Documentación - SolidApp

1 Introducción

1.1 - Nombre del proyecto

SolidApp – Plataforma de donaciones con sistema de recompensas

1.2 - Descripción general

SolidApp es una aplicación web que conecta organizaciones solidarias con usuarios interesados en colaborar mediante donaciones, incorporando un sistema de puntos que pueden canjearse por beneficios ofrecidos por empresas colaboradoras.

1.3 - Problema que resuelve

- Falta de visibilidad de campañas solidarias.
- Baja motivación en usuarios para donar.
- Empresas que quieren colaborar pero no tienen un canal digital centralizado.

1.4 - Objetivos

Objetivo general

Desarrollar una plataforma web que centralice campañas solidarias y fomente la participación mediante un sistema de recompensas.

Objetivos específicos

- Permitir a organizaciones publicar campañas.
- Permitir a usuarios donar y acumular puntos.

- Implementar un sistema de canje de cupones.
- Permitir a las empresas poder cargar sus beneficios(cupones).

1.5 - Justificación del proyecto

Actualmente, muchas organizaciones solidarias enfrentan dificultades para difundir sus campañas y captar la participación de la comunidad, debido a la falta de plataformas centralizadas que faciliten la interacción entre organizaciones, usuarios y empresas.

SolidApp surge como una solución tecnológica que permite centralizar la gestión de campañas solidarias, incentivar la participación mediante un sistema de recompensas y facilitar la colaboración del sector empresarial.

La implementación de esta plataforma permite mejorar la visibilidad de las campañas, aumentar la participación de los usuarios y optimizar la gestión de beneficios, contribuyendo al fortalecimiento del ecosistema solidario mediante el uso de tecnologías digitales.

1.6 - Alcance del proyecto

Alcance del sistema

El sistema SolidApp incluye el desarrollo de una aplicación web que permite:

- Registro e inicio de sesión de usuarios, empresas y organizaciones.
- Creación y gestión de campañas solidarias por parte de organizaciones.
- Sistema de donaciones con asignación de puntos.
- Sistema de canje de puntos por cupones.
- Panel de administración para gestión global del sistema.
- Panel de organización para gestión de campañas y las donaciones
- Panel de empresa para gestión de beneficios (Crear cupones o editarlos)
- Visualización de campañas, ranking y tienda.

El sistema será accesible a través de navegadores web en dispositivos desktop y móviles.

1.7 - Exclusiones del Proyecto

Exclusiones del Proyecto

El sistema no incluye:

- Integración con pasarelas de pago reales.
- Aplicación móvil nativa.
- Integración con sistemas externos gubernamentales.
- Sistema de auditoría avanzada.
- Inteligencia artificial para recomendación de campañas.
- Infraestructura cloud productiva (solo entorno local)

1.8 - Supuestos

- Se asume que las organizaciones validan correctamente las donaciones.
- Se asume que las empresas cumplen con los beneficios publicados.
- Se asume que los usuarios utilizan información verídica.
- Se asume conectividad a internet estable.
- Se asume que los usuarios se encargan de la logística de las donaciones a las organizaciones

2 Análisis del sistema

2.1 Actores

- **Usuario:** usuario-usuario -> Único que puede donar y recibir puntos por la donacion realizada

- **Organización:** usuario-organización -> Único que puede crear campañas, editarlas y darlas de baja.

- **Empresa:** usuario-empresa -> Único que puede crear cupones y modificarlo

- **Administrador:** usuario-administrador -> Único que tiene control total de la aplicación. Puede dar de alta y baja tanto las campañas,cupones,usuarios/usuarios,usuarios/organizacion,usuarios/empresas.

2.2 Requerimientos funcionales

- El sistema debe permitir el **registro e inicio de sesión** de **usuarios**.
- El sistema debe permitir a organizaciones **crear campañas** de **donación**.
- El sistema debe permitir a **usuarios visualizar campañas activas**.
- El sistema debe asignar **puntos** automáticamente al confirmar una **donación**.
- El sistema debe permitir **canjear** puntos por **cupones** disponibles.
- El sistema debe permitir a **empresas cargar cupones**.
- El sistema debe permitir al **administrador** poder **modificar estados** (alta/baja) de las **campañas,cupones,usuarios,organizaciones y empresas.**
- El sistema debe permitir a la **organización** ver las **donaciones realizadas** por las **campañas** de esa organización y así, poder aceptar o rechazar esas donaciones, y ver las **campañas** para poder **editarlas**.
- El sistema debe permitir a las empresas visualizar sus **cupones** y poder editarlos.

2.3 Requerimientos no funcionales

🔐 1. Seguridad

- El sistema debe implementar autenticación basada en **JSON Web Tokens (JWT)**.
- Las **contraseñas** deben almacenarse utilizando algoritmos de hash seguros (ej: bcrypt).
- El sistema debe aplicar **control de acceso** basado en **roles (Usuario, Organización, Empresa, Administrador)**.
- La comunicación entre **cliente y servidor** debe realizarse mediante HTTPS.
- El sistema debe prevenir **accesos no autorizados** a recursos protegidos.

⚡ 2. Rendimiento

- El sistema debe responder a solicitudes en un tiempo menor a **2 segundos bajo condiciones normales**.
- La carga inicial del frontend debe estar optimizada mediante **renderizado eficiente**.
- Las consultas a **base de datos** deben estar **optimizadas** mediante relaciones correctamente definidas e índices cuando sea necesario.

🗄️ 3. Persistencia y Gestión de Datos

- El sistema debe utilizar una **base de datos relacional** para garantizar integridad referencial.
- Las relaciones entre entidades deben cumplir restricciones de clave foránea.
- La información crítica **(donaciones, puntos, canjes)** debe almacenarse de forma **persistente.**
- El sistema debe garantizar **consistencia** en la asignación de puntos tras una donación.

🧱 4. Arquitectura

- El sistema debe seguir una arquitectura **cliente-servidor**.
- La aplicación debe estar estructurada en capas (**Presentación, Lógica de Negocio, Persistencia**).
- El backend debe implementarse siguiendo principios de **modularidad**.
- La API debe seguir **estándares REST**.

📱 5. Usabilidad

- La interfaz debe ser **responsive** y adaptarse a dispositivos móviles, tablets y desktop.
- El sistema debe presentar una navegación **intuitiva**.
- Las acciones críticas deben mostrar **mensajes** de **confirmación**.
- El usuario debe poder visualizar su **historial de puntos** y **canjes** de manera clara.

🔄 6. Escalabilidad

- El sistema debe permitir agregar nuevas funcionalidades sin afectar la estructura existente.
- La arquitectura debe permitir el despliegue futuro en servicios cloud.
- El diseño modular debe facilitar la integración con **pasarelas de pago** en el futuro.

🧪 7. Mantenibilidad

- El código debe seguir una estructura **modular**.
- El sistema debe utilizar **DTOs** para **validación de datos**.
- Las funciones deben estar separadas por responsabilidad.
- La aplicación debe permitir futuras mejoras sin reestructuración completa.

3️⃣ Arquitectura del sistema

- **Frontend**: Next.js / Sweet Alert 2 / shadcn/ui
- **Backend**: NestJS / TypeORM / Platform Express / Winston / Multer / bcrypt / passport-jwt /
- **Base de datos**: (Microsoft Sql Management Studio).
- **Comunicación**: API REST

Modelo arquitectónico:

Arquitectura cliente-servidor de tres capas:

- **Presentación** (Frontend)
- **Lógica de negocio** (Backend)
- **Persistencia** (Base de datos)

4️⃣ Modelo de datos

🧑‍💻 1. Entidad: Usuarios (`dbo.usuarios`)

📌 ¿Qué representa?

Representa a las personas registradas en la plataforma que pueden donar a campañas y canjear beneficios.

🔹 Atributos principales

- `id` (PK)
- `nombre`
- `apellido`
- `documento`
- `correo`
- `clave`
- `rol`
- `dirección` (calle, número, departamento, ciudad, provincia, codigoPostal)
- `telefono`
- `imagen`
- `fecha_registro`
- `ultima_conexion`

- `deshabilitado`
- `puntos`

🔗 Relaciones

- 1:N con **donaciones**
- 1:N con **usuarios_beneficios**
- 1:1 con **donador**
- 1:1 con **ranking_donador**

🎯 2. Entidad: Donador (`dbo.donador`)

📌 ¿Qué representa?

Registra el acumulado de puntos obtenidos por un usuario a partir de sus donaciones.

🔹 Atributos principales

- `id`
- `id_usuario` (FK)
- `puntos`

🔗 Relaciones

- N:1 con **usuarios**

🏆 3. Entidad: Ranking Donador (`dbo.ranking_donador`)

📌 ¿Qué representa?

Almacena el ranking general de donadores según la cantidad total de puntos acumulados.

🔹 Atributos principales

- `id_usuario` (PK/FK)
- `puntos`

🔗 Relaciones

- 1:1 con **usuarios**

🏢 4. Entidad: Empresas (`dbo.empresas`)

📌 ¿Qué representa?

Contiene la información de las empresas que colaboran con la plataforma ofreciendo beneficios o cupones.

🔹 Atributos principales

- `id` (PK)
- `nroDocumento`

- `razon_social`
- `nombre_fantasia`
- `descripcion`
- `rubro`
- `telefono`
- `direccion`
- `web`
- `correo`
- `clave`
- `verificada`
- `deshabilitado`
- `fecha_registro`

🔗 Relaciones

- 1:N con **beneficios (cupones)**
- 1:N con **imagenes_empresa**

🎁 5. Entidad: Beneficios (`dbo.beneficios`)

📌 ¿Qué representa?

Registra los cupones o beneficios ofrecidos por las empresas que pueden ser canjeados por puntos.

🔹 Atributos principales

- `id` (PK)
- `titulo`
- `detalle`
- `tipo`
- `cantidad`
- `valor`
- `estado`
- `id_empresa` (FK)
- `fecha_registro`

🔗 Relaciones

- N:1 con **empresas**
- 1:N con **usuarios_beneficios**
- 1:N con **imagenes_beneficio**

🛍️ 6. Entidad: Usuarios_Beneficios (`dbo.usuarios_beneficios`)

📌 ¿Qué representa?

Registra el canje de beneficios realizado por los usuarios.

🔹 Atributos principales

- `id`
- `id_usuario` (FK)
- `id_beneficio` (FK)
- `cantidad`
- `usados`
- `estado`
- `fecha_reclamo`
- `fecha_uso`

🔗 Relaciones

- N:1 con **usuarios**
- N:1 con **beneficios**

🏥 7. Entidad: Organizaciones (`dbo.organizaciones`)

📌 ¿Qué representa?

Contiene la información de las organizaciones solidarias que crean campañas.

🔹 Atributos principales

- `id`
- `nroDocumento`
- `razon_social`
- `nombre_fantasia`
- `descripcion`
- `telefono`
- `direccion`
- `web`
- `correo`
- `clave`
- `verificada`
- `deshabilitado`

🔗 Relaciones

- 1:N con **campañas**
- 1:N con **imagenes_organizacion**

📢 8. Entidad: Campañas (`dbo.campañas`)

📌 ¿Qué representa?

Representa las campañas de donación creadas por las organizaciones.

🔹 Atributos principales

- `id`
- `titulo`
- `descripcion`
- `objetivo`
- `fecha_inicio`
- `fecha_fin`
- `estado`
- `puntos`
- `id_organizacion` (FK)

🔗 Relaciones

- N:1 con **organizaciones**
- 1:N con **donaciones**
- 1:N con **imagenes_campaña**

🤝 9. Entidad: Donaciones (`dbo.donaciones`)

📌 ¿Qué representa?

Registra cada donación realizada por un usuario a una campaña, quedando sujeta a aprobación o rechazo por parte de la organización.

🔹 Atributos principales

- `id`
- `titulo`
- `detalle`
- `tipo`
- `cantidad`
- `puntos`
- `estado`
- `motivo_rechazo`

- `fecha_registro`
- `fecha_estado`
- `id_campaña` (FK)
- `usuarioId` (FK)

🔗 Relaciones

- N:1 con **usuarios**
- N:1 con **campañas**
- 1:N con **imagenes_donacion**

🖼️ 10. Entidades de Imágenes

Estas tablas almacenan recursos multimedia asociados a cada entidad principal.

📷 Imagenes_Beneficio

- `id`
- `imagen`
- `id_beneficio` (FK)

Relación: N:1 con beneficios

📷 Imagenes_Campaña

- `id`
- `imagen`
- `campañas_id` (FK)
- `esPortada`

Relación: N:1 con campañas

📷 Imagenes_Donacion

- `id`
- `imagen`
- `donaciones_id` (FK)

Relación: N:1 con donaciones

📷 Imagenes_Empresa

- `id`
- `logo`
- `banner`
- `empresas_id` (FK)

Relación: N:1 con empresas

📷 Imagenes_Organizacion

- `id`
- `logo`
- `banner`
- `organizaciones_id` (FK)

Relación: N:1 con organizaciones

5️⃣ Diseño del sistema

5.1 Diagrama de casos de uso

Ejemplo:

Usuario:

- Registrarse
- Iniciar sesión
- Ver campañas
- Donar
- Canjear cupón

Organización:

- Crear campaña

- Editar campaña

Empresa:

- Subir cupón

Administrador:

- Bloquear usuarios
- Moderar contenido

---

6 Implementación

El backend del sistema fue desarrollado utilizando **NestJS**, siguiendo una arquitectura modular basada en el patrón controlador-servicio, separando responsabilidades y manteniendo una estructura escalable y mantenible.

La aplicación se organiza en módulos independientes, cada uno responsable de una funcionalidad específica del sistema (usuarios, campañas, donaciones, beneficios, empresas, organizaciones, autenticación, etc.).

Backend (NestJS)

🔐 6.1.1 Módulo de Autenticación (`auth`)

El módulo `auth` gestiona:

- Registro de usuarios
- Validación de credenciales
- Generación de tokens JWT
- Protección de rutas

Se implementa autenticación basada en JWT mediante:

- `AuthService`
- `AuthController`
- `JwtStrategy`
- Guards personalizados

El token se genera tras validar correo y contraseña, y se utiliza para proteger rutas mediante decoradores y guards.

Esto permite control de acceso basado en roles:

- Usuario
- Organización
- Empresa
- Administrador

👤 6.1.2 Módulo de Usuarios

Gestiona:

- Creación de usuarios
- Actualización de datos
- Gestión de estado (habilitado/deshabilitado)
- Consulta de puntos

Incluye:

- `usuarios.controller.ts`
- `usuarios.service.ts`
- `usuarios.entity.ts`
- DTOs correspondientes

Se aplican validaciones antes de persistir datos en la base.

🏥 6.1.3 Módulo de Organizaciones

Permite:

- Registro de organizaciones
- Creación de campañas
- Gestión de donaciones recibidas
- Verificación y deshabilitación

Mantiene relación 1:N con campañas.

🏢 6.1.4 Módulo de Empresas

Gestiona:

- Registro de empresas
- Verificación
- Carga de beneficios

- Gestión de imágenes asociadas

Cada empresa puede publicar múltiples beneficios.

📢 6.1.5 Módulo de Campañas

Responsable de:

- Crear campañas
- Editar campañas
- Listar campañas activas
- Relacionar campañas con organizaciones

Incluye manejo de fechas (inicio y fin) y estado de la campaña.

🤝 6.1.6 Módulo de Donaciones

Este módulo es clave en el flujo del sistema.

Gestiona:

- Registro de donación
- Cambio de estado (pendiente, aprobada, rechazada)
- Asignación de puntos al usuario
- Registro de motivo de rechazo

Cuando una donación es aprobada:

1. Se actualiza el estado.

2. Se asignan puntos al usuario.
3. Se actualiza el ranking.

Esta lógica reside en el Service, no en el Controller.

🎁 6.1.7 Módulo de Beneficios

Permite:

- Crear beneficios asociados a empresas
- Controlar cantidad disponible
- Gestionar estado
- Asociar imágenes

Se relaciona con la tabla intermedia `usuarios_beneficios` para registrar canjes.

🏆 6.1.8 Módulo Ranking Donador

Calcula y almacena:

- Usuarios con mayor cantidad de puntos
- Posicionamiento en ranking general

Este módulo mejora la motivación dentro de la plataforma.

🖼️ 6.1.9 Módulos de Imágenes

Se implementaron módulos específicos para:

- Imágenes de campañas
- Imágenes de empresas
- Imágenes de organizaciones
- Imágenes de beneficios
- Imágenes de donaciones

Esto permite desacoplar recursos multimedia de la lógica principal.

🗄️ 6.1.10 Persistencia de Datos

Se utilizó TypefORM como ORM para mapear entidades a tablas en SQL Server.

Cada entidad define:

- **Claves primarias**
- **Columnas**
- **Relaciones** (OneToMany, ManyToOne)

El uso del ORM permite:

- **Integridad referencial**
- **Manejo** sencillo de **relaciones**
- Reducción de **consultas manuales**

🧱 6.1.11 Principios Aplicados

Durante el desarrollo del backend se aplicaron:

- Arquitectura **modular**
- Separación de **responsabilidades**
- Inyección de **dependencias**
- Validación mediante **DTOs**
- Control de acceso por **roles**
- Centralización de **lógica de negocio** en Services

Frontend - NextJs

El frontend del sistema SolidApp fue desarrollado utilizando **Next.js**, empleando TypeScript y una arquitectura basada en componentes reutilizables y separación por dominios funcionales.

La aplicación implementa el enfoque del **App Router**, permitiendo una organización estructurada de las vistas y una navegación eficiente entre módulos del sistema.

📁 6.2.1 Estructura General del Proyecto

El proyecto se organiza dentro de la carpeta `src/`, separando claramente:

- `app/` → Páginas del sistema
- `components/` → Componentes reutilizables
- `API/` → Capa de acceso al backend
- `context/` → Manejo de estado global
- `types/` e `interfaces/` → Tipado fuerte
- `validations/` → Validación de formulários
- `styles/` → Estilos mediante CSS Modules

Esta estructura permite escalabilidad y mantenimiento sencillo.

🧭 6.2.2 Organización por Módulos (Carpeta app)

La carpeta `app/` contiene las vistas principales del sistema:

- `adminPanel/`
- `campaign-catalogo/`
- `empresaPanel/`
- `organizacionPanel/`
- `ranking/`
- `tienda/`
- `userPanel/`
- `login/`
- `inicio/`
- `como-participar/`
- `Context`
- `quienes-somos`

Cada sección representa una funcionalidad específica del sistema, separando claramente los distintos roles:

- Usuario
- Organización
- Empresa
- Administrador

Esto refleja una segmentación por dominio funcional.

🧩 6.2.3 Componentización

La carpeta `components/` agrupa componentes reutilizables organizados por responsabilidad:

- Pages:
    - **Donaciones** - **donarModal**: tiene como función abrir un form que permite escribir los datos necesarios para que la persona pueda donar y seleccionar los datos a enviar.
    - **Inicio - Ranking**: Tiene el boton dentro de inicio que contiene la ruta de /ranking
    - **Login**: Formulario para ingresar al sitio web.
    - **Perfil**: El componente perfil contiene datos de, **Historial de cupones** del usuario (cantidad,usos), tiene **enviosInfo** (componente que permite ver datos de envío de las donaciones), **myAccount** (contiene los datos de la cuenta del usuario), **user&Pass** (contiene el formulario para poder cambiar la contraseña del usuario)
    - **Tienda**: el componente presenta: **Beneficios** (muestra los cupones de forma general o por empresa dentro de la tienda), **CanjeModa**l (componente que permite agregar cantidad de cupones que quiere el usuario canjear, suma la cantidad y se ve reflejado con botones de canjear y hace toda la lógica necesaria para que se envíen los datos para, restar los cupones de la empresa y que se le habiliten al usuario), **Empresa** (carga de forma general, todas las empresas que actúan en el sistema y deja clickearlas para poder ver por empresa los cupones que están habilitados), **Tienda** (Módulo que representa la tienda que contiene los cupones mezclados por empresas).

- **Footer y Navbar:**
- **Navbar:**Representa de forma global lo que sería el menú de navegación, conteniendo inicio,donar,tienda,quienes-somos,ranking, como participar, puntos del usuario, y el perfil de usuario.
- **Footer:** Contiene los enlaces del navbar, también contiene una frase que representa la aplicación.

🌐 6.2.4 Capa de Comunicación con Backend

Se implementó una capa de abstracción en la carpeta `API/`, la cual centraliza todas las llamadas HTTP hacia el backend desarrollado en NestJS.

Incluye:

- `baseApi.ts`
- `service.ts`
- Clases específicas por entidad:
    - **Campañas**
    - **Donaciones**
    - **Beneficios**
    - **Empresas**
    - **Organizaciones**
    - **Ranking**
    - **UsuarioBeneficios**

Esta capa permite:

- **Centralizar** la configuración de **fetch**
- Manejar **tokens** de **autenticación**
- Tipear correctamente las **respuestas**
- Evitar **repetición** de código

Esto demuestra la separación entre lógica de presentación y lógica de comunicación.

🔐 6.2.5 Manejo de Autenticación y Estado Global

Se implementó un sistema de autenticación que incluye:

- **Context API** (`UserContext`)
- Almacenamiento de **token JWT**
- Control de **rol** de **usuario**
- **Renderizado** condicional según permisos

El estado global permite:

- **Mantener** sesión activa
- **Identificar** tipo de usuario
- **Restringir** vistas según rol

Esto garantiza coherencia en la experiencia de usuario.

📦 6.2.6 Tipado Fuerte con TypeScript

Se definieron interfaces y tipos para todas las entidades del sistema:

- **Usuario**
- **Campaña**
- **Donación**
- **Beneficio**
- **Empresa**
- **Organización**
- **Ranking**

El uso de tipado fuerte permite:

- **Reducción** de errores en tiempo de desarrollo
- Mejor **mantenibilidad**
- Mayor **claridad** en la estructura de datos

📝 6.2.7 Validaciones

La carpeta `validation` contiene esquemas del registro para validar formularios antes de enviar datos al backend.

Esto permite:

- **Prevenir** envío de datos inválidos
- **Mejorar** experiencia de usuario
- **Reducir** carga innecesaria en el servidor

🎨 6.2.8 Estilos

Se utilizaron CSS Modules organizados por sección funcional.

Ventajas:

- **Encapsulamiento** de estilos
- **Evita** conflictos globales
- **Facilita** mantenimiento
- **Permite** diseño responsive

La interfaz fue diseñada para adaptarse a:

- **Dispositivos móviles**
- **Tablets**
- **Escritorio**

También se utilizó **Shadcn/ui** para realizar algunos de los **formularios, botones, inputs, calendarios**.

Sweet Alert: Se utilizó para utilizar las alertas y así dar más información al usuario al realizar acciones dentro del sistemas.

🧱 6.2.9 Principios Aplicados

Durante el desarrollo del frontend se aplicaron:

- **Arquitectura modular**
- **Separación de responsabilidades**
- **Componentización**
- **Reutilización**
- **Tipado fuerte**
- **Centralización de llamadas a API**
- **Manejo de estado global controlado**
- **Renderizado condicional por roles**

🎓 Conclusión Técnica del Frontend

La implementación del frontend se diseñó priorizando:

- **Escalabilidad**
- **Mantenibilidad**
- **Organización** por dominios
- **Experiencia de usuario intuitiva**
- **Integración** segura con el backend
- **Responsive**

La arquitectura modular permite incorporar nuevas funcionalidades sin afectar la estructura existente.

7 Flujo completo del sistema

Modo usuario:

1. Registrarse utilizando el register usando un mail y los datos de cuenta para poder registrarse.
2. Una vez creada la cuenta, ingresa mediante el login.
3. Una vez que el usuario loguea, tiene la opción de navegar por el navbar, pero en caso que quiera donar ingresa a Donar
4. El usuario ingresa a campaigns-catálogo, el cual puede mirar todas las opciones de donación.
5. Elige una y clickea en ver más información.
6. Aparecerá la campaña con su información (el objetivo y la empresa que lo requiere)
7. El usuario clickea en Donar ahora
8. Aparecerá una ventana la cual pide un detalle de la donación para poder escribir y explicar la donación que se va a realizar, y también la cantidad que el usuario va a donar(ejemplo 5 sillas).
9. Una vez el usuario envía la petición de donación, dentro del perfil en la sección de historial de donaciones, se encontrará la información para enviar el producto, con la dirección de envío a la organización.
10. Una vez que la organización da como confirmada la llegada de la donación, el sistema libera los puntos al usuario para poder utilizarse dentro de la aplicación.
11. Al usuario al liberarse los puntos puede ingresar a tienda, reclamar el cupón y este le restara los puntos dependiendo de cuantos reclamó; y estos aparecerán en el perfil en la sección de cupones, donde encontrará la información del cupón en si.

Mogo organizacion:

1. Registrarse en modo organización utilizando los datos de la organización.
2. Una vez creada la cuenta, ingresas mediante el login de la organización.
3. Una vez logueado, ingresas al inicio donde puedes ingresar al panel de organización que se encuentra en la esquina superior a la derecha de la página al costado del nombre del usuario.
4. Una vez logueado podrás ver que dentro del panel se encuentra las campañas que posee la organización, y las donaciones que recibe la organización mediante sus campañas.

5. La organización puede mediante las acciones de (x y tilde), deshabilitar las donaciones con la “x” , y habilitar con el tilde. Esto permite a la organización dar el ok de que la donación llegó tal cual se definió en el detalle enviado por el usuario.
6. También en el panel de campañas la organización puede editar la campaña en caso que unos de los datos al crear la campaña sean erróneos.

Modo empresa:

1. Registrarse de modo empresa, utilizando el registro de modo de empresa.
2. Loguearse una vez creada la cuenta.
3. En el inicio la empresa tiene la capacidad de acceder al panel de empresa en la esquina superior a la derecha, al costado del usuario-empresa.
4. Una vez ingresado al panel de empresa al clickear en el botón, se podrá observar que dentro del panel, la empresa podrá observar todos sus cupones creados dentro de la aplicación y también podrá crearlos mediante el botón de crear coupon y completar los datos del cupón mediante el form.
5. También la empresa tiene la opción de modificar los cupones existentes ya que se puede confundir en la carga de datos y quiere modificarlo desde el panel, lo puede hacer.

8 Riesgos del Proyecto

- Baja participación de usuarios.
- Falta de empresas colaboradoras.
- Errores en validación de donaciones.
- Vulnerabilidades de seguridad.
- Escalabilidad ante alto volumen de usuarios.
- Dependencia de validación manual por parte de organizaciones.
- Falta de participación de empresas

9️⃣ Limitaciones Técnicas

- No se implementó sistema de notificaciones en tiempo real.
- No se implementó sistema de cacheo.
- No se implementaron tests automatizados.
- No se implementó métricas para demostrar cuantas donaciones se realizaron a la donación, cuantos cupones se reclamaron a la empresa, cuántas personas visitaron la página web.

🔟 Plan de Evolución

Evolución Futura

- Integración con pasarelas de pago.
- Implementación de aplicación móvil.
- Sistema de notificaciones push.
- Panel de estadísticas avanzadas.
- Gamificación más avanzada.
- Integración con APIs externas.

1️⃣1️⃣Conclusión

- El Sistema Web de Gestión de Campañas, Beneficios y Participación,representa una solución tecnológica orientada a fortalecer la vinculación entre el sector solidario, las

organizaciones y el sector empresarial, mediante una plataforma centralizada que facilita la difusión, gestión y participación en campañas con impacto social.

- Desde una perspectiva social, el sistema contribuye a mejorar la visibilidad de iniciativas solidarias y promueve la participación activa de los usuarios, incentivando el compromiso mediante mecanismos como rankings, beneficios y recompensas. Esto permite generar una comunidad más comprometida, transparente y participativa.
- Asimismo, la plataforma permite una integración eficiente entre organizaciones y empresas, brindando herramientas para la creación, administración y control de campañas, beneficios y recompensas. Esta integración facilita la colaboración interinstitucional, optimiza la gestión de recursos y mejora la comunicación entre los distintos actores involucrados.
- En términos técnicos, el sistema fue desarrollado utilizando una arquitectura moderna basada en tecnologías web escalables, lo que permite su crecimiento futuro sin afectar su rendimiento ni su estabilidad. La estructura modular implementada facilita la incorporación de nuevas funcionalidades, como la expansión de la tienda, la integración de nuevos tipos de campañas o la incorporación de nuevos roles de usuario.
- Como línea de evolución futura, el sistema presenta un alto potencial de integración con pasarelas de pago digitales, lo que permitiría habilitar donaciones, compras dentro de la tienda y transacciones seguras directamente desde la plataforma. Esto ampliará significativamente las capacidades del sistema y su alcance operativo.
- En conclusión, el sistema desarrollado no solo cumple con los objetivos funcionales propuestos, sino que también establece una base sólida, escalable y sostenible para el crecimiento futuro, contribuyendo tanto al fortalecimiento del ecosistema solidario como a la modernización de su gestión mediante el uso de tecnologías digitales.