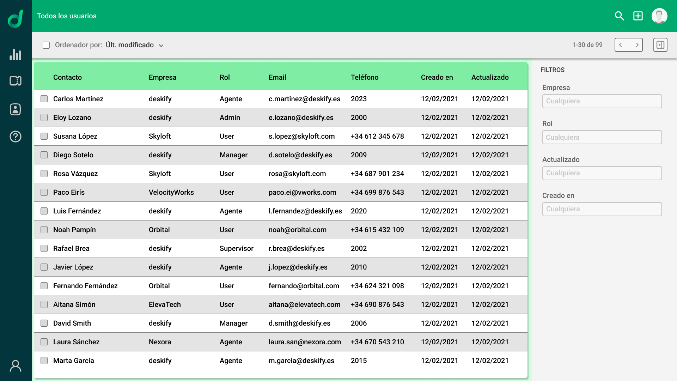
Los siguientes mockups representan la apariencia visual definitiva de la aplicación web de Deskify, basándose en la estructura ya definida en los *wireframes*. Cada pantalla ha sido diseñada teniendo en cuenta la paleta de colores seleccionada y el *moodboard* propuesto, garantizando coherencia visual y una experiencia de usuario fluida. Estas vistas reflejan cómo interactuarán los distintos perfiles dentro de la plataforma, desde el inicio de sesión hasta la gestión de tickets, usuarios y estadísticas, mostrando una interfaz moderna, intuitiva y adaptada a cada rol del sistema.



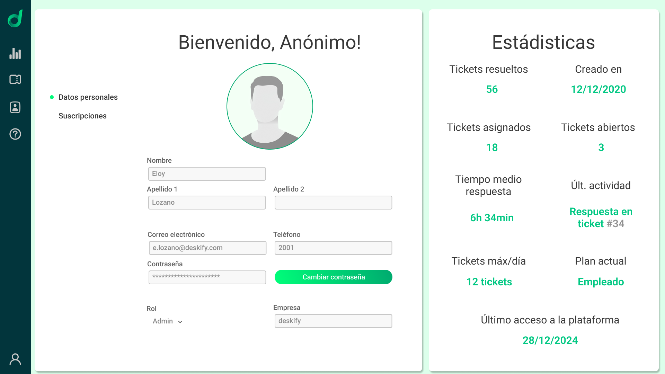
Como se puede observar, la página de inicio de sesión aplica los colores corporativos de forma coherente, incorporando un fondo minimalista que dirige la atención hacia el formulario principal y mantiene una estética equilibrada en toda la estructura.



En el apartado del *dashboard*, se incluyen gráficos ilustrativos que muestran la distribución de los tickets por categoría, así como estadísticas generales del sistema. Además, se destacan los agentes con sus métricas detalladas, ofreciendo una visión clara del rendimiento y la gestión dentro de la plataforma.



Esta vista detallada de los usuarios se presenta en forma de tabla, con un diseño de filas alternas (cebrado) que facilita la lectura. La cabecera de la tabla destaca en verde, siguiendo la identidad visual de Deskify, y permite identificar fácilmente cada columna de información.



Al hacer clic en un usuario, se redirigirá a una vista más extensa y detallada de su perfil. En esta nueva vista, se podrán ver estadísticas resaltadas en verde, que corresponden a su rendimiento dentro de la plataforma, junto con la foto de perfil de cada usuario, ofreciendo una visualización clara y completa de la información relevante.

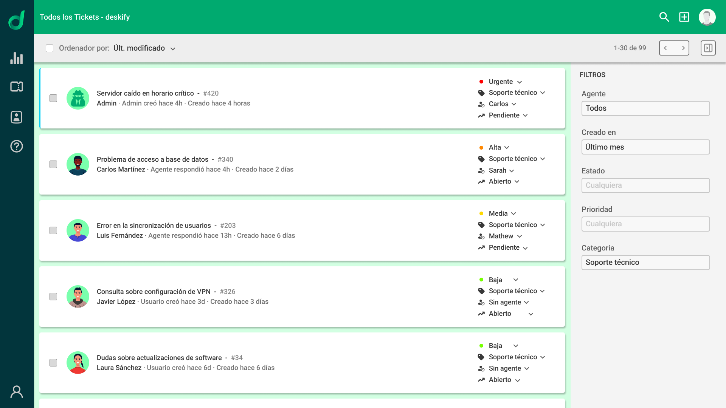
Ahora me gustaría hacer hincapié este apartado, ya que a la hora del diseño surgió una duda de sobre la estructura de diseño para mostrar los tickets, en tabla o lista con cartas. Ambas tienen ventajas y desventajas.

#### Vista en tabla

En la vista en tabla, se ofrece una presentación más clara y estructurada de la información de cada ticket, lo que facilita su lectura. Además, permite ordenar los tickets en orden ascendente o descendente, mejorando la organización.

* Pros: Lectura más sencilla, estructura definida, permite la ordenación.
* Contras: No permite modificar estados, prioridades ni categorías directamente desde el menú.

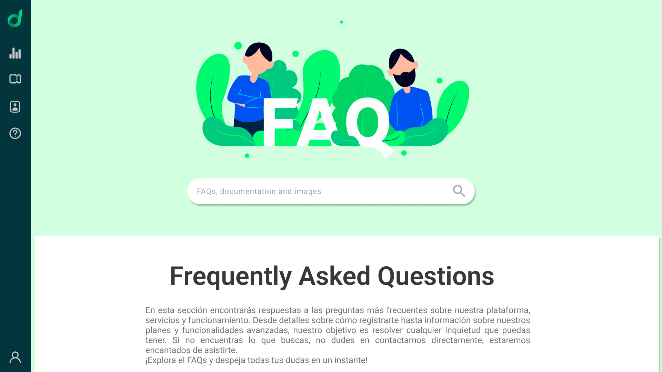
#### Vista en lista



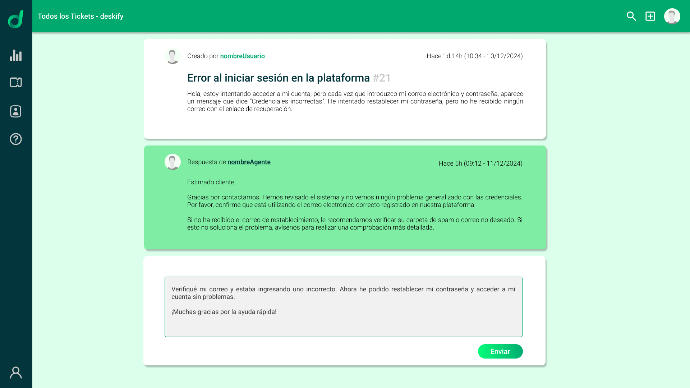
Esta vista facilita la identificación de cada incidencia y permite modificar las categorías, estados, prioridades y agentes directamente desde el menú.

* Pros: Permite la edición desde el menú, mejora la identificación de las incidencias.
* Contras: No permite la ordenación de los tickets.

Después de debatirlo con varios profesores y compañeros, se concluyó que la vista en tabla es la opción más adecuada, ya que facilita la ordenación de los tickets y ofrecerá una experiencia de uso más eficiente a largo plazo.



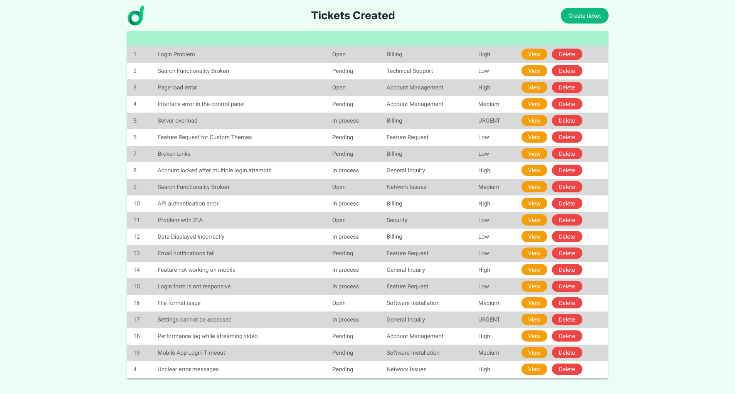
En la página de FAQs, se incorporó una imagen representativa en la cabecera para ayudar a orientar al usuario, haciendo que este apartado, que podría no ser tan frecuentemente visitado, sea más accesible y visualmente claro.

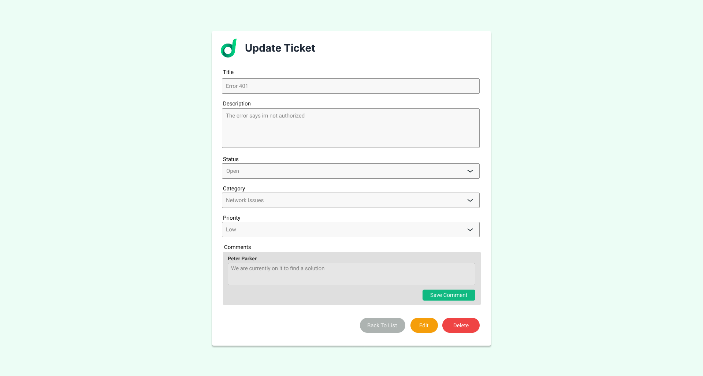


La vista de tickets para el usuario estará diseñada con una diferenciación de colores entre blanco y verde, donde el verde se utilizará específicamente para resaltar los comentarios realizados por los agentes. Esta distinción de colores ayudará a mejorar la claridad y facilitará la identificación rápida de la información proporcionada por los agentes.

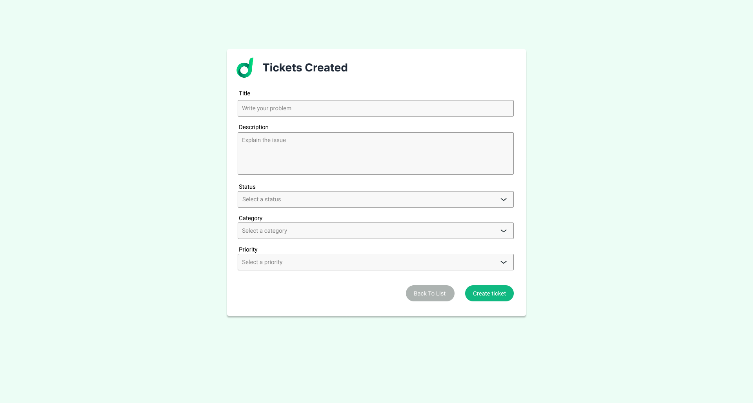
Por último, se presenta la vista de la incidencia desde el perfil del agente. Al igual que en la vista anterior, contará con la diferenciación de colores. Además, incluirá un menú lateral desde el cual se podrán modificar las características del ticket, como el estado, la prioridad, la categoría o el agente asignado, facilitando así una gestión más ágil y eficiente.

### Panel de administrador

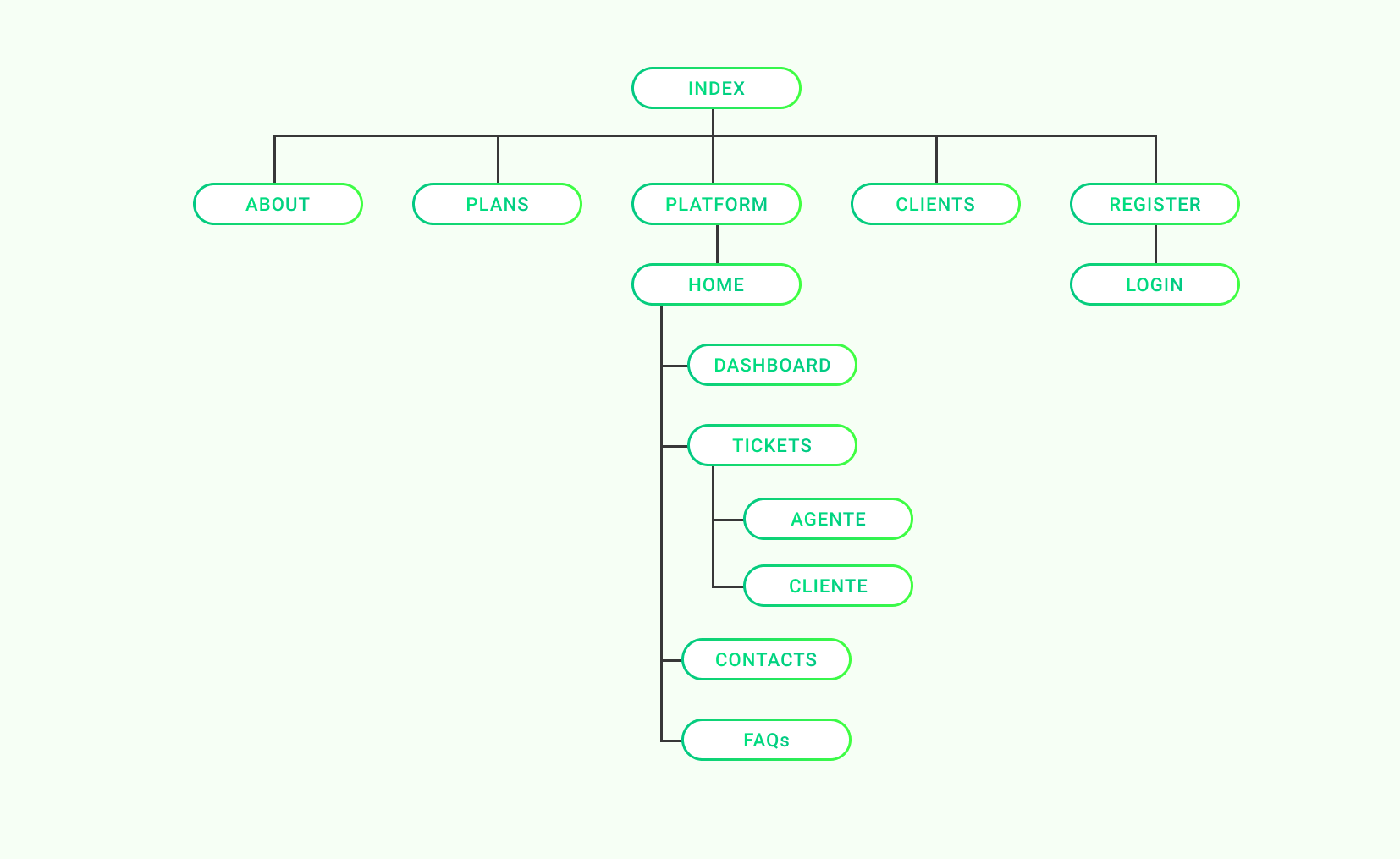
Como se mostró anteriormente, aquí se puede ver una versión final del panel de administración, la tabla de los tickets creados, que esta vez muestra un color corporativo con una paleta de colores ajustada a su funcionalidad para ayudar al usuario a identificarse mejor.



Cuando el administrador pulse el botón de editar, irá a la página individual de este ticket y podrá guiarse por la paleta de colores para realizar las acciones pertinentes. Podrá editarlo, borrarlo o volver al menú mediante los botones de control.

Por último, en la página de creación de incidencia, podrá crear un nuevo ticket, rellenando los campos necesarios, una vez completados se redirigirá automáticamente a la lista de tickets, para verificar la creación de la incidencia.

## Mapa de navegación

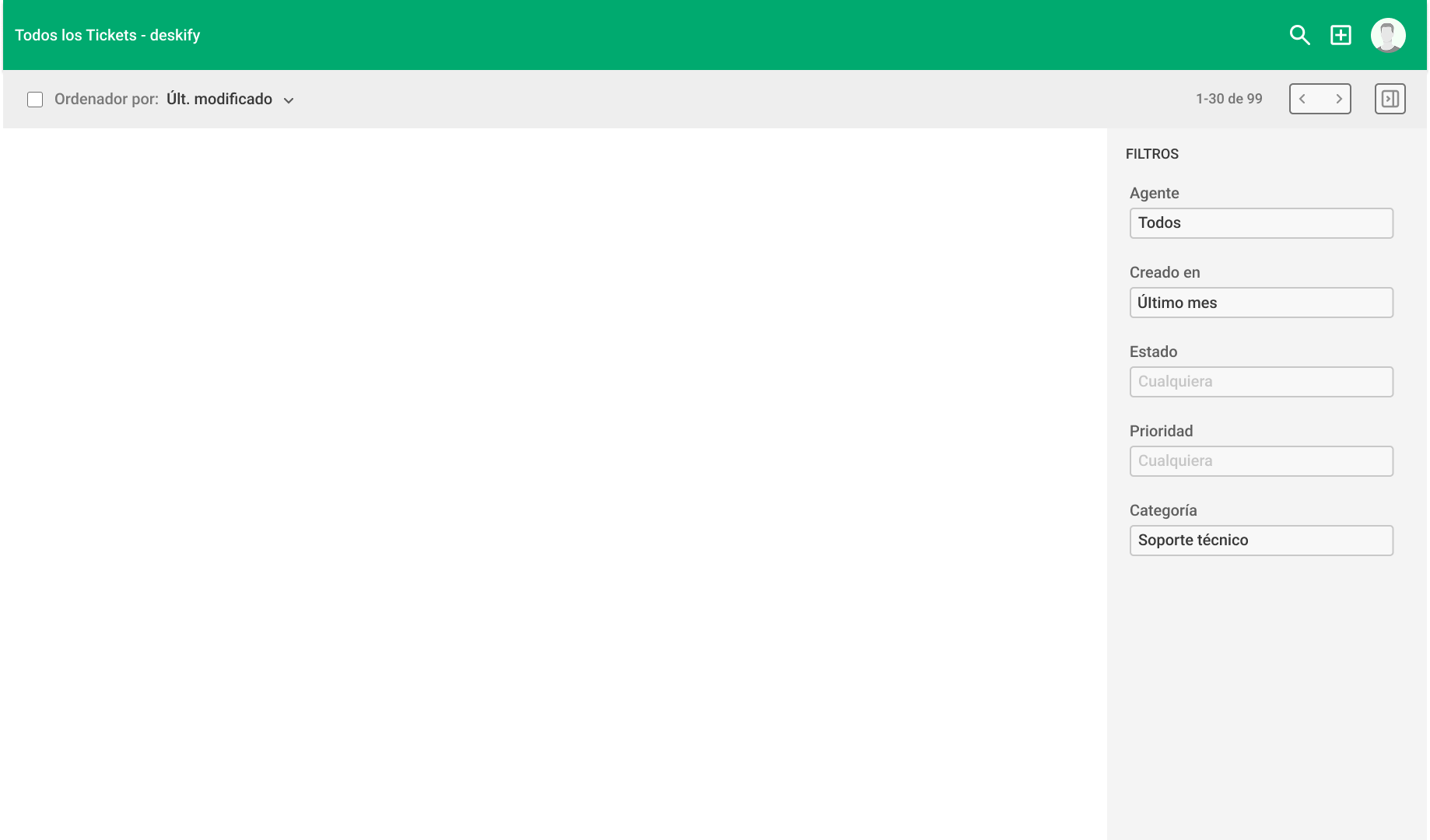
El mapa de navegación representa la estructura jerárquica de la plataforma, mostrando cómo se conectan entre sí las diferentes secciones y páginas. Este esquema facilita la comprensión del flujo de navegación y ayuda a visualizar la experiencia del usuario dentro de la aplicación. Gracias a este mapa, es posible identificar de forma clara las rutas principales, los accesos a funciones específicas según el rol del usuario y la organización lógica del contenido, lo cual resulta esencial para garantizar una navegación coherente, intuitiva y eficiente.

## Componentes UI

Los componentes UI (interfaz de usuario) son los elementos visuales reutilizables que conforman la estructura y funcionalidad de la plataforma, como botones, formularios, tarjetas, menús o modales. Cada uno de ellos ha sido diseñado siguiendo la guía de estilos y la paleta de colores definida previamente, asegurando coherencia visual y una experiencia de usuario fluida. Además, se han tenido en cuenta aspectos como la accesibilidad, la claridad y la usabilidad, buscando siempre que la interacción con la aplicación sea intuitiva, eficiente y agradable para todos los perfiles de usuario.

A continuación, se muestran los componentes usados en toda la web:





Faltan comprobaciones lighthouse, accesibilidad y usabilidad.

# Desarrollo del Proyecto

El desarrollo de Deskify ha estado marcado por decisiones técnicas cuidadosamente meditadas, siempre buscando el equilibrio entre el aprendizaje personal, la eficiencia del sistema y la escalabilidad futura del proyecto. A continuación, se describen las decisiones más relevantes tomadas a lo largo del proceso, así como la motivación detrás de cada una de ellas.

Desde el inicio del proyecto, uno de los pilares fundamentales fue la elección de tecnologías que no solo resolvieran las necesidades funcionales del sistema, sino que también me permitieran profundizar y aplicar de forma práctica los conocimientos adquiridos en el ciclo formativo. Por este motivo, se optó por una arquitectura dividida entre backend y frontend, empleando tecnologías modernas, bien documentadas y con una amplia comunidad.

## Backend - API con Spring

Para el backend de Deskify, se eligió Spring, el framework principal impartido en el ciclo formativo. Esta decisión fue clave para reforzar conceptos como la inyección de dependencias, la gestión de peticiones REST, la utilización de JPA/Hibernate para la persistencia de datos y el manejo de DTOs para la separación de lógica y datos.

Además de su potencia y madurez, Spring me permitió diseñar una arquitectura por capas clara y mantenible, con controladores, servicios y repositorios bien diferenciados, facilitando tanto las pruebas como futuras ampliaciones del proyecto. También se implementaron funcionalidades como:

* Autenticación y autorización mediante roles.
* Validación de formularios desde el backend.
* Mapeo con ModelMapper para mayor limpieza del código.

Gracias a esta elección, he podido experimentar con un entorno de desarrollo muy próximo al que se encontraría en una empresa real, preparándome mejor para mi futura etapa profesional.

## Frontend - Landing Page con Astro

Para la parte pública del proyecto, la landing page, se seleccionó Astro, una herramienta moderna para construir sitios estáticos optimizados para el rendimiento. Esta elección vino motivada por varias razones:

* La naturaleza estática de la página la hace ideal para Astro.
* Su compatibilidad con múltiples frameworks (React, Svelte, Vue) permite escalar en el futuro si se desea.
* Su enfoque en la entrega de contenido con la menor cantidad de JavaScript posible mejora la velocidad de carga, algo crucial en páginas de presentación.

La landing incluye información básica sobre Deskify, el objetivo del servicio y una llamada a la acción clara para registrarse. Se acompañó de un diseño minimalista y moderno siguiendo la paleta de colores previamente definida.

## Frontend - Aplicación con Svelte

En lo relativo a la aplicación web principal de Deskify, donde los usuarios gestionan sus tickets, se barajaron dos opciones tecnológicas: Angular y Svelte. Ambas se analizaron detenidamente en función de los siguientes criterios:

* Angular: Framework robusto y completo, ideal para aplicaciones empresariales. Su ecosistema incorpora muchas herramientas listas para usar, como validación de formularios, sistema de rutas, consumo de APIs o inyección de dependencias. Esto lo hace especialmente útil en un contexto como Deskify, donde se requiere una interfaz rica en interacciones y datos dinámicos.
* Svelte: Un framework emergente con un enfoque innovador. A diferencia de otros frameworks, Svelte compila a código JavaScript altamente optimizado, reduciendo así el tamaño del bundle y mejorando significativamente los tiempos de carga. Al no depender del virtual DOM, mejora el rendimiento, lo cual es especialmente útil en dispositivos de gama baja o conexiones lentas.

Tras valorar ambas opciones, se optó finalmente por Svelte, una tecnología moderna que destaca por su simplicidad, rendimiento y un enfoque distinto al de otros frameworks tradicionales. Esta decisión se tomó motivada por varios factores:

* Svelte representa una alternativa menos común que Angular, lo que me permite explorar una tecnología innovadora y ganar experiencia con herramientas emergentes del ecosistema frontend.
* Su curva de aprendizaje es más amigable, lo que facilita el desarrollo de interfaces limpias y mantenibles sin una sobrecarga de complejidad.
* Al compilar el código en tiempo de desarrollo y no depender de un virtual DOM, Svelte ofrece un excelente rendimiento, especialmente en términos de carga inicial y consumo de recursos.

Elegir Svelte me permite mantener un enfoque alineado con mis objetivos formativos, apostando por tecnologías innovadoras, al mismo tiempo que construyo una aplicación eficiente, moderna y fácil de mantener.

## Gestión de tickets y sistema de roles

Uno de los aspectos más importantes del desarrollo fue la gestión de tickets y el sistema de roles jerárquicos. Se implementó un control estricto de accesos para garantizar que cada usuario solo pudiera acceder a las funcionalidades que le correspondían. Esto implicó diseñar cuidadosamente:

* El modelo de datos.
* Las rutas protegidas por roles.
* La visualización condicional de la interfaz según el tipo de usuario.

El sistema es fácilmente ampliable para añadir más roles o permisos específicos en futuras versiones.

## Pruebas y validación

Durante el desarrollo se realizaron pruebas funcionales, pruebas con datos reales simulados, así como validaciones manuales del flujo de creación, visualización, filtrado y cierre de tickets. Se detectaron algunos puntos de mejora, como:

* Mejorar el feedback visual tras enviar un formulario.
* Revisar los casos en los que no se cargaban bien los datos desde la API tras operaciones encadenadas.

# Manual del Administrador

# Manual del Usuario

# Viabilidad Tecno-Económica

## Viabilidad Técnica

La viabilidad técnica del proyecto Deskify se basa en el uso de tecnologías modernas, eficientes y de código abierto, lo que permite su implementación sin necesidad de adquirir licencias costosas. El desarrollo se ha planteado de forma escalable, utilizando herramientas ampliamente adoptadas en el entorno profesional, lo que garantiza el mantenimiento y las futuras ampliaciones. A continuación, se detallan los aspectos técnicos clave que respaldan la posibilidad real de llevar a cabo el proyecto de forma funcional y sostenible.

* Lenguajes y frameworks usados:
  + FrontEnd (Svelte / Astro + Tailwind): Permiten crear una interfaz rápida, moderna, responsiva y fácil de mantener.
  + BackEnd (SpringBoot): Framework robusto, escalable y ampliamente usado en entornos empresariales.
  + Base de datos (MySQL): Sistema de gestión relacional gratuito y fiable.
* Servicios y herramientas:
  + Amazon S3: Para el almacenamiento de archivos estáticos, como imágenes de perfil o adjuntos en los tickets.
  + Git y GitHub: Control de versiones, colaboración y despliegue del proyecto.
* Diseño y prototipado:
  + Figma: Para wireframes, mockups, guía de estilos y moodboard.
  + StorySet, Freepik, iconify: Recursos gráficos usados en el diseño visual.
  + Adobe Firefly/ DeepL: Para generar imágenes y traducciones en fase de documentación.
* Infraestructura mínima para producción:

El desarrollo del proyecto ha estado a cargo de una única persona, que se ha encargado de todas las fases del proceso: desde la investigación inicial y el diseño de la interfaz, hasta la programación del frontend y del backend, incluyendo la gestión de la base de datos y el despliegue. Esta estructura unipersonal ha permitido tener una visión global del sistema y garantizar su coherencia. No obstante, durante el proceso se han recibido sugerencias y recomendaciones de profesores y compañeros que han contribuido a mejorar la calidad y funcionalidad del producto final.

## Viabilidad Económica

Para evaluar la viabilidad económica del proyecto, se han tenido en cuenta los costes necesarios para su implementación, tanto en cuanto a software como hardware, así como el posible retorno o utilidad que ofrecería la plataforma a largo plazo.

Los principales costes que se estiman a la hora de la creación del proyecto son los siguientes:

1. Costes de software: La mayoría de las tecnologías utilizadas para el desarrollo de la plataforma son de código abierto y gratuitas, lo que reduce considerablemente el coste total. Entre ellas destacan:
   * Svelte / Tailwind / Astro: Herramientas de desarrollo frontend totalmente gratuitas.
   * Vite.js: Plataforma de ejecución frontend gratuita.
   * Herramientas de diseño:
     + Figma: Gratuito en su versión básica.
     + Adobe Firefly: depende del plan, pero usado únicamente para generación de imágenes conceptuales.
     + Storyset / Freepik: se han usado recursos gratuitos bajo licencia de atribución.
     + Adobe Photohsop: usado para retoque de imágenes, pero tiene planes mensuales bastante asequibles por 12,07€/mes.
2. Costes de hardware/infraestructura:
   * Servidor VPS o hosting: Aproximadamente entre 5€ y 15€ al mes, dependiendo del proveedor.
   * Dominio web: Sobre 20€/año.
   * Certificado SSL: Gratuito con Let’s Encrypt.
   * Copias de seguridad/almacenamiento: Puede costar 0.05 por GB si se empieza a escalar el uso.
3. Tiempo de desarrollo: El tiempo que ha llevado desarrollar el proyecto, teniendo en cuenta que se ha realizado de forma unipersonal, ha supuesto a veces una gran carga de trabajo y quizá más duradera que si estuviera repartida entre un pequeño equipo. Partiendo de la base de que el proyecto ha supuesto unas 400 y 450 horas de trabajo durante 6 meses, incluyendo las fases de análisis, diseño, programación, pruebas y documentación. Si se estima un coste hipotético de 17 €/hora, el coste de desarrollo ascendería a unos 6800-7650 €, siendo este el mayor coste del proyecto.
4. Otros posibles costes:
   * Mantenimiento o soporte técnico: Si se delega a un tercero tendría un coste mensual, dependiendo del plan.
   * Diseño adicional o marketing: Variable según estrategia.

Estimación Total:

El coste inicial aproximado del proyecto es de 8.000€ - 9.000€, considerando tanto las inversiones en infraestructura como en tiempo de desarrollo. A partir de ahí, los costes recurrentes (hosting, almacenamiento, mantenimiento) son relativamente bajos y escalables, lo que hace que el proyecto sea económicamente viable a medida que crece su base de usuarios.

En términos de rentabilidad, el proyecto tiene el potencial de generar ingresos si se monetiza a través de suscripciones o servicios premium, por lo que, aunque la inversión inicial sea significativa, el retorno a medio y largo plazo puede justificarla.

# Trabajo a futuro

# Conclusiones

# Biblioteca de Recursos

Freepik Company, F. (2024): *Freepik – Recursos gráficos gratuitos*, Freepik, disponible en:  
<https://www.freepik.com>

Storyset, S. (2023): *Storyset – Ilustraciones animadas y personalizables*, Freepik, disponible en:  
<https://storyset.com>

Iconify, I. (2021): *Iconify – Biblioteca de iconos open source*, Iconify, disponible en:  
<https://iconify.design>

Adobe, A. (2023): *Adobe Firefly – Generación de imágenes con IA*, Adobe, disponible en:  
<https://firefly.adobe.com>

DeepL GmbH, D. (2024): *DeepL – Traductor automático de alta calidad*, DeepL, disponible en:  
<https://www.deepl.com>