PROYECTO DE FINAL

ASIGNATURA: BASES DE DATOS 2

Alumnos:

Santiago de Olivera

Juan Pérez

Walter Taño

Resumen / Abstract:

El proyecto consiste en la creación de una aplicación que permita a los miembros de una comunidad ofrecer y recibir ayuda para diferentes tipos de necesidades. En ella, los usuarios podrán cargar las necesidades que tengan y las habilidades que puedan ofrecer. De esta manera, se podrá crear una red de apoyo mutuo, en la cual las personas puedan ayudarse entre sí, y así poder mejorar la calidad de vida de la comunidad por medio de la solidaridad de las personas.

Durante el transcurso del proyecto, se identificaron los requisitos funcionales presentes en la solicitud de la letra, se creó un modelo de entidad-relación para representar a las entidades del mundo real que participan del software y se desarrolló una aplicación capaz de cumplir con los requisitos identificados. A su vez, se encontraron diferentes soluciones posibles al problema planteado, las cuales fueron evaluadas por su complejidad y el valor aportado al usuario para elegir la mejor.

Este esfuerzo culminó en una herramienta que permite a sus usuarios identificarse, cargar un conjunto de habilidades y ofrecer su ayuda a otros usuarios. Al mismo tiempo, las personas registradas podrán crear una necesidad para un problema en específico y evaluar las postulaciones de ayuda de usuarios solidarios. Una vez establecida una conexión consensuada, se presentará la información de contacto de ambos usuarios para coordinar la actividad.

Contenido

Lanzamiento del proyecto:

El problema nos lo fue entregado en mano por el profesor de la materia, compuesto por 3 hojas donde figuraban los requerimientos del proyecto, y la letra del proyecto.

En resumen, la letra del proyecto nos pide que creemos un sistema de apoyo mutuo, en el cual las personas puedan ofrecer sus habilidades, y al mismo tiempo puedan buscar a alguien que pueda ayudarlos en algo que necesiten.

El sistema debe permitir que las personas puedan registrarse, y así poder ofrecer sus habilidades, y al mismo tiempo poder buscar a alguien que pueda ayudarlos en algo que necesiten. Un dato no menor es que las personas deben poder validar su identidad, para que no se creen usuarios falsos, y así poder tener una base de datos confiable, y que las personas puedan confiar en el sistema.

Con esta idea en mente, nosotros comenzamos a pensar en cómo podíamos llevar a cabo este proyecto, y así poder cumplir con todos los requerimientos que nos fueron entregados.

Dentro de las primeras cosas que nos planteamos tenemos:

El alcance del proyecto, es decir, que cosas vamos a poder hacer, y que cosas no vamos a poder hacer.

El tiempo que tenemos para realizar el proyecto, y así poder planificar el proyecto, y así poder cumplir con los plazos.

Los objetivos que queremos lograr, y así poder cumplir con los requerimientos del proyecto.

El propósito del proyecto, al cual nosotros le dimos un enfoque social, ya que el proyecto no persigue un fin económico, sino que un fin social, en el cual las personas puedan ayudarse entre sí, y así poder mejorar la calidad de vida de la comunidad.

Las restricciones que se presentan como, por ejemplo, el lenguaje de programación que vamos a utilizar, el motor de base de datos que vamos a utilizar, etc.

Los aspectos más relevantes y asunciones que vamos a tomar como, por ejemplo, que las personas van a ser honestas al momento de registrarse, y que van a ser honestas al momento de ofrecer sus habilidades, y al momento de buscar a alguien que pueda ayudarlos en algo que necesiten.

Algo no menor, la estimación de costos en tiempo de trabajo y recursos utilizados para el desarrollo del proyecto.

Y por último, la planificación del proyecto, en la cual vamos a definir las tareas que vamos a realizar, y los tiempos que vamos a utilizar para realizar cada tarea, y así poder cumplir con los plazos.

Alcance del proyecto

Cuando nos referimos a alcance del proyecto estamos hablando de que cosas vamos a poder hacer, y que cosas no vamos a poder hacer. En un inicio nos propusimos poder solventar los pedidos de la letra del proyecto, pero a medida que fuimos avanzando fuimos de a poco pensando en posibles mejoras al estilo GoldPlating que podríamos implementar, y así poder mejorar el sistema, y así poder cumplir con los requerimientos del proyecto, y al mismo tiempo poder cumplir con los objetivos que nos propusimos.

Tiempo

El tiempo que tenemos para realizar el proyecto es de 3 meses, y así poder planificar el proyecto, y así poder cumplir con los plazos. En un inicio nos propusimos realizar el proyecto en 2 meses, pero a medida que fuimos avanzando nos dimos cuenta de que no íbamos a poder cumplir con los plazos, y por eso decidimos extender el tiempo de desarrollo del proyecto a 3 meses. Uno de los motivos fundamentales para extender el tiempo de desarrollo del proyecto es que nosotros no teníamos experiencia en el desarrollo de aplicaciones web, y por eso nos costó mucho más de lo que pensamos en un inicio. Asimismo, el conocimiento de base de datos que es el centro del proyecto, no lo teníamos, porque a medida que fuimos avanzando en el semestre fuimos adquiriendo nuevas capacidades que nos ayudaron a poder desarrollar el proyecto.

Objetivos

Los objetivos que nos propusimos al enfrentar el problema del proyecto fueron los siguientes:

- Crear un sistema de apoyo mutuo, en el cual las personas puedan ofrecer sus habilidades, y al mismo tiempo puedan buscar a alguien que pueda ayudarlos en algo que necesiten.
- Crear un sistema que permita que las personas puedan registrarse y autenticarse, validando su identidad con un medio de confianza, para garantizar la seguridad de los usuarios del sistema y la calidad de los datos.
- Crear una base de datos extensible, que provea una infraestructura de datos y organización de estos para poder cumplir los requisitos planteados y adaptarse a requisitos nuevos.
- Crear una API adaptable para la fácil extensión e integración del sistema.
- Crear una aplicación web accesible, responsiva y extensible.

- Tener un producto final que cumpla con los requerimientos del proyecto, y que cumpla con los objetivos que nos propusimos. Que sea un producto que pueda ser utilizado por la comunidad, y que pueda ser mejorado en el futuro.

Propósito

El propósito fundamental del proyecto es poder ser capaces de desarrollar un sistema sobre un asunto tomado de la vida real que nos permita poner en práctica los conocimientos adquiridos durante el semestre, demostrar la capacidad de trabajo en equipos ante proyectos de similar índole.

Restricciones

En cuanto a las restricciones que tuvimos que afrontar a la hora de encarar el proyecto, fueron de diversa índole, pero las más importantes fueron las siguientes:

- El lenguaje de programación que vamos a utilizar, el cual fue Angular y NodeJS.
- El motor de base de datos que vamos a utilizar, el cual fue PostgreSQL.
- El tiempo que tenemos para realizar el proyecto, el cual fue de 3 meses.
- El conocimiento que teníamos sobre el desarrollo de aplicaciones web, el cual era muy poco.
- El conocimiento que teníamos sobre el desarrollo de base de datos, el cual era muy poco.

Aspectos relevantes y asunciones

El desarrollo de un proyecto como este que es entregado con una serie de indicaciones y sin un cliente que nos guíe en el proceso, nos obliga a tomar una serie de decisiones que pueden afectar el resultado final del proyecto. Es por esto por lo que es importante tener en cuenta los aspectos más relevantes y las asunciones que tomamos para poder desarrollar el proyecto.

Dentro de los aspectos más relevantes que tuvimos que afrontar fue la capacidad de generar un sistema confiable y seguro, que permita a los usuarios poder confiar en el sistema, y así poder utilizarlo. Otro aspecto importante fue la capacidad de generar un sistema que sea escalable, y que pueda ser mejorado en el futuro. Y por último, la capacidad de generar un sistema que sea fácil de utilizar, y que sea intuitivo, para que los usuarios puedan utilizarlo sin problemas.

Dentro de las asunciones que tomamos para poder desarrollar el proyecto, fue que las personas van a ser honestas al momento de registrarse, y que van a ser honestas al momento de ofrecer sus habilidades, y al momento de buscar a alguien que pueda ayudarlos en algo que necesiten.

Estimación de costos

La estimación de costos en tiempo de trabajo y recursos utilizados para el desarrollo del proyecto, inicialmente nos dividimos varios de los puntos que incluía la letra.
Esto lo hicimos por separado y después lo integramos al conjunto, para poder tener una estimación más acertada, que nos permitiera abordar todos los puntos de la letra.
Este tipo de tareas las realizamos en forma asincrónica, ya que cada uno de nosotros tiene sus tiempos y sus responsabilidades, y por eso no pudimos realizarlas en forma sincrónica. A pesar de esto el trabajo salió bien, y pudimos cumplir con los plazos que nos propusimos, el poder utilizar las redes sociales como medio de comunicación nos ayudó mucho, ya que nos permitió poder comunicarnos en cualquier momento, y así poder coordinar las tareas que teníamos que realizar.
La división de tareas nos permitió ahorrar tiempo y poder dilucidar los puntos más relevantes en cuestión de apenas semanas, con esta forma de trabajo pudimos tener los requerimientos en tiempo y forma, de modo que después solo bastaba el tener esperar por la operación con la base de datos, que fue lo que más tiempo nos llevó juntamente con el desarrollo del front-end.

Planificación del proyecto

La planificación del proyecto tal como ha sido descrita en los puntos anteriores nos facilitó enormemente la parte de desarrollo y sobre todo poder tener una idea clara de lo que queríamos hacer, y como lo queríamos hacer.

Aspectos relevantes del problema y del sistema

A la hora del planteo de la solución del problema, lo primero que vimos fue el cómo identificar a los posibles actores del sistema, para eso los dividimos en 2 clases. Por un lado, tenemos a los colaboradores que son aquellas personas que ofrecen sus habilidades, y por otro lado tenemos a los necesitados que son todas aquellas personas que marcaron una necesidad en el sistema.
En términos generales identificamos que el sistema debe hacer operaciones CRUD (Create, Read, Update, Delete) sobre las necesidades y las habilidades, y que debe permitir a los usuarios poder registrarse, y así poder ofrecer sus habilidades, y al mismo tiempo poder buscar a alguien que pueda ayudarlos en algo que necesiten.
El sistema debe permitir a dichos usuarios poder cargar cada una de sus necesidades, y al mismo tiempo poder cargar cada una de sus habilidades, y así poder generar una red de apoyo mutuo, de manera sencilla.

El sistema cuenta con una lista de habilidades, que es precargada por un administrador, y que los usuarios pueden elegir al momento de cargar sus habilidades, esto nos permite tener un control sobre las habilidades que se cargan en el sistema, y así evitar que se carguen habilidades que no sean reales, que no sean de utilidad, o que sean ofensivas.

El registro en el sistema es necesario para operar en el mismo, ya que de otra manera no se les permite a las personas poder cargar sus necesidades, ni tampoco se les permite poder cargar sus habilidades, ni tampoco se les permite poder buscar a alguien que pueda ayudarlos en algo que necesiten. Con este tipo de medidas nos aseguramos de darle cierta seguridad al sistema, y así evitar que se creen usuarios falsos, promoviendo una colaboración genuina entre los usuarios. Los pasos para el registro serán explicados en otro apartado de este documento.

Las necesidades de los usuarios serán relacionadas con una habilidad, y así poder generar una red de apoyo mutuo, de manera sencilla, dicho de otro modo, todas aquellas necesidades tendrán asociadas o podrán ser satisfecha por alguna de las habilidades existentes en el sistema. Los pasos para la carga de necesidades serán explicados en otro apartado de este documento.

Las personas con necesidades podrán buscar a alguien que pueda ayudarlos en algo que necesiten, o bien una persona que pueda ayudar a otra con su habilidad se postulará para ayudar a esa persona. Esta postulación es necesaria para poder validar que la persona que se postula para ayudar, realmente puede ayudar a la persona que necesita ayuda. Los pasos para la postulación serán explicados en otro apartado de este documento.

Comentando sobre el sistema el mismo permite filtrar a los usuarios según sus habilidades o necesidades según sea el caso, y así poder encontrar a alguien que pueda ayudarlos en algo que necesiten, o bien encontrar a alguien que pueda ayudar a otra persona con su habilidad. Se cuenta también con un mapa que es una representación tomada de Google Maps en la cual se reflejan los usuarios del sistema.

Al momento de implementar una notificación a los usuarios sobre otras personas que cumplan con los requisitos de poder actuar en mutuo apoyo con ellos, se decidió que la misma se realizara por medio de un correo electrónico, ya que es un medio de comunicación que todos los usuarios tienen, y que todos los usuarios utilizan. Asimismo, el sistema cuenta con información como el teléfono de la otra persona con la que estamos relacionándonos, y así poder comunicarnos con ella, y así poder coordinar la ayuda que necesitamos, o la ayuda que podemos brindar.

Las necesidades podrán ser almacenadas en la base de datos de manera tal que se pueda acceder a ellas en cualquier momento, y así poder buscar a alguien que pueda ayudarlos en algo que necesiten. Dentro de las asunciones que tomamos, consideramos que las personas no van a tener una necesidad de por vida, por lo tanto, se decidió que las necesidades tengan una fecha de vencimiento, y así poder eliminarlas de la base de datos, y así poder mantener la base de

datos lo más limpia posible, y así poder mejorar el rendimiento del sistema. Para esta tarea de eliminar datos obsoletos de la base de datos se identificaron 2 posibles formas, la primera y la que debería ser más utilizada, es que el usuario que cargo la necesidad, la elimine cuando ya no la necesite, y la segunda es que el sistema elimine las necesidades que ya no son necesarias, y que ya no son de utilidad. También existe los casos donde las necesidades ya han quedado resueltas y por tal motivo ya no es necesario tenerla en la lista de necesidades activas, y por tal motivo se decidió que el sistema elimine las necesidades que ya han quedado resueltas, y que ya no son de utilidad.

En referencia a las necesidades y la capacidad de iteración con otras personas, se decidió que las necesidades puedan ser modificadas, y así poder mejorar la descripción de la misma, y así poder mejorar la comunicación con las personas que puedan ayudarlos en algo que necesiten. También se decidió que las interrelaciones existentes entre necesitado y colaborador quedaran registradas en el sistema, como forma de llevar registros de las personas que se ayudan entre sí, y así poder tener un control sobre las personas que se ayudan entre sí, y así poder mejorar la seguridad del sistema.

Las personas que colaboran con otras tendrán la posibilidad de indicarlo para esto el sistema envía una notificación al usuario que cargo la necesidad, esta solicitud de permiso para ayudar quedará registrada en el sistema y tendrá una fecha de creación, con esto nos aseguramos de que todas aquellas postulaciones que pasen un tiempo determinado serán eliminadas.

Discusiones y valoración de alternativas

Consideramos varias alternativas durante el proceso de desarrollo, pero las más relevantes fueron las siguientes:

- Valoración de los usuarios colaboradores: Evaluamos la posibilidad de incluir un sistema de valoración para las personas que colaboran entre sí. Esto nos permitiría tener un control y mejorar la seguridad del sistema. Sin embargo, tras un análisis exhaustivo, decidimos no implementar esta funcionalidad, ya que consideramos que no era necesaria y no aportaba valor adicional al sistema.
- Sistema de mensajería entre usuarios colaboradores: Consideramos la opción de incorporar un sistema de mensajería para facilitar la comunicación entre las personas que colaboran. Esta funcionalidad nos ayudaría a tener un mayor control sobre las interacciones y mejorar la seguridad del sistema. No obstante, después de evaluarlo detenidamente, decidimos no

incluir esta característica, ya que consideramos que no era imprescindible y no proporcionaba beneficios significativos al sistema.

- Inclusión de comentarios para usuarios colaboradores: Evaluamos la posibilidad de permitir comentarios entre las personas que colaboran como medida de control y seguridad en el sistema. Sin embargo, después de una cuidadosa consideración, decidimos no implementar esta funcionalidad, ya que consideramos que no era necesaria y no aportaba valor adicional al sistema.
- Inclusión de una lista predictiva: Consideramos la opción de desarrollar una lista predictiva que permitiera a los usuarios buscar a personas que pudieran ayudar en sus necesidades o encontrar a alguien con habilidades específicas para brindar asistencia. No obstante, estimamos que el costo asociado para implementar esta funcionalidad era muy alto y podría afectar el flujo normal de selección de personas. Por lo tanto, decidimos descartar esta opción.
- En cuanto al diseño de la interfaz gráfica, consideramos dos alternativas importantes:
    - Inclusión de un mapa interactivo: Evaluamos la idea de integrar un mapa que permitiera a los usuarios seleccionar ubicaciones de manera más sencilla, con el objetivo de mejorar la experiencia del usuario. Inicialmente, consideramos una entrada manual de datos como ciudad, departamento y dirección, pero finalmente optamos por la opción del mapa interactivo.
    - Presentación de resultados en forma de tarjetas: Consideramos mostrar los resultados de las búsquedas de personas con habilidades o necesidades en forma de tarjetas, en lugar de una simple lista. Esta decisión se tomó con el fin de mejorar la experiencia del usuario.
- En cuanto al diseño de la base de datos, llevamos a cabo varias evaluaciones y revisiones antes de llegar al modelo de diseño entidad-relación final, que se explicará más adelante.

Investigación de antecedentes

En la actualidad existen muchas aplicaciones que permiten a las personas poder ayudarse entre sí, y así poder generar una red de apoyo mutuo, de manera sencilla. En este apartado se describirán algunas de las aplicaciones que existen en la actualidad, y que tienen como objetivo poder ayudar a las personas a poder ayudarse entre sí, y así poder generar una red de apoyo mutuo, de manera sencilla.

TaskRabbit

Es una plataforma en línea que permite a las personas contratar a otros para realizar tareas y trabajos diversos. La plataforma se basa en un modelo de economía colaborativa, conectando a personas que necesitan ayuda con tareas específicas, llamadas "tareas", con proveedores de servicios locales dispuestos a realizar esas tareas a cambio de una compensación.

La idea principal detrás de TaskRabbit es proporcionar una solución conveniente para la contratación de servicios y la realización de tareas cotidianas. Los usuarios pueden publicar las tareas que necesitan realizar, como tareas del hogar, montaje de muebles, limpieza, mudanzas, reparaciones, entrega de productos, entre otros. Luego, los proveedores de servicios registrados en TaskRabbit pueden presentar ofertas para realizar esas tareas.

Los usuarios pueden examinar las ofertas recibidas, revisar el perfil y las calificaciones de los proveedores de servicios, y seleccionar al candidato adecuado para realizar la tarea. Una vez que se completa la tarea, el usuario puede calificar y revisar al proveedor de servicios, lo que ayuda a mantener la confiabilidad y la calidad en la plataforma.

TaskRabbit opera en múltiples ciudades y áreas metropolitanas, y está disponible en varios países. La plataforma facilita la contratación y el pago de servicios, con TaskRabbit actuando como intermediario para asegurar que tanto los usuarios como los proveedores de servicios estén protegidos durante el proceso.

Nextdoor

Es una red social en línea diseñada específicamente para conectar a personas que viven en la misma área geográfica, como vecindarios y comunidades locales. A diferencia de otras redes sociales más amplias, Nextdoor se enfoca en promover la interacción y colaboración entre vecinos y en fomentar la construcción de comunidades locales.

Nextdoor permite a los usuarios crear perfiles y unirse a grupos específicos de su vecindario o área cercana. Dentro de la plataforma, los usuarios pueden compartir información, realizar publicaciones sobre eventos locales, noticias relevantes, recomendaciones de servicios locales, vender artículos usados, buscar recomendaciones o consejos, e incluso reportar situaciones de seguridad o emergencias en la comunidad.

La idea principal de Nextdoor es facilitar la comunicación y colaboración entre vecinos, promoviendo una mayor interacción cara a cara en la vida real y fortaleciendo los lazos comunitarios. La plataforma se ha utilizado para organizar actividades sociales, eventos comunitarios, grupos de voluntariado y otras iniciativas que involucran a los residentes de un área específica.

Es importante destacar que Nextdoor está diseñado para ser una red social privada y segura. Los usuarios deben verificar su dirección residencial para unirse a la plataforma y solo pueden interactuar con personas que viven en su vecindario o áreas cercanas. Esto ayuda a fomentar un sentido de confianza y pertenencia en la comunidad.

Airtasker

Similar a TaskRabbit, Airtasker es una plataforma en línea que conecta a personas que necesitan realizar tareas específicas con aquellos que tienen las habilidades y el tiempo para completar esas tareas. Se basa en un modelo de economía colaborativa, donde los usuarios pueden publicar tareas que necesitan ser realizadas y los proveedores de servicios pueden ofrecer sus habilidades y presentar ofertas para completar esas tareas.

La plataforma abarca una amplia gama de categorías, que incluyen servicios del hogar (limpieza, reparaciones, jardinería), tareas informáticas y tecnológicas, servicios de entrega, cuidado de mascotas, trabajos de escritura y diseño, entre otros. Los usuarios pueden encontrar a alguien dispuesto a realizar la tarea que necesitan y negociar los detalles, como el precio y el tiempo de entrega.

La idea detrás de Airtasker es permitir que las personas obtengan ayuda para realizar tareas específicas de manera conveniente y eficiente. Los usuarios pueden leer las reseñas y calificaciones de los proveedores de servicios para tomar decisiones informadas sobre a quién contratar. Una vez que se completa la tarea, el usuario puede calificar al proveedor de servicios y dejar comentarios sobre su experiencia.

Airtasker opera en varios países, aunque su disponibilidad puede variar según la ubicación. La plataforma cobra una tarifa de servicio a los proveedores de servicios por las tareas

completadas, y los pagos entre usuarios se manejan a través del sistema de pago seguro de Airtasker.

Meetup

Es una aplicación y plataforma en línea que facilita la creación y participación en grupos sociales basados en intereses comunes. La aplicación está diseñada para ayudar a las personas a conectarse con otras que comparten sus pasiones, hobbies, actividades o metas específicas en la vida.

La idea central de Meetup es permitir a los usuarios crear grupos locales o unirse a grupos ya existentes en su área geográfica. Estos grupos pueden estar centrados en una amplia variedad de temas, como deportes, música, arte, tecnología, lectura, bienestar, voluntariado, entre otros. Los miembros de un grupo pueden organizar y asistir a eventos y reuniones relacionados con su interés común.

La aplicación Meetup ofrece funciones como la búsqueda de grupos por tema o ubicación, la visualización de eventos próximos y la comunicación con otros miembros a través de mensajes y chats. Los usuarios también pueden recibir notificaciones sobre eventos y actualizaciones relevantes de los grupos a los que se han unido.

La principal ventaja de Meetup es que proporciona una plataforma para que las personas con intereses similares se reúnan en la vida real, lo que fomenta la interacción social y la creación de comunidades locales. Los eventos organizados en Meetup pueden variar desde pequeñas reuniones informales hasta grandes encuentros o conferencias, según el tamaño y la popularidad del grupo.

LinkedIn

Es una red social profesional en línea que se utiliza principalmente con fines de networking y desarrollo profesional. Es una plataforma diseñada para que los profesionales, empresas y empleadores se conecten, compartan información y establezcan relaciones laborales.

LinkedIn permite a los usuarios crear perfiles profesionales detallados, donde pueden incluir su experiencia laboral, habilidades, educación y logros. Los usuarios también pueden agregar una foto profesional y una descripción de sí mismos. Además, LinkedIn proporciona un espacio para que los usuarios publiquen contenido relacionado con su campo profesional, como artículos, actualizaciones y opiniones.

La red de contactos es un componente clave de LinkedIn. Los usuarios pueden conectar con colegas, compañeros de clase, empleadores, clientes y otras personas relevantes en su campo profesional. Estas conexiones pueden brindar oportunidades de networking, colaboración, mentoría y desarrollo de carrera.

LinkedIn también ofrece herramientas para buscar y postularse a empleos, así como para que las empresas publiquen ofertas de trabajo y busquen candidatos potenciales. Los usuarios pueden unirse a grupos temáticos, seguir empresas y recibir actualizaciones relevantes sobre el mundo laboral.

Además de las funciones básicas de la red social, LinkedIn ofrece servicios adicionales como cursos de aprendizaje en línea (LinkedIn Learning), herramientas de ventas y marketing (LinkedIn Sales Navigator) y soluciones de contratación (LinkedIn Recruiter).

Thumbtack

Es una plataforma en línea que conecta a personas que necesitan contratar servicios con proveedores de servicios locales. La plataforma abarca una amplia gama de categorías, como mejoras para el hogar, eventos y entretenimiento, bienestar y fitness, clases y tutorías, servicios profesionales, entre otros.

La idea principal de Thumbtack es brindar una solución conveniente para la contratación de servicios, al permitir a los usuarios publicar detalles sobre el trabajo o servicio que necesitan y recibir propuestas y cotizaciones de proveedores de servicios calificados en su área. Los usuarios pueden revisar las calificaciones, reseñas y perfiles de los proveedores para tomar decisiones informadas.

Una vez que se selecciona un proveedor de servicios, los usuarios pueden comunicarse con ellos a través de la plataforma para discutir los detalles, precios, plazos y cualquier otra información relevante para completar el trabajo. Thumbtack actúa como intermediario entre los clientes y los proveedores de servicios, proporcionando una plataforma segura para la comunicación y la contratación.

Thumbtack es utilizado por personas y empresas que buscan servicios profesionales en diversas áreas. Los proveedores de servicios pueden utilizar la plataforma para promocionar sus habilidades y servicios, llegar a un público más amplio y obtener clientes potenciales.

Formas de validación de identidad

Actualmente existen diversas formas de verificar la identidad de una persona, y así poder asegurarse de que la persona que se está registrando en el sistema es quien dice ser. En este apartado se describirán algunas de las formas de verificación de identidad que existen en la actualidad, y que son relevantes para la aplicación.

Verificación en Persona

El método más común y sencillo para verificar la identidad de una persona de forma presencial es solicitándole una tarjeta de identificación emitida por el gobierno que contenga una fotografía (como una licencia de conducir, tarjeta de identificación estatal o pasaporte). Antes que nada, es importante verificar que la fotografía coincida con la persona que se está presentando.

Algunos consejos para verificar la identidad de una persona mediante una foto son:

1. Observar detenidamente la identificación con foto y determinar qué tipo de identificación es. Sin embargo, debido a las leyes de privacidad y seguridad de la información, no se debe hacer una copia de la identificación ni anotar su número a menos que sea absolutamente necesario para el servicio solicitado.
2. Tener en cuenta que existen tarjetas de identificación con fotografía falsas. Es recomendable familiarizarse con el formato de las identificaciones gubernamentales utilizadas para la verificación y examinar cuidadosamente la identificación. Por ejemplo, muchas licencias de conducir estatales tienen microimpresiones, lo cual es difícil de falsificar. Si no está familiarizado con el formato del número de identificación (por ejemplo, el número de licencia de conducir de un país o estado), puede realizar una búsqueda en línea para verificarlo.
3. Si la fotografía en la identificación no coincide claramente con la persona presente o si se requiere mayor seguridad para confirmar su identidad, se puede solicitar una segunda identificación que puede o no incluir una foto. Algunos ejemplos de esto son la tarjeta de Seguro Social, tarjeta de crédito, factura de servicios públicos con el nombre y dirección correctos, tarjeta de identificación de la escuela o universidad, entre otros. Al igual que con la primera identificación, no se debe hacer una copia de estos documentos ni anotar los números de identificación a menos que sea necesario para el servicio solicitado, ya que los números de seguridad social y tarjetas de crédito están protegidos por las leyes de privacidad y seguridad de la información.

4. Si se requiere un nivel adicional de seguridad, se puede hacer una pregunta específica. Sin embargo, es importante asegurarse de que la pregunta y la respuesta se mantengan en privado y seguras, en caso de que haya personas cercanas que puedan escuchar la respuesta. Por ejemplo, si se necesita solicitar información específica, se puede pedir a la persona que la escriba en un papel para que pueda ser verificada con los registros correspondientes. Después, se debe asegurar de triturar el papel en presencia de la persona.

Al iniciar sección en una aplicación o sistema

En el inicio de sesión de una aplicación o sistema, además de utilizar una contraseña o frase de contraseña, es posible que se requiera un token adicional. La inteligencia artificial también se está utilizando para detectar rostros como método de acceso a aplicaciones o sistemas. Este enfoque proporciona una mayor seguridad y ayuda a prevenir estafas y el lavado de dinero de manera más confiable.

Al iniciar sección en una red social

Verificar la identidad de una persona a través de las redes sociales (Facebook, Google+, Twitter, LinkedIn, etc.) no se considera recomendable. Es relativamente sencillo crear cuentas y perfiles falsos en estas plataformas, y la información compartida en ellas está destinada a ser pública. Por lo tanto, no son entornos adecuados para compartir datos de verificación que estén protegidos por leyes de privacidad y seguridad de la información, como documentos o números de identificación.

Tarjetas y lectores biométricos

Las tarjetas de proximidad o deslizantes y los lectores biométricos, como los escáneres de huellas dactilares, de manos o de geometría de manos, son métodos mecánicos ampliamente utilizados para realizar verificaciones de identidad automatizadas en las personas. Estas tecnologías permiten una autenticación precisa y eficiente.

Identificación Digital

Verificar la identidad se ha convertido en un requisito fundamental para realizar trámites y asuntos relacionados con una persona. El Estado Uruguayo ha establecido diversas formas de verificación de identidad digital, que incluyen diferentes niveles de seguridad, ya que se reconoce que no todos los trámites requieren el mismo nivel de seguridad.

En el nivel básico, se realiza un registro completando un formulario en línea y se confirma a través de correo electrónico. El acceso se realiza con un nombre de usuario y contraseña, lo que permite iniciar y dar seguimiento a trámites y servicios tanto públicos como privados.

En el nivel intermedio, se verifica la seguridad de la Identidad Digital para confirmar que la persona es quien dice ser. Existen dos formas de elevar el nivel de seguridad de Básico (Autorregistrado) a Intermedio (Verificado): a través de firma digital o de manera presencial en puntos autorizados.

Por último, se encuentra un nivel que incluye la Cédula de Identidad con Chip, el ID Digital de Abitab y el TuID de Antel.

La Cédula de Identidad con Chip requiere tener la cédula de identidad con chip, un lector de cédula y conocer el PIN correspondiente.

El ID Digital de Abitab requiere tener una Identidad Digital de Abitab certificada de manera presencial en una sucursal de Abitab, además de descargar una aplicación en el teléfono celular para la identificación. Este método permite la firma digital.

El TuID de Antel requiere tener una Identidad Digital de Antel certificada de manera presencial en una sucursal de Antel, además de descargar una aplicación en el teléfono celular para la identificación. Este método también permite la firma digital.

Verificación de 2 pasos

La verificación en dos pasos es un método de autenticación que agrega una capa adicional de seguridad a las cuentas en línea. En lugar de depender únicamente de una contraseña, este proceso requiere dos elementos distintos para verificar la identidad del usuario.

El funcionamiento típico de la verificación en dos pasos es el siguiente:

Contraseña: El usuario ingresa su nombre de usuario y contraseña, como lo haría normalmente para acceder a su cuenta en línea.

Factor adicional: Una vez que la contraseña es verificada, se solicita al usuario un segundo factor de autenticación. Este factor adicional puede ser algo que el usuario posee, como un código de

verificación generado en una aplicación de autenticación o enviado a través de un mensaje de texto, o algo inherente al usuario, como su huella digital o reconocimiento facial.

Verificación exitosa: Si el usuario ingresa correctamente el segundo factor de autenticación, se le concede el acceso a su cuenta.

El objetivo de la verificación en dos pasos es dificultar el acceso no autorizado a una cuenta, incluso si alguien descubre o roba la contraseña. Incluso si un atacante obtiene la contraseña, aún necesitaría el segundo factor de autenticación para poder acceder a la cuenta. Esto agrega una capa adicional de seguridad y reduce el riesgo de que una cuenta sea comprometida.

Es importante destacar que los métodos exactos de verificación en dos pasos pueden variar según la plataforma o el servicio en línea utilizado. Algunas aplicaciones pueden utilizar códigos de verificación generados por aplicaciones de autenticación como Google Authenticator o enviar códigos a través de mensajes de texto. Otros métodos pueden incluir el uso de llaves de seguridad físicas o tecnologías biométricas, como huellas dactilares o reconocimiento facial.

Utilización de la API de Google Maps

La API de Google Maps es una interfaz de programación de aplicaciones (API, por sus siglas en inglés) proporcionada por Google que permite a los desarrolladores integrar y utilizar las funcionalidades de los mapas de Google en sus propias aplicaciones y sitios web. Esta API ofrece acceso a una amplia gama de servicios y datos relacionados con la ubicación geográfica, como la visualización de mapas interactivos, la búsqueda de lugares, la obtención de indicaciones de ruta, la geocodificación (asociar una dirección con coordenadas geográficas) y la inversión de geocodificación (obtener una dirección a partir de coordenadas geográficas).

Al utilizar la API de Google Maps, los desarrolladores pueden crear aplicaciones que aprovechan las características y capacidades de Google Maps para proporcionar a los usuarios una experiencia de mapas enriquecida y personalizada. Esto incluye la integración de mapas interactivos en aplicaciones móviles, sitios web, servicios de transporte, aplicaciones de viaje, aplicaciones de entrega y muchas otras.

La API de Google Maps ofrece una documentación detallada, ejemplos de código y diversas opciones de configuración para adaptarse a las necesidades y requisitos específicos de cada proyecto. Es ampliamente utilizada en la industria para crear aplicaciones basadas en la ubicación y proporcionar a los usuarios información geográfica precisa y actualizada.

PostgreSQL

Aunque la cátedra nos dio la libertad de elegir cualquier base de datos SQL para cargar desde una Máquina Virtual, decidimos utilizar PostgreSQL. Habíamos estado utilizando PostgreSQL en la materia de Base de Datos II y creímos que era la opción más adecuada para resolver el problema, ya que ya estábamos familiarizados con esta herramienta.
A continuación, se procederá a explicar algunos detalles prácticos de la Base de Datos PostgreSQL, que nos pareció oportuno añadir al documento.

¿Qué es PostgreSQL?

PostgreSQL es un sistema de bases de datos de código abierto reconocido por su flexibilidad y cumplimiento con el estándar SQL. En la actualidad, se considera uno de los motores de bases de datos más avanzados de la industria.
Este proyecto es gratuito y de código abierto, impulsado por la comunidad de desarrolladores PGDG. Ofrece una amplia gama de opciones avanzadas para administrar bases de datos relacionales en programación. Una característica destacada de PostgreSQL es su control de concurrencia multiversión (MVCC), que proporciona una vista del estado de la base de datos a cada transacción.
Las capacidades de PostgreSQL brindan notables ventajas en términos de rendimiento y manejo de datos. Al aprovechar el MVCC, este sistema permite realizar transacciones de manera eficiente, optimizando el rendimiento general del sistema y mejorando la experiencia del usuario.

¿Para qué sirve PostgreSQL?

PostgreSQL es una poderosa herramienta que se utiliza en diversos sectores, como el gobierno, la industria, la educación y el comercio, para gestionar grandes volúmenes de datos. A continuación, se describen algunas de sus funciones principales:

- **Almacenamiento eficiente de datos**: PostgreSQL organiza y almacena los datos de manera eficiente, garantizando su integridad y consistencia. Proporciona mecanismos robustos para asegurar la integridad de los datos y mantener la coherencia en todo el sistema.
- **Consultas avanzadas**: Este sistema de bases de datos permite realizar consultas complejas y realizar análisis de datos utilizando el lenguaje SQL. Ofrece un amplio conjunto de

funciones y operadores para manipular y extraer información valiosa de los datos almacenados.

- **Backend confiable para aplicaciones**: PostgreSQL es ampliamente utilizado como backend confiable en el desarrollo de aplicaciones web y móviles. Proporciona un almacenamiento seguro y escalable para los datos de la aplicación, lo que garantiza un rendimiento óptimo incluso en entornos con altas cargas de trabajo.
- **Soporte geoespacial**: PostgreSQL cuenta con funciones avanzadas para manejar datos geoespaciales, lo que lo convierte en una opción ideal para aplicaciones que requieren análisis y visualización de mapas. Puede almacenar y manipular datos relacionados con ubicaciones geográficas, como coordenadas, polígonos y rutas, y realizar consultas espaciales para obtener información geoespacial precisa.

¿Cuáles son las principales características de PostgreSQL?

PostgreSQL es un sistema de bases de datos con muchas características que lo hacen diferente de otros sistemas de gestión.

Muchos usuarios al mismo tiempo

Una característica especial de PostgreSQL es que puede manejar a muchas personas usando la base de datos al mismo tiempo. Gracias a su método de control de concurrencia multiversión (MVCC), puede asegurarse de que todas las personas puedan hacer transacciones al mismo tiempo sin problemas.

Diferentes tipos de datos

PostgreSQL ofrece muchos tipos de datos diferentes para almacenar información. No solo puede guardar números y palabras, sino también fechas, direcciones IP, datos geográficos y mucho más. Esto hace que sea más fácil diseñar una base de datos que se adapte a las necesidades de cada proyecto.

Consultas complicadas

Además, PostgreSQL permite hacer consultas complicadas usando su lenguaje SQL poderoso. Esto ayuda a analizar los datos y obtener información valiosa de ellos.

Replicación y disponibilidad

PostgreSQL también puede replicar datos y asegurarse de que siempre estén disponibles. Esto ayuda a crear sistemas fuertes y que puedan crecer si es necesario.

Seguridad avanzada

Por si fuera poco, PostgreSQL también se preocupa mucho por la seguridad. Tiene funciones avanzadas para asegurar los datos, como autenticación, encriptación y control de acceso a diferentes partes de la información.

Datos geográficos

PostgreSQL tiene herramientas especiales para trabajar con datos geográficos. Es una excelente opción para aplicaciones que necesitan hacer análisis y mostrar mapas.

Útil para aplicaciones

Finalmente, PostgreSQL es muy útil para construir aplicaciones web y móviles. Puede guardar los datos de manera segura y hacer que la aplicación funcione bien, incluso si hay muchos usuarios.

Plataforma de desarrollo Angular

Después de un análisis exhaustivo de las diferentes opciones disponibles, el grupo tomó la decisión de desarrollar el proyecto en Angular en lugar de Java. Esta elección se basó en consideraciones de practicidad y facilidad de uso.

Angular, siendo un framework de desarrollo web ampliamente utilizado y con una curva de aprendizaje accesible, nos brindó la confianza de poder desarrollar el proyecto de manera eficiente y efectiva. Su estructura modular y su enfoque en la creación de aplicaciones de una sola página fueron aspectos que nos resultaron muy beneficiosos para nuestro proyecto.

Además, optamos por utilizar la biblioteca de Angular Material para el diseño de la interfaz gráfica. Consideramos que Angular Material era la mejor opción debido a sus numerosos componentes de interfaz de usuario predefinidos, que cumplen con los principios de diseño de Material Design. Estos componentes nos permitieron desarrollar una interfaz atractiva y coherente de manera rápida y sencilla.

La combinación de Angular como framework de desarrollo y Angular Material para el diseño de la interfaz gráfica nos brindó una solución integral que nos permitió enfocarnos en la lógica del proyecto sin tener que preocuparnos demasiado por los aspectos técnicos y estilísticos. En general, consideramos que estas elecciones fueron acertadas y nos facilitaron el proceso de desarrollo del proyecto de manera significativa.

¿Qué es Angular?

Angular es una poderosa plataforma de desarrollo web que se destaca por su enfoque en la modularidad y su base en el lenguaje TypeScript. Su principal fortaleza radica en el sistema de componentes, que permite crear y reutilizar piezas de código independientes y autónomas.

Esta plataforma ha ganado una gran popularidad en la industria debido a su capacidad para abordar proyectos de gran envergadura y escalabilidad. Es especialmente adecuada para construir aplicaciones web complejas y de una sola página (SPA), donde se requiere un manejo eficiente de los componentes y un flujo de datos bidireccional.

Una de las ventajas de Angular es su extensa colección de bibliotecas integradas, conocida como Angular Material. Estas bibliotecas ofrecen una amplia gama de componentes y funciones predefinidos, lo que facilita el desarrollo rápido y consistente de interfaces de usuario atractivas y responsivas. Además, Angular Material sigue los principios de diseño Material Design de Google, lo que brinda una apariencia moderna y coherente a las aplicaciones.

Otra característica destacada de Angular es su enfoque en la productividad del desarrollador. Proporciona una amplia gama de herramientas, como Angular CLI (Command Line Interface),

que automatiza tareas comunes de desarrollo, como la generación de componentes, servicios y pruebas unitarias. Además, Angular cuenta con una sólida comunidad de desarrolladores que comparten recursos, tutoriales y soluciones a problemas comunes, lo que facilita el aprendizaje y la resolución de desafíos.

Las principales partes de Angular

La plataforma de desarrollo Angular, está compuesta por 3 ideas centrales que la hacen ser tan versátil. Entre ellas están los Componentes, las Plantillas y la Inyección de dependencias.

Componentes

Los componentes son los elementos fundamentales que conforman una aplicación. Cada componente está compuesto por una clase de TypeScript con un decorador, una plantilla HTML y estilos asociados. El decorador, identificado con @Component(), proporciona información específica de Angular.

- Dentro del decorador, se encuentra un selector de CSS que define cómo se utiliza el componente en una plantilla. Los elementos HTML de la plantilla que coinciden con este selector se convierten en instancias del componente.
- Además, el componente incluye una plantilla HTML que le indica a Angular cómo debe representarse visualmente.
- Opcionalmente, se pueden añadir estilos CSS que definen la apariencia de los elementos HTML dentro de la plantilla.

Esta característica de Angular, junto a su plantilla correspondiente, es esencial para la reutilización de lógica y diseños con los cuales mostrar la información de la aplicación y permitirle interactuar con la misma a los usuarios.

Plantillas

Cada componente en Angular cuenta con una plantilla HTML que define cómo se mostrará ese componente en la interfaz de usuario. Esta plantilla puede ser declarada directamente en el código del componente o en un archivo separado.

Angular introduce una serie de elementos de sintaxis adicionales al HTML estándar para permitir la inserción de valores dinámicos provenientes del componente. Estos valores pueden ser variables, expresiones o incluso directivas estructurales que controlan la lógica de visualización.

Una de las características clave de Angular es que se encarga de actualizar automáticamente el Document Object Model (DOM) cuando el estado del componente cambia. Esto significa que, si hay cambios en los datos del componente, la plantilla se actualizará automáticamente para reflejar dichos cambios en la interfaz de usuario, sin necesidad de manipular directamente el DOM.

Inyección de Dependencias

La inyección de dependencia es un concepto importante en Angular que le permite declarar las dependencias que necesita una clase de TypeScript sin preocuparse por crear las instancias de esas dependencias. En lugar de eso, es Angular quien se encarga de crear las instancias por los desarrolladores. Este enfoque se basa en un patrón de diseño que tiene ventajas significativas, como facilitar la prueba del código y brindar flexibilidad en su desarrollo.

Cuando se utiliza la inyección de dependencia en Angular, pueden declararse las dependencias de una clase mediante anotaciones. Angular se encargará de crear y proporcionar automáticamente esas dependencias cuando se instancie la clase en tiempo de ejecución. Esto simplifica el proceso de crear y mantener objetos y evita acoplamientos directos entre las clases. Muchos aspectos de Angular, como los servicios, los componentes y los proveedores, se benefician de la inyección de dependencia y la utilizan en cierta medida.

Al aprovechar la inyección de dependencias, podemos crear lógica altamente reutilizable, ubicada en servicios, para que el cliente de la aplicación pueda interactuar con el servidor y realizar operaciones en la base de datos.

La librería Angular Material

Angular Material es una biblioteca de componentes de interfaz de usuario (UI) diseñada para facilitar la creación de aplicaciones web atractivas y coherentes en Angular. Una de sus principales ventajas es que todos sus componentes siguen los principios de Material Design de Google, lo que garantiza una apariencia moderna y consistente en todas las aplicaciones web.

Angular Material provee una gran variedad de componentes listos para utilizar y con APIs claramente definidas. Estos incluyen botones, formularios, menús, barras de herramientas, tablas, entre otros. La versatilidad de estos componentes nos permite personalizarlos y utilizarlos extensamente, lo que facilita la creación de aplicaciones web que se adaptan sin problemas a cualquier dispositivo.

Además de los componentes, Angular Material proporciona servicios útiles para personalizar la apariencia de nuestra aplicación. Por ejemplo, el servicio de paleta de colores de Angular

Material nos permite definir una paleta personalizada para adaptar los colores de la aplicación para mantener cierta apariencia de marca deseada. Si bien en este proyecto estamos utilizando una de las paletas predefinidas de Angular, cambiarla por una personalizada sería una modificación simple de estilos y no requeriría modificaciones de código en ningún componente de la aplicación.

Back-end en Node.js con Express

Para implementar el back-end de nuestro proyecto, optamos por utilizar Node.js, un entorno de ejecución de JavaScript que permite ejecutar código JavaScript fuera del navegador. Node.js se destaca por su modelo de operaciones E/S sin bloqueo y orientado a eventos, lo cual lo hace eficiente y liviano. Además, cuenta con el ecosistema de paquetes más grande del mundo, npm, que ofrece una amplia gama de bibliotecas de código abierto.

Node.js goza de una gran popularidad y se utiliza en diversos tipos de aplicaciones web, desde aquellas que requieren tiempo real y manejan grandes volúmenes de usuarios, hasta servidores web capaces de manejar miles de solicitudes por minuto. Empresas de renombre como Netflix, LinkedIn, Uber, PayPal, NASA y Yahoo!, entre otras, confían en Node.js para sus aplicaciones.

En nuestro proyecto, también aprovechamos Express, un framework de Node.js que simplifica la creación de aplicaciones web y APIs. Express ofrece una variedad de características para el desarrollo de aplicaciones web y móviles. Su facilidad de aprendizaje y uso lo convierten en una opción destacada tanto para desarrolladores principiantes como experimentados.

Desarrollo en Node.js con Express

En el contexto de Node.js con Express, las dependencias como Controller, Router, Helper y Middlewares son conceptos clave que ayudan en el desarrollo de aplicaciones web. Aquí hay una breve explicación de cada uno:

- **Controller:** Un controlador es una parte de la arquitectura de una aplicación que se encarga de manejar las solicitudes y respuestas HTTP. En Express, un controlador es una función que define la lógica de negocio para una ruta específica. Ayuda a separar la lógica de manejo de solicitudes del enrutamiento y facilita la reutilización del código.
- **Router:** El enrutador (Router) en Express es un objeto que nos permite definir las rutas y las acciones que se deben realizar para cada ruta. Proporciona una forma de organizar y gestionar las diferentes rutas de una aplicación. Permite asociar controladores a rutas específicas y manejar las solicitudes y respuestas correspondientes.

- **Helper:** Un helper (ayudante) en Node.js con Express es una función o conjunto de funciones que se utilizan para realizar tareas comunes en el desarrollo de una aplicación. Estas funciones auxiliares pueden incluir operaciones repetitivas, validaciones, formateo de datos, manejo de errores, entre otras. Los helpers ayudan a mantener el código más limpio, modular y reutilizable.
- **Middlewares:** Los middlewares son funciones que se ejecutan entre la solicitud y la respuesta en Express. Actúan como interceptores, procesando la solicitud antes de que llegue al controlador y modificando la respuesta antes de que se envíe al cliente. Los middlewares se utilizan para realizar tareas como la autenticación, la validación de datos, el registro de solicitudes, el manejo de errores, entre otros. Permiten agregar funcionalidades adicionales a la aplicación sin necesidad de modificar directamente el controlador.

Eventos de negocio

Comencemos con definir que es un evento de negocio, un evento de negocio se refiere a un suceso o acción significativa relacionada con una organización o empresa que tiene implicaciones para el funcionamiento de sus sistemas y procesos informáticos.

Eventos de negocios identificados

A continuación, enumeramos una lista de eventos de negocios que nosotros entendimos como los principales para el desarrollo del sistema.

1. Una persona se registra en la app
2. Una persona se logue en la app
3. Una persona publica sus habilidades
4. Una persona publica sus necesidades
5. Una necesidad es satisfecha
6. Una necesidad expira
7. Una persona busca personas con ciertas habilidades
8. Una persona se ofrece a ayudar con una necesidad
9. Una persona aprueba una solicitud para ayudar con una necesidad
10. Una persona rechaza una solicitud para ayudar con una necesidad
11. Una persona activa el sistema de geolocalización
12. Una persona capaz de resolver una necesidad personal está en la cercanía (Completo)
13. Una persona edita una necesidad propia.
14. Una persona edita su perfil (datos personales, habilidades)

Casos de uso del producto

Los casos de uso de productos se refieren a la descripción detallada de las interacciones y escenarios específicos en los que los usuarios utilizan un producto o sistema. Estos casos de uso son fundamentales para comprender cómo los usuarios interactúan con el producto y determinar qué funcionalidades y características son necesarias para satisfacer sus necesidades. En el ámbito del desarrollo de software, los casos de uso de productos se emplean para capturar los requisitos funcionales del sistema y proporcionar una descripción minuciosa de la forma en que se espera que los usuarios interactúen con él. Estos casos de uso describen las acciones que los usuarios realizan, las respuestas que se espera del sistema y los resultados deseados.

Cada caso de uso incluye una descripción detallada de una interacción específica entre el usuario y el producto, detallando los pasos que el usuario sigue, las entradas que proporciona, las salidas que espera recibir, así como cualquier condición o restricción relevante.
Los casos de uso de productos son una herramienta efectiva para comunicar y documentar los requisitos del sistema de manera comprensible tanto para los desarrolladores como para los usuarios y otras partes interesadas. Contribuyen a establecer una base clara para el diseño, desarrollo y pruebas del producto, asegurando que se satisfagan las necesidades y expectativas de los usuarios.

Casos de uso de producto

- **Caso 1**:
  - BE: Registro de usuarios
  - PUC: Una persona se registra en la app
  - Precondiciones: La persona no debe haberse registrado anteriormente
  - Stakeholders interesados: Todos (?)
  - Stakeholders activos: El usuario
  - Pasos:
    - 1. El usuario entra a la aplicación de sign up
    - 2. El usuario llena la información solicitada:
      - Nombre de usuario
      - C.I.
      - Nombres
      - Apellidos
      - Correos electrónicos
      - Teléfonos
      - Contraseña
      - Repetir contraseña
      - Otros (?)
    - 3. El usuario envía el formulario
    - 4. Se verifican los datos
      - E4.1. Los datos son inválidos de algún modo:
        - El nombre de usuario ya está ocupado
        - El CI ingresado no es válido, o no existe

        - La contraseña no es lo bastante segura
        - Las contraseñas no coinciden
      - E4.1.1. Se notifica al usuario
    - 5. Se notifica que el proceso está hecho
      - A5.1. La notificación llega por correo electrónico.
      - A5.2. La notificación es a través de la aplicación.
  - Resultado: El usuario está registrado y puede acceder a las funcionalidades del servicio.

- **Caso 2**:
  - BE: Login de usuarios
  - PUC: Una persona se logue en la app
    - Precondiciones: La persona debe haberse registrado anteriormente, pero no está logueada
    - Stakeholders interesados: Todos (?)
    - Stakeholders activos: El usuario
    - Pasos:
      - 1. El usuario entra en la aplicación de log in
      - 2. El usuario llena la información solicitada:
        - Nombre de usuario
        - Contraseña
        - Otros (?)
      - 3. El usuario envía el formulario
      - 4. Se verifican los datos
        - E4.1. Los datos son inválidos de algún modo
        - E4.1.1. Se notifica al usuario
        - E4.2. La autenticación falla
        - E4.2.1. Se notifica al usuario
        - E4.3. La contraseña es incorrecta
        - E4.3.1. Se notifica al usuario
      - 5. Se notifica que el proceso está hecho
    - Resultado: El usuario está logueado

- **Caso 3**:

- BE: Cargar habilidades
- PUC: Una persona publica sus habilidades
- Precondiciones: El usuario debe estar logueado, la persona debe tener completados los datos básicos de información
- Stakeholders interesados: ...
- Stakeholders activos: El usuario
- Pasos:
    - 1. El usuario entra a la aplicación de ingresar habilidades
    - 2. El usuario llena el formulario:
        - Tipo
        - Descripción
        - Límite de tiempo
    - 3. El usuario envía el formulario
        - E3.1. Hay algún problema con los datos enviados
        - E3.1.1. Se notifica al usuario
    - 4. Se notifica que el proceso está hecho
- Resultado: El usuario tiene una nueva habilidad almacenada

- **Caso 4**:
    - BE: Cargar necesidades
    - PUC: Una persona publica sus necesidades
    - Precondiciones: El usuario debe estar logueado, la persona debe tener completados los datos básicos de información.
    - Stakeholders interesados: El usuario con necesidad y el ayudado.
    - Stakeholders activos: El usuario necesitado
    - Pasos:
        - 1. El usuario entra a la aplicación con Login
        - 2. El usuario ingresa la necesidad en el menú con las opciones preestablecidas.
        - 3. El usuario indica fecha, hora y lugar que necesita la ayuda.
            - A3.1. El sistema le sugiere la necesidad si el usuario tiene histórico de necesidades.
            - E3.1. El sistema valida que la fecha ingresada sea mayor al día actual.
            - E3.2. El sistema válido la ubicación ingresada.
        - 4. El usuario recibe validación de que su solicitud quedo ingresada.

- Resultado:
    - El usuario genero una solicitud de necesidad de habilidades.

- **Caso 5:**
    - BE: Una necesidad es satisfecha
    - PUC: El sistema debe guardar la cooperación de las personas (necesidad – ayuda)
    - Precondiciones: El usuario tiene que cargar su necesidad, otro usuario debe poder ayudarlo.
    - Stakeholders interesados: El usuario de la aplicación, el usuario que ayuda.
    - Stakeholders activos: El usuario final
    - Pasos:
        - 1. El sistema registra que existe una necesidad
        - 2. Los usuarios se contactan a través de una aplicación externa.
        - 3. Los usuarios registran la actividad.
            - A3.1. El usuario ayudado, indica que el usuario (ayudante) le apoyo a solucionar necesidad.
            - A3.2. El usuario ayudante, indica que ayudo a otro usuario.
        - 4. El sistema procesa que ambas partes acordaron que estaba realizada la actividad y que se cumplió lo esperado.
    - Resultado:
        - El sistema registra la nueva necesidad que fue satisfecha.

- **Caso 6 :**
    - BE: Una necesidad expira
    - PUC: Expiración de necesidades
    - Precondiciones: La necesidad debe tener un límite de tiempo, el cual haya acabado
    - Stakeholders interesados: ...
    - Stakeholders activos: El dueño de la necesidad
    - Pasos:
        - 1. Se marca a la necesidad como pausada
        - 2. Se notifica al dueño que la necesidad expiró
        - 3. Se pregunta si quiere actualizarlo
            - A3.1. El usuario quiere actualizar la fecha de expiración
            - A3.1.1. Se realiza el PUC entero de edición de necesidades

        - A3.2. El usuario quiere eliminar la necesidad
        - A3.2.1. Se elimina la necesidad
        - A3.3. El usuario quiere dejar la necesidad sin publicar o no actúa
        - A3.3.1. No se hace nada
    - Resultado: La necesidad está actualizada, eliminada o temporalmente almacenada en la base de datos sin acceso público

- **Caso 7:**
    - BE: Una persona busca personas con ciertas habilidades
    - PUC: Búsqueda de personas con habilidades.
    - Precondiciones:
        - La persona debe estar logueada.
        - Deben existir registros de personas con habilidades.
        - La ubicación de la persona debe ser visible en el mapa.
    - Stakeholders interesados:
        - Personas con necesidades
        - Personas con habilidades
    - Stakeholders activos:
        - Personas con necesidades
    - Pasos:
        - 1. La persona con necesidades abre el menú de búsqueda.
        - 2. La persona ingresa los datos de la habilidad que desea buscar.
        - 3. El sistema lista las personas que tienen la habilidad que se indicó.
            - A3.1. El sistema permite ver las personas con habilidades en el mapa, indicando la ubicación de estas.
            - A3.2. El sistema ranquea a las personas con habilidades según su calidad de colaboración.
        - 4. La persona solicita ayuda a la/las persona que más le conviene/n según su ubicación y necesidad.
    - Resultado:
        - El usuario recibe una lista con personas que pueden satisfacer su necesidad.
        - El/los usuarios con habilidades seleccionados reciben una notificación para ayudar con una necesidad.

- **Caso 8:**
  - BE: Persona ofrece ayuda
  - PUC: Una persona se ofrece a ayudar con una necesidad.
  - Precondiciones:
    - La persona debe estar logueada.
    - Deben existir registros de personas con necesidades.
    - La ubicación de la persona debe ser visible en el mapa.
  - Stakeholders interesados:
    - Personas con necesidades
    - Personas con habilidades
  - Stakeholders activos:
    - Personas con habilidades
  - Pasos:
    - 1. La persona con habilidades abre el menú de búsqueda.
    - 2. La persona ingresa los datos de la habilidad posee para buscar.
    - 3. El sistema lista las personas que tienen la necesidad que se indicó.
      - El sistema permite ver las personas con necesidades en el mapa, indicando la ubicación de las mismas.
      - El sistema ranquea a las personas con habilidades según su calidad de colaboración.
    - 4. El sistema permite que la persona con habilidades pueda contactarse con la persona con necesidad.
  - Resultado:
    - El usuario recibe una lista con personas que puede ayudar.
    - El usuario con habilidad se contacta con usuarios con necesidades.

- **Caso 9:**
  - BE: Una persona aprueba una solicitud para ayudar con una necesidad.
  - PUC: Aprobación de apoyo de una necesidad.
  - Precondiciones:
    - La persona debe estar logueada.
    - La ubicación de la persona debe ser visible en el mapa.
    - El usuario debe tener al menos una necesidad
    - Debe existir al menos una postulación de ayuda para la necesidad.

- Stakeholders interesados:
  - Personas con necesidades
  - Personas con habilidades
- Stakeholders activos:
  - Personas con necesidades
- Pasos:
  - 1. Persona accede al detalle de su necesidad.
  - 2. Producto muestra postulantes.
  - 3. Persona verifica información de postulante(s).
  - 4. Persona aprueba postulante.
  - 5. Producto actualiza el estado de la postulación a “Aprobada” en el sistema.
  - 6. Producto envía una notificación de aprobación con información de la necesidad al postulante.
- ~~Pasos: (evidencia anterior)~~
  - ~~1. La persona con habilidades recibe el pedido de ayuda.~~
  - ~~2. La persona con habilidades valida que pueda cumplir con las necesidades que tiene la otra persona.~~
  - ~~3. El sistema muestra la información de la persona con necesidades.~~
  - ~~4. La persona con habilidades aprueba la petición de ayuda.~~
- Resultado:
  - La postulación de ayuda es aprobada.
  - El usuario solidario recibe una notificación.

- **Caso 10:**
  - BE: Rechazo de apoyo de una necesidad.
  - PUC: Una persona rechaza una solicitud para ayudar con una.
  - Precondiciones:
  - La persona debe estar logueada.
  - Deben existir registros de personas con necesidades.
  - La ubicación de la persona debe ser visible en el mapa.
  - Debe existir una petición de ayuda con una habilidad.
  - Stakeholders interesados:
  - Personas con necesidades
  - Personas con habilidades

- Stakeholders activos:
- Personas con habilidades
- Pasos:
- 1.La persona con habilidades recibe el pedido de ayuda.
- 2. La persona con habilidades valida que pueda cumplir con las necesidades que tiene la otra persona.
- 3. El sistema muestra la información de la persona con necesidades.
- 4. La persona con habilidades rechaza la petición de ayuda.
- Resultado:
  - La persona con habilidad rechaza el pedido de ayuda, y la petición se marca como no resuelta.
  - El usuario con necesidades recibe la notificación de que no lo pueden ayudar de apoyo.

- **Caso 11:**
  - BE: Activar geolocalización.
  - PUC: Una persona activa el sistema de geolocalización.
  - Precondiciones:
    - La persona debe estar logueada.
    - Se debe disponer de la geolocalización.
  - Stakeholders interesados:
    - Personas con necesidades
    - Personas con habilidades
  - Stakeholders activos:
    - Personas con habilidades
    - Personas con necesidades
  - Pasos:
    - 1. La persona indica el lugar donde se encuentra.
    - 2. La persona indica el rango de distancia en donde puede actuar.
    - 3. La persona confirma la ubicación ingresada.
  - Resultado:
    - El sistema proporciona la información de la ubicación de las personas según la indicación.

- **Caso 12:**
  - BE: Una persona capaz de resolver una necesidad personal está en la cercanía.
  - PUC: Notificación de cercanía.
  - Precondiciones:
    - Ambas personas deben haberse registrado previamente.
    - Ambas personas deben haber habilitado la geolocalización e ingresado una dirección.
    - Una de las dos personas tiene una necesidad activa.
    - La otra tiene una habilidad capaz de satisfacer dicha necesidad
    - Ambas personas deben estar en el rango de geolocalización de la otra
  - Stakeholders interesados:
    - Personas con necesidades
    - Personas con habilidades
  - Stakeholders activos:
    - Personas con necesidades
  - Pasos:
    - 1. Se genera una notificación informándole al usuario (necesitado) que una persona (colaborador) capaz de resolver una de sus necesidades se encuentra cerca.
    - 2A1. El necesitado pide ayuda al colaborador
      - 2A1.1 El colaborador recibe una notificación de la petición de ayuda.
      - 2A1.2 El colaborador puede navegar hacia la necesidad a través de la notificación.
    - 2A2. El usuario descarta la notificación.
  - Resultado:
    - El usuario es notificado de la proximidad de un posible colaborador.
    - El colaborador es notificado de su capacidad de contribuir a resolver una necesidad.

- **Caso 13:**
  - BE: Modificar necesidad.
  - PUC: Una persona edita una necesidad propia.
  - Precondiciones:
    - La persona debe estar logueada.

        - La persona con necesidad debe tener cargada una necesidad.
        - La necesidad no puede estar en proceso de gestión.
    - Stakeholders interesados:
        - Personas con necesidades
    - Stakeholders activos:
        - Personas con necesidades
    - Pasos:
        - 1. La persona consulta sus necesidades solicitadas.
        - 2. La persona selecciona una de sus necesidades.
        - 3. La persona modifica la necesidad.
        - 4. La persona confirma el cambio de necesidad.
    - Resultado:
        - La persona cambia el tipo de necesidad que tenía inicialmente.

- **Caso 14:**
    - BE: Modificar Perfil.
    - PUC: Una persona edita su perfil (datos personales, habilidades).
    - Precondiciones:
        - La persona debe estar logueada.
        - La persona debe tener cargados sus datos.
        - En el caso de habilidades, no se pueden modificar las que estén en gestión (ayudando).
    - Stakeholders interesados:
        - Personas con habilidades
    - Stakeholders activos:
        - Personas con habilidades
    - Pasos:
        - 1. La persona consulta sus datos.
        - 2. La persona selecciona actualizar sus datos.
        - 3. La persona modifica los datos que crea desactualizados o que necesiten modificación.
        - 4. La persona confirma que los datos que ingreso sean válidos.
    - Resultado:
        - La persona logro cambiar los datos de su perfil.

Requisitos funcionales Snow Card

La Snow Card es una herramienta utilizada en el enfoque de requisitos Volere para documentar los requisitos de un sistema de manera concisa y estructurada. Fue introducida por James Robertson y Suzanne Robertson en su libro "Mastering the Requirements Process" (Dominando el Proceso de Requisitos).

Una Snow Card es una tarjeta que contiene información esencial sobre un requisito específico. Se utiliza como una forma práctica y visual de capturar los atributos clave de un requisito. Estos atributos incluyen:

Identificador: Un número o código único para identificar el requisito.

Nombre: Un título o nombre descriptivo para el requisito.

Prioridad: La importancia relativa del requisito en comparación con otros.

Descripción: Una explicación clara y concisa del requisito, que incluye los detalles necesarios para comprenderlo completamente.

Fuente: La persona o entidad que proporcionó el requisito.

Riesgo: Una evaluación del nivel de riesgo asociado con el requisito.

Estabilidad: Indica si el requisito es probable que cambie en el futuro.

Estado: El estado actual del requisito, como propuesto, aprobado, implementado, etc.

La Snow Card se utiliza como una herramienta de comunicación entre los diferentes interesados en el proyecto. Proporciona una forma rápida y clara de capturar y compartir información sobre los requisitos, lo que facilita la colaboración y el entendimiento común.

Es importante destacar que la Snow Card se complementa con otros artefactos de documentación de requisitos, como casos de uso, diagramas y especificaciones detalladas. La Snow Card sirve como una forma sucinta de representar la esencia del requisito y puede ser utilizada como un punto de partida para una discusión más detallada.

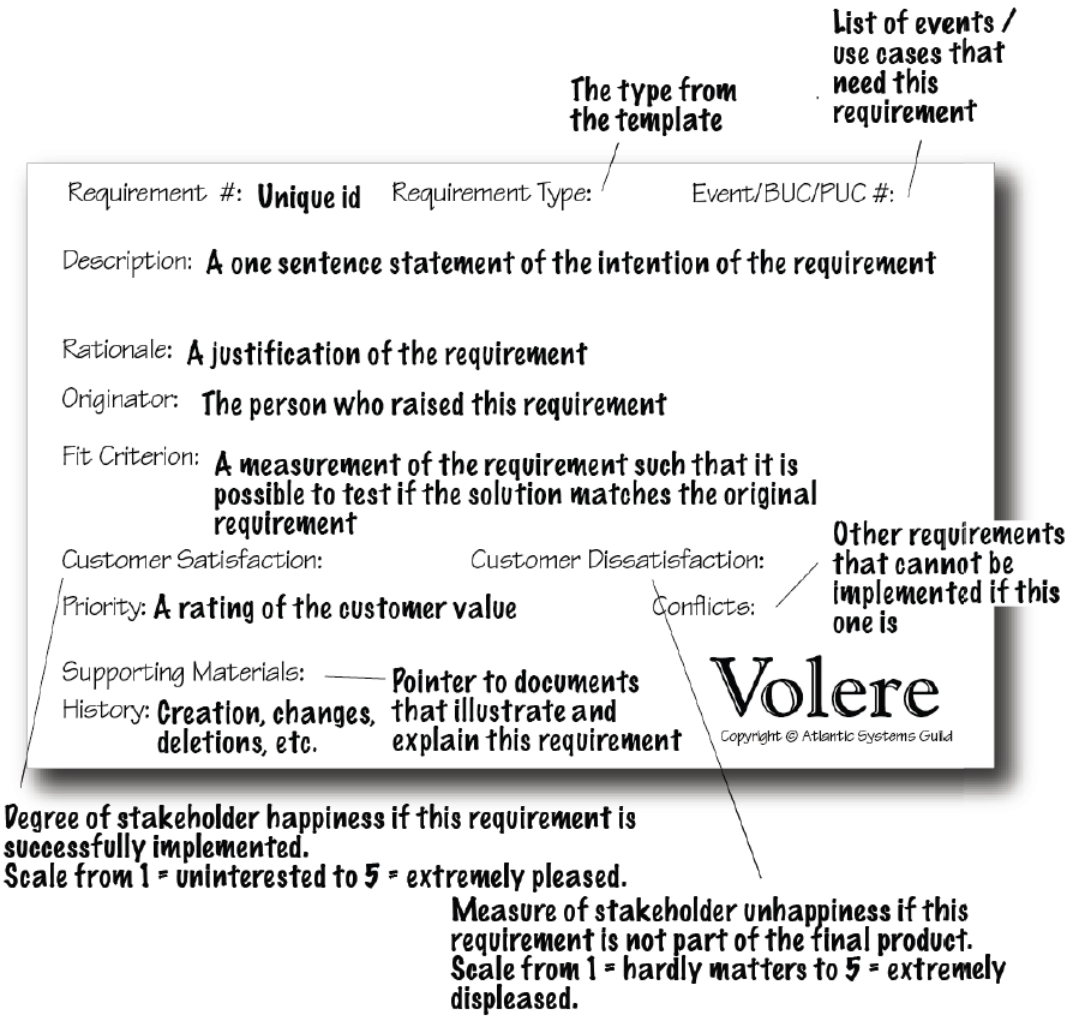


Imagen 1 Snow Card Volere, extraído de Mastering the Requirements Process

Requisitos del Sistema

Requisito	#1	Tipo	Funcional	PUC	#1
Descripción	La aplicación debe permitir que los usuarios se registren en ella, ingresando sus datos.				
Rationale	Esto permitirá a los usuarios poder logearse de forma segura para que no se vulnere su información y le permite al sistema contar con los datos de los usuarios para validar su identidad y tener información sobre ellos.				
Origen	Los usuarios				
Fit Criterion	Es posible acceder al formulario de registro desde la pantalla de inicio. Además, si no están completados los campos obligatorios, no debe ser posible realizar el registro.				
Satisfacción	4		Penalización	5	

Prioridad	4	Conflictos	-
Costo	4		
Material Soporte	Ley N°18.331, ley de protección de datos personales https://www.impo.com.uy/bases/leyes/18331-2008		
Historial	El requisito surge de evaluar y discutir cuando debería poder logearse el usuario a la aplicación, como modo de poder restringir los datos que se muestran al público.		
Campos obligatorios	Ci, nombre, apellidos, correo electrónico, contraseña, foto CI, teléfono		

Requisito	#2	Tipo	No Funcional	PUC	#1
Descripción	El formulario con los datos del usuario se deberá enviar en no más de 5 segundos.				
Rationale	Al usuario puede ocasionar insatisfacción si la aplicación demora en enviar el formulario.				
Origen	El espónsor (nosotros)				
Fit Criterion	El promedio de duración en las pruebas de carga no debe ser mayor a 5 segundos.				
Satisfacción	4		Penalización	3	
Prioridad	3			Conflictos	-
Costo	2				
Material Soporte	-				
Historial	El requisito parte de la idea que el sistema debe ser ágil para poder dejar satisfechos a quienes lo utilicen.				

Requisito	#3	Tipo	No Funcional	PUC	#
Descripción	El sistema debe verificar que los datos introducidos como el usuario y contraseña sean válidos y que estén correctos.				
Rationale	La seguridad de los datos de los usuarios depende de que este control se realice de la forma más optima.				
Origen	Los usuarios a quienes les preocupa su información.				
Fit Criterion	Al ingresar un dato no válido, el sistema debe emitir un aviso de error en alguno de los campos y no debe permitir continuar con la operación.				
Satisfacción	4		Penalización	4	
Prioridad	3			Conflictos	-

Costo	4
Material Soporte	Ley N°18.331, ley de protección de datos personales https://www.impo.com.uy/bases/leyes/18331-2008
Historial	El requisito surge de la realización de brainstorming inicial, donde se decidieron que datos debían ser considerados importantes.

Requisito	#4	Tipo	Funcional	PUC	#1
Descripción	El sistema debe enviar una notificación después del intento de creación de un usuario, sobre el resultado del registro.				
Rationale	El usuario final debe saber si se ha podido crear el usuario o en su defecto si tuvo algún error en la creación del mismo.				
Origen	El espónsor, el usuario.				
Fit Criterion	El sistema debe emitir una notificación sobre el estado del registro y validar que haya sido distribuida de forma correcta.				
Satisfacción	4		Penalización	3	
Prioridad	3			Conflictos	-
Costo	1				
Material Soporte	-				
Historial	Se investigo como poder realizar el envío de confirmación a través de correo electrónico, pero se descartó para hacer más ágil el sistema de registrar usuarios.				

Requisito	#5	Tipo	No Funcional	PUC	#1
Descripción	El sistema debe enviar la notificación de registro de forma ágil.				
Rationale	El sistema no debe demorar en comunicar a los usuarios el estado de su registro, el no hacerlo puede generar una mala experiencia.				
Origen	El espónsor, el usuario				
Fit Criterion	El promedio de duración en las pruebas de carga no debe ser mayor a 5 segundos.				
Satisfacción	2		Penalización	3	
Prioridad	2			Conflictos	-
Costo	3				
Material Soporte	-				
Historial	Esta RNF surge de que a la gente no le gusta esperar para poder seguir con los pasos en las App/Sistemas, entonces en el proceso de brainstorming un punto considerado fue hacer que la app fuera lo más ágil, acorde a los medios dados.				

Requisito	#6	Tipo	Funcional	PUC	#2
Descripción	El sistema debe permitir que los usuarios registrados inicien sesión ingresando sus credenciales.				
Rationale	Asegurar la seguridad de los usuarios y verificar la identidad de los mismos mientras usan la aplicación.				
Origen	El espónsor				
Fit Criterion	Solo las personas con usuarios en el sistema que ingresen sus correspondientes correos y contraseñas podrán ingresar a éste.				
Satisfacción	4		Penalización	5	
Prioridad	4			Conflictos	-

Costo	2
Material Soporte	OWASP, Test user registration process https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/03-Identity_Management_Testing/02-Test_User_Registration_Process
Historial	Este requisito surgió de la discusión en equipo sobre las medidas de seguridad que debe tener un usuario para proteger sus datos personales y las funcionalidades a las que tiene acceso dentro del sistema.
Campos	Ci , Contraseña

Requisito	#7	Tipo	No Funcional	PUC	#2
Descripción	El formulario de creación de usuario debe enviarse en no más de 5 segundos después de que el usuario lo complete y haga clic en "Enviar".				
Rationale	Al usuario puede ocasionar insatisfacción si la aplicación demora en enviar el formulario				
Origen	El espónsor				
Fit Criterion	Hacer pruebas de carga para evaluar que el tiempo de demora promedio este por debajo de los 5 seg.				
Satisfacción	5		Penalización	4	
Prioridad	3			Conflictos	-
Costo	2				
Material Soporte					
Historial	El requisito parte de la idea que el sistema debe ser ágil para poder dejar satisfechos a quienes lo utilicen.				

Requisito	#8	Tipo	Funcional	PUC	#2
Descripción	El sistema debe verificar que los datos introducidos en el formulario de creación de usuario (como el usuario y contraseña) sean válidos y que estén correctos.				
Rationale	Garantizar la integridad y calidad de los datos almacenados en la base de datos.				
Origen	El espónsor (nosotros)				
Fit Criterion	Al ingresar un dato no válido, el sistema debe emitir un aviso de error si la contraseña introducida no corresponde al usuario ingresado y no debe permitir continuar con la operación.				
Satisfacción	3		Penalización	4	
Prioridad	4			Conflictos	-
Costo	4				
Material Soporte	Cómo escoger una contraseña segura (y mantenerla): lo que dicen los expertos- https://www.xataka.com/basics/como-escoger-contrasena-segura-mantenerla-que-dicen-expertos				

Historial	Una vez hecha la investigación previa sobre las contraseñas y las validaciones de las mismas, llegamos a la conclusión que esta es la forma correcta de hacerlo.

Requisito	#9	Tipo	Funcional	PUC	#3
Descripción	El sistema permite al usuario ingresar una habilidad.				
Rationale	Al ser un sistema de cooperación mutua, en la que se busca conectar personas. Entendemos que todos los usuarios tienen que tener al menos una habilidad en la que se puedan registrar (todos pueden ofrecer algo). Repensar si TODAS las personas deberían tener una habilidad.				
Origen	El usuario solidario				
Fit Criterion	El sistema debe mostrar en el perfil del usuario la habilidad ingresada.				
Satisfacción	4		Penalización	4	
Prioridad	5			Conflictos	-
Costo	3				
Material Soporte					
Historial	Al evaluar las responsabilidades de los usuarios y el entorno en si del sistema, creímos conveniente que todos los usuarios puedan servirle a otro que tenga una necesidad, por tal motivo todos deben tener al menos una habilidad.				
Campos					

Requisito	#10	Tipo	Funcional	PUC	#4
Descripción	El sistema debe permitir al usuario registrar una necesidad.				
Rationale	En un sistema que busca conectar personas con habilidades, deben existir personas que tengan habilidades y otras con necesidades. Entonces el sistema debe permitir que aquellas personas que tengan necesidades puedan cargarlas.				
Origen	El usuario necesitado				
Fit Criterion	El sistema debe mostrar en el perfil de usuario la nueva necesidad ingresada.				
Satisfacción	4		Penalización	4	
Prioridad	5			Conflictos	-
Costo	3				
Material Soporte	-				
Historial	-				
Campos	-				

Requisito	#11	Tipo	Funcional	PUC	#4

Descripción	El sistema debe permitir cargar una fecha límite y lugar para una necesidad.		
Rationale	Las necesidades pueden ser algo planificado para un día determinado (una mudanza) y en algún lugar diferente a donde tiene su ubicación registrada (el usuario se está mudando).		
Origen	El usuario, el espónsor		
Fit Criterion	El sistema debe registrar una fecha (mayor o igual al día actual) y un lugar seleccionable en el mapa.		
Satisfacción	3	Penalización	3
Prioridad	5	Conflictos	-
Costo	3		
Material Soporte	-		
Historial	La funcionalidad surge de poder brindarle más confort a los usuarios al usar el sistema, a su vez es una funcionalidad que permite gestionar mejor las ayudas brindadas e identificar mejor las necesidades.		

Requisito	#12	Tipo	Funcional	PUC	#4
Descripción	El sistema debe permitir la creación de necesidades en un tiempo puntual y no puntual (un periodo).				
Rationale	Para contemplar necesidades que deben resolverse en una fecha en particular (como una mudanza)				
Origen	El espónsor				
Fit Criterion	Es posible crear una necesidad con un rango temporal (fecha de inicio y fecha de expiración) o con una fecha puntual.				
Satisfacción	3		Penalización	4	
Prioridad	3			Conflictos	-
Costo	2				
Material Soporte	-				
Historial	Refactorización de requisitos relacionados a rango temporal de las fechas de las necesidades.				

Requisito	#13	Tipo	Funcional	PUC	#4
Descripción	El sistema debe validar que el rango fechas de una necesidad no puntual sea válido.				
Rationale	Para que el rango temporal especificado se encuentre en el futuro y sus límites tengan sentido				
Origen	El espónsor				
Fit Criterion	Si el rango de fechas ingresado no es válido, se muestra un error.				
Satisfacción	3		Penalización	4	
Prioridad	3			Conflictos	-
Costo	1				
Material Soporte	-				
Historial	La funcionalidad surge de poder brindarle más confort a los usuarios al usar el sistema, a su vez es una funcionalidad que permite gestionar mejor las ayudas brindadas e identificar mejor las necesidades.				
Criterio	Fecha actual < Fecha de inicio < Fecha de fin				

Requisito	#14	Tipo	Funcional	PUC	#4

Descripción	El sistema debe validar que toda necesidad tenga asociada una dirección.			
Rationale	Se necesita conocer la ubicación de la necesidad para estimar la distancia de la persona que está dispuesta a ayudar con ella.			
Origen	Los usuarios			
Fit Criterion	La descripción del lugar debe ser aceptado por las API existentes en el mercado.			
Satisfacción	3	Penalización	4	
Prioridad	3		Conflictos	-
Costo	3			
Material Soporte	-			
Historial	La funcionalidad surge de poder brindarle más confort a los usuarios al usar el sistema, a su vez es una funcionalidad que permite gestionar mejor las ayudas brindadas e identificar mejor las necesidades.			

Requisito	#15	Tipo	Funcional	PUC	#5
Descripción	El sistema debe permitir acceder a los datos de contacto de los postulantes aprobados para una necesidad.				
Rationale	Las personas deben poder contactarse con la otra persona en una necesidad para coordinar el apoyo (cubrir la necesidad). Para eso el sistema debe brindarles a las personas el contacto con la otra persona.				
Origen	El espónsor				
Fit Criterion	El sistema debe mostrar el número de contacto/correo electrónico, para que la persona se pueda contactar y también debe mostrar la disponibilidad en que se ofrece la habilidad.				
Satisfacción	5		Penalización	5	
Prioridad	4			Conflictos	-
Costo	3				
Material Soporte	-				
Historial	Para implementar esta funcionalidad, creemos que es necesario que ambos usuarios puedan registrar que están participando en una necesidad para poder registrar que se cumplió (acuerdo de partes).				

Requisito	#16	Tipo	Funcional	PUC	#5
Descripción	El sistema debe permitir marcar una necesidad como resuelta.				
Rationale	Las necesidades que ya fueron resultas deben quedar registradas como completas/finalizadas para llevar un control y registro de necesidades resueltas.				
Origen	El espónsor				
Fit Criterion	El sistema crea un nuevo registro de necesidad resuelta. La necesidad no debe seguir figurando en el usuario solicitante.				
Satisfacción	5		Penalización	5	
Prioridad	4			Conflictos	-
Costo	3				
Material Soporte	-				
Historial	Para implementar esta funcionalidad, creemos que es necesaria, ya que debemos llevar un listado de actividades resueltas para saber que los usuarios participaron. Asimismo, llevar el registro de actividades en que participo cada usuario.				

Requisito	#17	Tipo	Funcional	PUC	#6
Descripción	El sistema debe pausar automáticamente una necesidad pendiente cuando se pasa de la fecha de expiración de esta.				
Rationale	Para no mostrar necesidades cuyo estado es incierto.				
Origen	Usuario con necesidades				
Fit Criterion	Al crear una necesidad, si en algún momento después de su fecha de expiración está pendiente, el estado de la necesidad en la aplicación debe ser observado como “Pausada”.				
Satisfacción	3		Penalización	3	
Prioridad	3			Conflictos	-
Costo	1				
Material Soporte	-				
Historial	-				

Requisito	#18	Tipo	Funcional	PUC	#6
Descripción	El sistema debe omitir las necesidades cuyo estado es “Pausada”, “Expirada” o “Completada” al realizar una búsqueda de necesidades.				
Rationale	Para no mostrar necesidades cuyo estado es incierto, ya no existen o ya fueron satisfechas.				
Origen	Usuario solidario				
Fit Criterion	Al realizar una búsqueda de necesidades, ninguna necesidad con los estados enumerados en la descripción debe aparecer en la búsqueda.				
Satisfacción	3		Penalización	5	
Prioridad	5			Conflictos	-
Costo	1				
Material Soporte	-				
Historial	-				

Requisito	#19	Tipo	Funcional	PUC	#6
Descripción	El sistema debe notificar a los usuarios con necesidades cuando una de sus necesidades expira.				
Rationale	Para que los usuarios puedan extender o eliminar una necesidad a la brevedad de su expiración.				

Origen	Usuario con necesidades			
Fit Criterion	Cuando una necesidad expira, el usuario dueño de esta debe tener una notificación visible en el sistema, indicando que una notificación expiró, y cuál.			
Satisfacción	4	Penalización	3	
Prioridad	3		Conflictos	-
Costo	2			
Material Soporte	-			
Historial	-			

Requisito	#20	Tipo	Funcional	PUC	#6
Descripción	La notificación de expiración de una necesidad debe redirigir al detalle de ésta cuando es clickeada.				
Rationale	Para que los usuarios puedan acceder rápidamente al detalle de una necesidad recientemente expirada y así accionar sobre ésta.				
Origen	Usuario con necesidades				
Fit Criterion	Al clickear la notificación de una necesidad expirada, debe abrirse el detalle de ésta.				
Satisfacción	4		Penalización	3	
Prioridad	3			Conflictos	-
Costo	1				
Material Soporte	-				
Historial	-				

Requisito	#22	Tipo	Funcional	PUC	#7
Descripción	El sistema debe permitir la búsqueda de personas con habilidades utilizando filtros.				
Rationale	Para que los usuarios con necesidades puedan buscar activamente a personas capaces de ayudarlos.				
Origen	Usuario con necesidades				
Fit Criterion	Al ingresar al sistema y realizar una búsqueda de personas con habilidades, se presenta una lista de usuarios solidarios que cumplen con los filtros especificados.				

Satisfacción	4	Penalización	5	
Prioridad	5		Conflictos	-
Costo	3			
Material Soporte	-			
Historial	-			
Campos Filtrables	Distancia, Habilidades (Multi seleccionable)			
Campos Mostrados	Nombre, Edad, Distancia, Habilidades			
Acciones Disponibles	Solicitar ayuda			
Consideraciones	Los usuarios que cumplen alguna de las siguientes condiciones no aparecerán en la funcionalidad de buscador: Usuarios sin dirección definida Usuarios que estén más lejos que su distancia configurada Usuarios que hayan desactivado el sistema de geolocalización (mejorar redacción)			

Requisito	# 23	Tipo	Funcional	PUC	#7
Descripción	El sistema debe mostrar los resultados de las búsquedas de personas en un mapa.				
Rationale	Para que los usuarios con necesidades puedan visualizar la distancia entre ellos y los usuarios con habilidades al mismo tiempo que ven el resto de su información.				
Origen	Usuario con necesidades				
Fit Criterion	Al ingresar al sistema y realizar una búsqueda de personas con habilidades, el resultado se presenta en un mapa, donde es posible visualizar marcadores correspondientes a los usuarios. Cuando un marcador es clickeado, muestra más información de su usuario.				
Satisfacción	5		Penalización	5	
Prioridad	5			Conflictos	# 24 Relación
Costo	5				
Material Soporte	-				
Historial	-				

Requisito	# 24	Tipo	Funcional	PUC	#7

Descripción	El sistema debe mostrar los resultados de las búsquedas de personas en una lista de cartas.			
Rationale	Para que los usuarios con necesidades puedan visualizar la información relevante de los resultados inmediatamente.			
Origen	Usuario con necesidades			
Fit Criterion	Al ingresar al sistema y realizar una búsqueda de personas con habilidades, el resultado se presenta en una lista de cartas, donde cada carta corresponde a un usuario solidario.			
Satisfacción	5	Penalización	5	
Prioridad	5		Conflictos	# 23
Costo	3			
Material Soporte	-			
Historial	-			

Requisito	# 24	Tipo	Funcional	PUC	#7
Descripción	El sistema debe permitir a los usuarios el envío de solicitudes de ayuda, en forma de notificaciones que redirigen a la necesidad.				
Rationale	Para que los usuarios con necesidades puedan solicitar ayuda de forma activa y fluida.				
Origen	Usuario con necesidades				
Fit Criterion	Al realizar una solicitud de ayuda, el usuario solidario objetivo debe recibir una notificación que lo redirige hacia el detalle de la necesidad.				
Satisfacción	5		Penalización	4	
Prioridad	4			Conflictos	-
Costo	3				
Material Soporte	-				
Historial	-				
Excepciones	El destinatario				

Requisito	# 25	Tipo	Funcional	PUC	#8
Descripción	El sistema debe permitir la búsqueda de necesidades utilizando filtros.				
Rationale	Para que los usuarios solidarios puedan buscar activamente a personas con necesidades.				
Origen	Usuario solidario				

Fit Criterion	Al ingresar al sistema y realizar una búsqueda de necesidades, se presenta una lista de necesidades que el usuario actual es capaz de resolver.			
Satisfacción	5	Penalización	5	
Prioridad	5		Conflictos	-
Costo	5			
Material Soporte	-			
Historial	-			
Campos Filtrables	Descripción, Distancia, Habilidades Requeridas (Multi seleccionable, las opciones son las habilidades del usuario actual)			
Campos Mostrado	Descripción, Distancia, Nombre del Dueño, Habilidades Requeridas			
Acción Disponible	Ofrecer ayuda			

Requisito	# 26	Tipo	Funcional	PUC	#8
Descripción	El sistema debe permitir a los usuarios el envío de postulaciones de ayuda.				
Rationale	Para que los usuarios solidarios puedan ofrecer ayuda de forma activa y fluida.				
Origen	Usuario solidario				
Fit Criterion	Al realizar una postulación de ayuda, la necesidad objetivo debe añadir a ese usuario como potencial ayudante.				
Satisfacción	5		Penalización	5	
Prioridad	5			Conflictos	-
Costo	2				
Material Soporte	-				
Historial	-				

Requisito	# 27	Tipo	Funcional	PUC	#8
Descripción	El sistema debe mostrarle al dueño de una necesidad la lista de postulantes solidarios en el detalle de esta.				
Rationale	Para que los usuarios con necesidades puedan elegir el postulante más apropiado.				
Origen	Usuario con necesidades				

Fit Criterion	En el detalle de una necesidad debe poder observarse una lista de postulantes con los usuarios solidarios que se postularon a la necesidad. Los usuarios que aparecen en la lista coinciden con los datos de los postulantes existentes.			
Satisfacción	4	Penalización	3	
Prioridad	4		Conflictos	-
Costo	1			
Material Soporte	06/05: Cambio rationale: de “Para que los usuarios con necesidades puedan observar sus postulantes” a “Para que los usuarios con necesidades puedan elegir el postulante más apropiado”			
Historial				
Campos	Nombre, Edad, Habilidades, Distancia, Estado de postulación			
Acciones	Aceptar postulación, Rechazar postulación			

Requisito	# 28	Tipo	Funcional	PUC	#8
Descripción	El sistema debe ordenar a los postulantes para ayudar en una necesidad por su potencial calidad de ayuda.				
Rationale	Para que los usuarios con necesidades puedan observar a los postulantes en el orden más relevante para su necesidad.				
Origen	Usuario con necesidades				
Fit Criterion	Al observar la lista de postulantes, el orden de éstos debe ser basado en la calidad de ayuda calculada, de mayor a menor.				
Satisfacción	3		Penalización	2	
Prioridad	2			Conflictos	-
Costo	2				
Material Soporte	-				
Historial	-				
Fórmula de Cálculo de Calidad	$\text{Calidad} = (\#\text{Habilidades Relevantes} * 1000) - \text{Distancia (m)}$				

Requisito	# 29	Tipo	Funcional	PUC	#9
Descripción	El sistema debe permitir la aprobación de las postulaciones de ayuda.				
Rationale	Para que los usuarios con necesidades puedan confirmar que la ayuda ofrecida será utilizada.				

Origen	Usuario con necesidades			
Fit Criterion	Al aprobar una postulación, el estado de esta debe ser actualizado a “Aprobada” y el usuario solidario objetivo debe recibir una notificación.			
Satisfacción	5	Penalización	5	
Prioridad	5		Conflictos	-
Costo	2			
Material Soporte	-			
Historial	-			

Requisito	# 30	Tipo	Funcional	PUC	#9
Descripción	El sistema debe mostrar la ubicación de una necesidad y la información de contacto de su dueño exclusivamente a los postulantes aprobados.				
Rationale	Para preservar la privacidad de los usuarios con necesidades hasta que la ayuda ofrecida es aceptada.				
Origen	Usuario con necesidades				
Fit Criterion	Al visualizar el detalle de una necesidad desde un usuario que no tiene una postulación aprobada a ésta, la ubicación de ésta y la información de contacto del dueño no es visible. Al realizar la misma prueba con un postulante aprobado, dicha información debe ser visible.				
Satisfacción	5		Penalización	5	
Prioridad	5			Conflictos	-
Costo	1				
Material Soporte	-				
Historial	-				

Requisito	# 31	Tipo	Funcional	PUC	#10
Descripción	El sistema debe permitir el rechazo de las postulaciones de ayuda.				
Rationale	Para que los usuarios con necesidades puedan confirmar que la ayuda ofrecida no será utilizada.				
Origen	Usuario con necesidades				
Fit Criterion	Al rechazar una postulación, el estado de esta debe ser actualizado a “Rechazada” y el usuario solidario objetivo debe recibir una notificación.				
Satisfacción	5		Penalización	5	

Prioridad	5	Conflictos	-
Costo	2		
Material Soporte	-		
Historial	-		

Requisito	# 32	Tipo	Funcional	PUC	#11
Descripción	El producto debe permitir al usuario buscar otros usuarios en base a varios criterios de búsqueda.				
Criterios de búsqueda	Por nombre, por nacionalidad, por distancia respecto al usuario que busca				
Rationale	Para que un usuario con necesidades pueda encontrar usuarios con habilidades.				
Origen	Los usuarios con necesidades				
Fit Criterion	Precondiciones: El usuario debe estar registrado Pasos: El usuario accede a la funcionalidad de buscador El usuario inserta los datos necesarios Palabras clave Filtros El usuario procede a la acción de buscar Resultado: El producto muestra los usuarios correspondientes a las palabras clave y filtros de búsqueda				
Satisfacción	1		Penalización	5	
Prioridad	5			Conflictos	-
Costo	3				
Material Soporte	-				
Historial	-				
Detalles	Los usuarios que cumplen alguna de las siguientes condiciones no aparecerán en la funcionalidad de buscador: Usuarios sin dirección definida Usuarios que estén más lejos que su distancia de geolocalización configurada Usuarios que hayan desactivado el sistema de geolocalización (mejorar redacción) Usuarios sin dirección definida				

Requisito	# 33	Tipo	Funcional	PUC	11
Descripción	El sistema debe permitir al usuario con habilidad estipular la distancia a la cual está dispuesto a ayudar.				
Rationale	Para que el usuario pueda evitar que reciba notificaciones de ayuda de gente que está demasiado lejos.				
Origen	Los usuarios				
Fit Criterion	Precondiciones: El usuario debe estar registrado Pasos: El usuario accede a la sección de geolocalización El usuario edita la distancia de geolocalización El usuario confirma la edición Resultado: La distancia de geolocalización fue cambiada por el usuario				
Satisfacción	2		Penalización	4	
Prioridad	4			Conflictos	-
Costo	1				
Material Soporte	Ninguno				
Historial	-				

Requisito	# 34	Tipo	Funcional	PUC	11
Descripción	El producto debe permitirle al usuario insertar, modificar o eliminar su dirección.				
Rationale	Para medir la distancia de los usuarios entre sí y así mostrar necesidades y usuarios cercanos.				
Origen	Los usuarios				
Fit Criterion	Precondiciones: El usuario debe estar registrado Pasos: El usuario accede a su perfil El usuario edita la dirección El usuario confirma la edición Resultado: La dirección está modificada				
Satisfacción	1		Penalización	4	
Prioridad	3			Conflictos	-
Costo	1				

Material Soporte	-
Historial	-

Requisito	# 35	Tipo	Funcional	PUC	11
Descripción	El producto debe permitir activar o desactivar el sistema de geolocalización.				
Rationale	Para que los usuarios solidarios puedan decidir si compartir su ubicación en el sistema o no.				
Origen	Los usuarios				
Fit Criterion	Cuando un usuario desactiva el sistema de geolocalización, deja de ser visible en búsquedas de usuarios con habilidades y el sistema deja de recuperar datos de su ubicación. De lo contrario, debe mantenerse visible y el sistema debe seguir recuperando datos de su ubicación.				
Satisfacción	3		Penalización	5	
Prioridad	3			Conflictos	-
Costo	1				
Material Soporte	-				
Historial	-				

Requisito	# 36	Tipo	Funcional	PUC	13
Descripción	El producto debe permitir al usuario modificar sus necesidades.				
Rationale	Porque es posible que ciertas necesidades cambien con el tiempo y, por lo tanto, el usuario debe ser capaz de reflejar dichos cambios en el producto.				
Origen	Los usuarios				
Fit Criterion	Precondiciones: El usuario debe estar registrado La necesidad no puede estar “en proceso” Pasos: El usuario accede a su lista de necesidades El usuario edita su necesidad El usuario confirma la edición Resultado: La necesidad está editada				
Satisfacción	3		Penalización	4	
Prioridad	3			Conflictos	-

Costo	3
Material Soporte	-
Historial	-

Requisito	# 37	Tipo	Funcional	PUC	13
Descripción	El sistema debe enviar una notificación a todos los postulantes de una necesidad si la misma es editada y debe modificar el estado de las postulaciones a “Suspendida”.				
Rationale	Para que los postulantes no se comprometan con una necesidad que es diferente a la que pensaban cuando se postularon.				
Origen	Los usuarios				
Fit Criterion	Al editar una necesidad con postulantes, todos los postulantes relacionados deben recibir una notificación indicando los cambios y solicitando la confirmación de su colaboración.				
Satisfacción	3		Penalización	5	
Prioridad	3			Conflictos	-
Costo	3				
Material Soporte	-				
Historial	-				

Requisito	# 38	Tipo	Funcional	PUC	13
Descripción	El sistema debe permitir que los usuarios solidarios confirmen o eliminen su postulación a una necesidad.				
Rationale	Para que los postulantes puedan cancelar sus postulaciones y confirmar sus postulaciones suspendidas.				
Origen	Los usuarios				
Fit Criterion	Al tener una postulación a una necesidad, el usuario puede eliminarla, lo cual remueve la postulación a esa necesidad permanentemente. Al tener una postulación suspendida, el usuario puede confirmarla nuevamente, modificando su estado a “Pendiente”.				
Satisfacción	4		Penalización	4	
Prioridad	4			Conflictos	-
Costo	3				
Material Soporte	-				

Historial	-

Requisito	# 39	Tipo	Funcional	PUC	13
Descripción	El producto debe permitir al usuario ver sus necesidades con su descripción, ordenadas descendentemente por su fecha de creación, y acceder al detalle de las mismas clickéandolas.				
Rationale	Para que el usuario pueda añadir, modificar y eliminar necesidades sin tener que recordarlas o guardar dicha información en otro lado.				
Origen	Los usuarios				
Fit Criterion	Al acceder a su lista de necesidades, el usuario puede ver todas sus necesidades ordenadas por fecha de creación desde la más reciente hasta la más antigua. Al clickear una necesidad, el usuario debe ser redirigido al detalle de la misma.				
Satisfacción	4		Penalización	5	
Prioridad	4			Conflictos	-
Costo	2				
Material Soporte	-				
Historial	-				

Requisito	# 40	Tipo	Funcional	PUC	14
Descripción	El sistema debe permitir a un usuario modificar los datos de su perfil.				
Rationale	Es muy probable que ciertos datos del usuario (como los correos electrónicos) cambien con el tiempo, por lo que el usuario debe poder reflejar dichos cambios en el producto.				
Origen	Los usuarios				
Fit Criterion	Al editar el perfil de un usuario, los datos observados deben reflejar los ingresados durante la edición.				
Satisfacción	2		Penalización	5	
Prioridad	5			Conflictos	-
Costo	3				
Material Soporte	-				
Historial	-				

Requisito	# 41	Tipo	Funcional	PUC	14
Descripción	El sistema debe permitir a un usuario cambiar su contraseña.				
Rationale	Es una muy buena práctica cambiar su contraseña cada cierto tiempo para facilitar un cierto nivel de seguridad de la cuenta.				
Origen	Los usuarios				
Fit Criterion	Precondiciones: El usuario debe estar registrado El usuario debe recordar su contraseña antigua Pasos: El usuario accede a su perfil El usuario procede a cambiar la contraseña El usuario inserta la nueva contraseña El usuario confirma el cambio Resultado: La contraseña ha sido cambiada				
Satisfacción	2		Penalización	4	
Prioridad	4			Conflictos	-
Costo	2				
Material Soporte	-				
Historial	-				

Modelo Entidad Relación

Una entidad, en el contexto del mundo real, se refiere a un objeto o concepto que puede ser claramente identificado y distinguido de otros objetos. Estas entidades pueden ser personas, objetos físicos, eventos o cualquier otro elemento que exista en nuestro entorno.

Un conjunto de entidades se compone de varias entidades del mismo tipo que comparten características o atributos similares. Estos atributos son las propiedades que describen y definen a cada miembro del conjunto de entidades.

La representación de una entidad se realiza a través de un conjunto de atributos, que son características específicas de esa entidad. Estos atributos pueden ser cualidades físicas, atributos de comportamiento o cualquier otra propiedad relevante para la entidad en cuestión. Cada entidad tiene un valor asignado a cada uno de sus atributos, lo que permite diferenciarlas y distinguirlas entre sí. Por ejemplo, si consideramos una entidad "persona", algunos de sus atributos podrían ser el nombre, la edad, la dirección y el número de teléfono, entre otros.

Cada atributo tiene un conjunto de valores permitidos, conocido como dominio, que establece las opciones válidas para ese atributo en particular. Por ejemplo, el atributo "género" podría tener como dominio los valores "masculino" y "femenino".

En el modelo E-R (Entidad-Relación), los atributos se clasifican en diferentes tipos:

- Atributos simples y compuestos: Los atributos simples hacen referencia a un solo valor indivisible, como el atributo "edad". Por otro lado, los atributos compuestos se subdividen en sub-atributos más pequeños y separados, como el atributo "dirección" que puede tener sub-atributos como "calle", "número" y "ciudad".
- Atributos monovalorados y multivalorados: Los atributos monovalorados solo pueden tener un único valor asignado, como el atributo "número de teléfono". En contraste, los atributos multivalorados son capaces de aceptar varios valores, como el atributo "hobbies" que podría contener múltiples opciones.
- Atributos derivados: Este tipo de atributo obtiene su valor a partir de otros atributos o entidades relacionadas. Por ejemplo, un atributo derivado podría ser "edad promedio" calculado a partir de las edades individuales de un grupo de entidades relacionadas.

Una relación es una asociación o conexión establecida entre diferentes entidades. Estas entidades pueden ser personas, objetos, eventos o cualquier otro elemento que sea relevante dentro de un contexto específico.

Un conjunto de relaciones se compone de varias relaciones del mismo tipo. Esto implica que comparten características y propiedades similares, lo que les permite agruparse y ser tratadas de manera conjunta.

La correspondencia de cardinalidades, también conocida como razón de cardinalidad, es un concepto fundamental en la descripción de las relaciones entre entidades. Esta correspondencia determina el número de entidades que pueden estar asociadas a través de un conjunto de relaciones.

La correspondencia de cardinalidades es especialmente útil cuando se describen conjuntos de relaciones binarias, donde solo se involucran dos conjuntos de entidades. Sin embargo, también puede aplicarse a conjuntos de relaciones que implican más de dos conjuntos de entidades, lo que amplía su utilidad y aplicación.

Para un conjunto de relaciones binarias R entre los conjuntos de entidades A y B, la correspondencia de cardinalidades puede adoptar una de las siguientes formas:

- Uno a uno: Cada entidad en el conjunto A se asocia como máximo con una entidad en el conjunto B, y viceversa. Esta correspondencia asegura que la asociación entre las entidades sea única y exclusiva.
- Uno a varios: Cada entidad en el conjunto A puede asociarse con cualquier número de entidades en el conjunto B, ya sea ninguna, una o varias. Sin embargo, una entidad en el conjunto B solo puede estar asociada como máximo con una entidad en el conjunto A.
- Varios a uno: Cada entidad en el conjunto A se puede asociar como máximo con una entidad en el conjunto B. Por otro lado, una entidad en el conjunto B puede estar asociada con cualquier número de entidades en el conjunto A, ya sea ninguna, una o varias.
- Varios a varios: Cada entidad en el conjunto A puede asociarse con cualquier número de entidades en el conjunto B, al igual que cada entidad en el conjunto B puede asociarse con cualquier número de entidades en el conjunto A. Esta correspondencia permite que múltiples asociaciones se establezcan entre los conjuntos de entidades.

Diagrama Entidad Relación

La estructura lógica de una base de datos se puede visualizar de manera gráfica mediante el uso de un diagrama E-R (Entidad-Relación). Estos diagramas son ampliamente utilizados debido a su simplicidad y claridad, lo que facilita la comprensión y comunicación de la estructura de la base de datos.

Un diagrama E-R consta de varios componentes principales que representan diferentes elementos de la base de datos:

- **Rectángulos**: Estos representan conjuntos de entidades, es decir, agrupaciones de entidades que comparten características similares.

- **Elipses**: Las elipses se utilizan para representar los atributos de las entidades. Los atributos son las propiedades o características que describen a cada miembro de un conjunto de entidades.
- **Rombos**: Los rombos en el diagrama E-R representan las relaciones entre los conjuntos de entidades. Estas relaciones indican cómo se conectan las entidades y cómo se relacionan entre sí.
- **Líneas**: Las líneas se utilizan para conectar los atributos a los conjuntos de entidades y los conjuntos de entidades a los conjuntos de relaciones. Estas conexiones representan las asociaciones y dependencias existentes en la base de datos.
- **Elipses dobles**: Las elipses dobles se utilizan para denotar atributos multivalorados, es decir, aquellos atributos que pueden tener múltiples valores para una entidad en particular.
- **Elipses discontinuas**: Las elipses discontinuas se utilizan para representar atributos derivados. Estos atributos se obtienen o calculan a partir de otros atributos o entidades relacionadas.
- **Líneas dobles**: Las líneas dobles indican la participación total de una entidad en un conjunto de relaciones. Esto significa que todas las entidades del conjunto deben estar relacionadas con al menos una entidad del otro conjunto en la relación.
- **Rectángulos dobles**: Los rectángulos dobles representan conjuntos de entidades débiles. Estas entidades dependen de otras entidades para existir y no pueden ser identificadas de manera única sin la existencia de la entidad principal a la que están vinculadas.

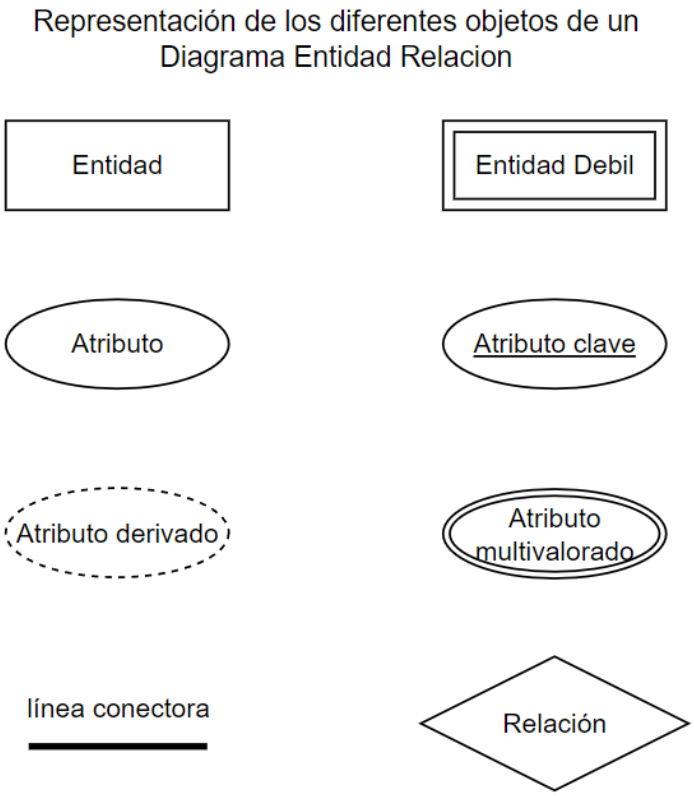


Imagen 2 Representación de los objetos del Diagrama ER, realizados en Draw.io

Modelo entidad relación del proyecto.

Usuario (CI, NOMBRE, APELLIDO, EMAIL, LATITUD Y LONGITUD(CIUDAD, DEPARTAMENTO, DIRECCION), TELEFONO, FOTO CI, CONFIGURACION_GEO(DISTANCIA, ACTIVADA))

Necesidad (ID, TITULO, ESTADO, DESCRIPCION, FECHA CREACION, UBICACIÓN(LAT, LONG), FECHA RESUELTO)

Habilidad (NOMBRE)

Solidario(ext Usuario) tiene Habilidad (N:N) (descripción, fecha creación)

Solidario (ext Usuario) se postula para Necesidad (N:N) (estado, fecha creación)

Necesitado(ext Usuario) tiene Necesidad (1:N)

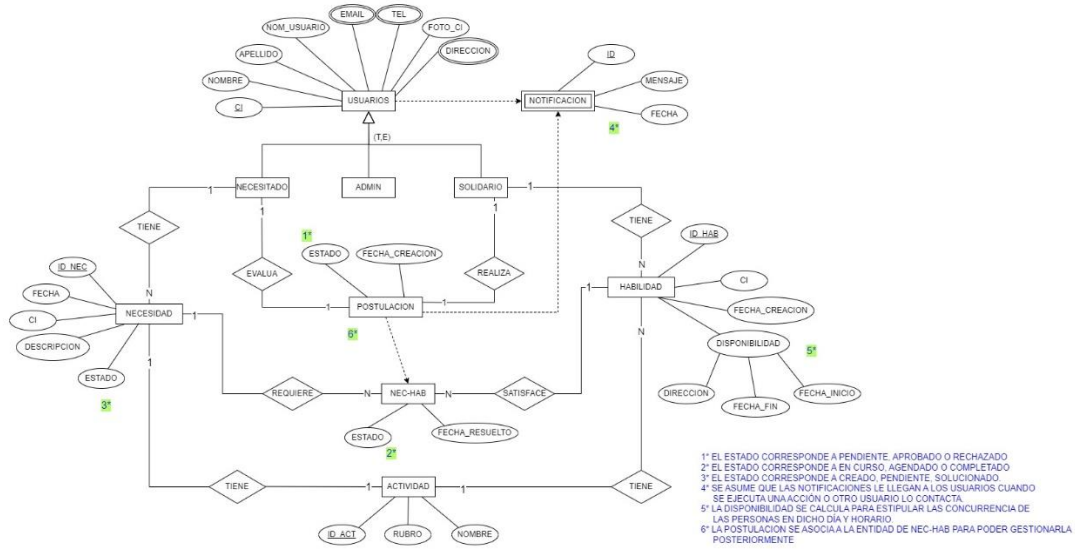
Necesidad requiere Habilidad (N:N)

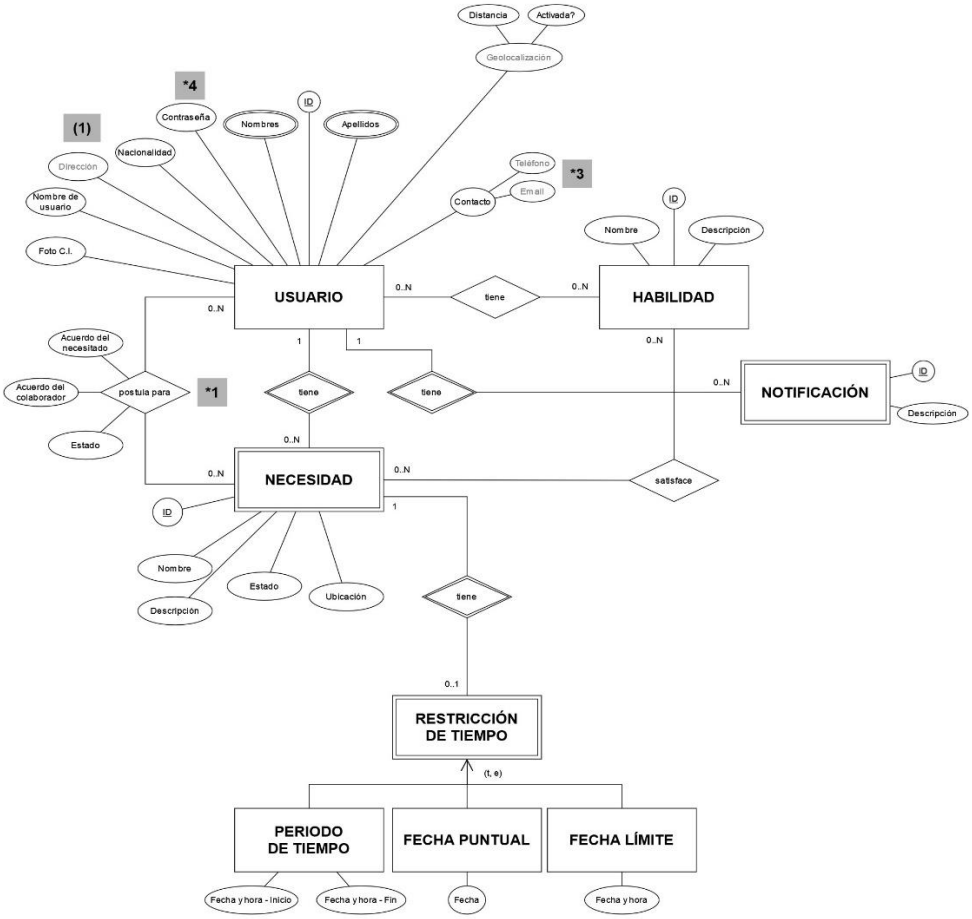
Necesidad temporal (fecha inicio, fecha fin) Agrega a Necesidad

Necesidad Puntual (fecha) Agrega a Necesidad

Diagrama Entidad Relación del Proyecto

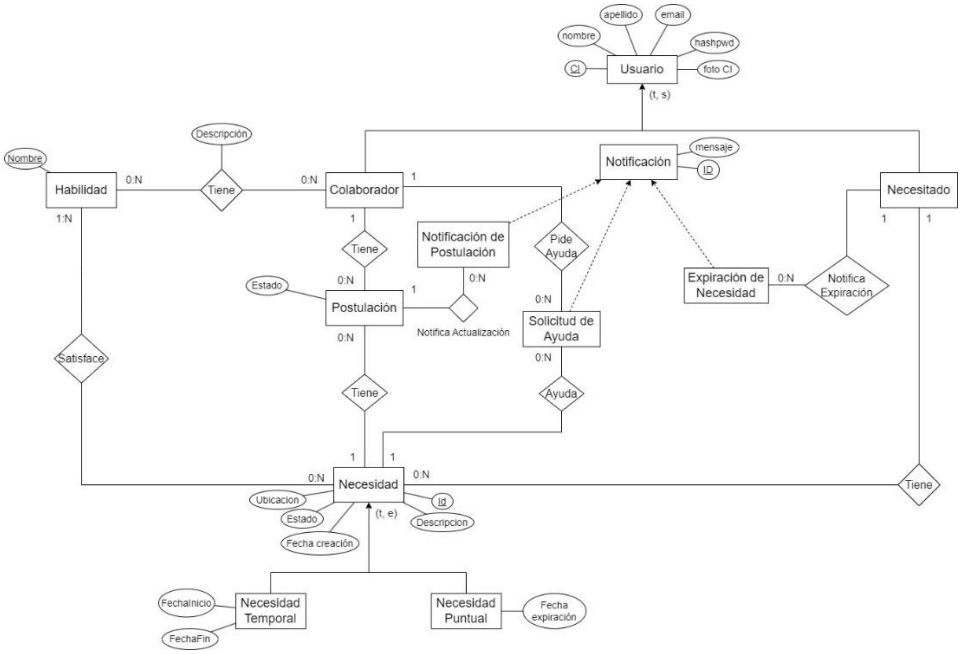
Antecedentes del MER





Normas informales:
*1: Un usuario que postula para con una necesidad de otro usuario debe tener una habilidad que la satisfaga.
*1: Un usuario no puede postular para una necesidad del mismo.
*1: Los atributos "acuerdo del colaborador" y "acuerdo del necesitado" representan si ambas partes están de acuerdo en la colaboración. Pueden tener tres valores: "aprobado", "rechazado" o "pendiente".
*2: Los atributos escritos en color gris son opcionales.
*3: Todo usuario debe tener al menos una información de contacto.
*4: La contraseña debe estar protegida (hash, salt, etc.)

Bitácora:
(1): Se asume que los usuarios solo pueden tener una dirección.
(1): Se asume que los usuarios pueden no tener una dirección.



Explicación de estos modelos presentados

En esta sección, se describirá el proceso seguido por el grupo para desarrollar el modelo de diseño entidad-relación, así como los factores que se tuvieron en cuenta durante dicho proceso. El trabajo presentado anteriormente es el resultado del esfuerzo conjunto de los miembros del grupo, quienes diseñaron el modelo en función de su comprensión del modelo de negocio. Este enfoque se basó en los requisitos funcionales y no funcionales previamente definidos en el documento.

Para crear el modelo, se eligió utilizar la herramienta de diseño de diagramas de entidad-relación de la plataforma Draw.io. Esta herramienta permitió a los miembros del grupo elaborar los diagramas de manera fácil y rápida. Además, brindó la flexibilidad de realizar modificaciones en cualquier momento y de manera colaborativa. También se destacó la capacidad de exportar los diagramas en diferentes formatos.

El uso de la herramienta Draw.io facilitó el proceso de diseño y permitió a los miembros del grupo visualizar y comunicar eficientemente las relaciones y entidades involucradas en el modelo. Esta elección se hizo teniendo en cuenta la simplicidad y la posibilidad de colaboración que proporciona la herramienta.

Modelo final del Diagrama Entidad Relación

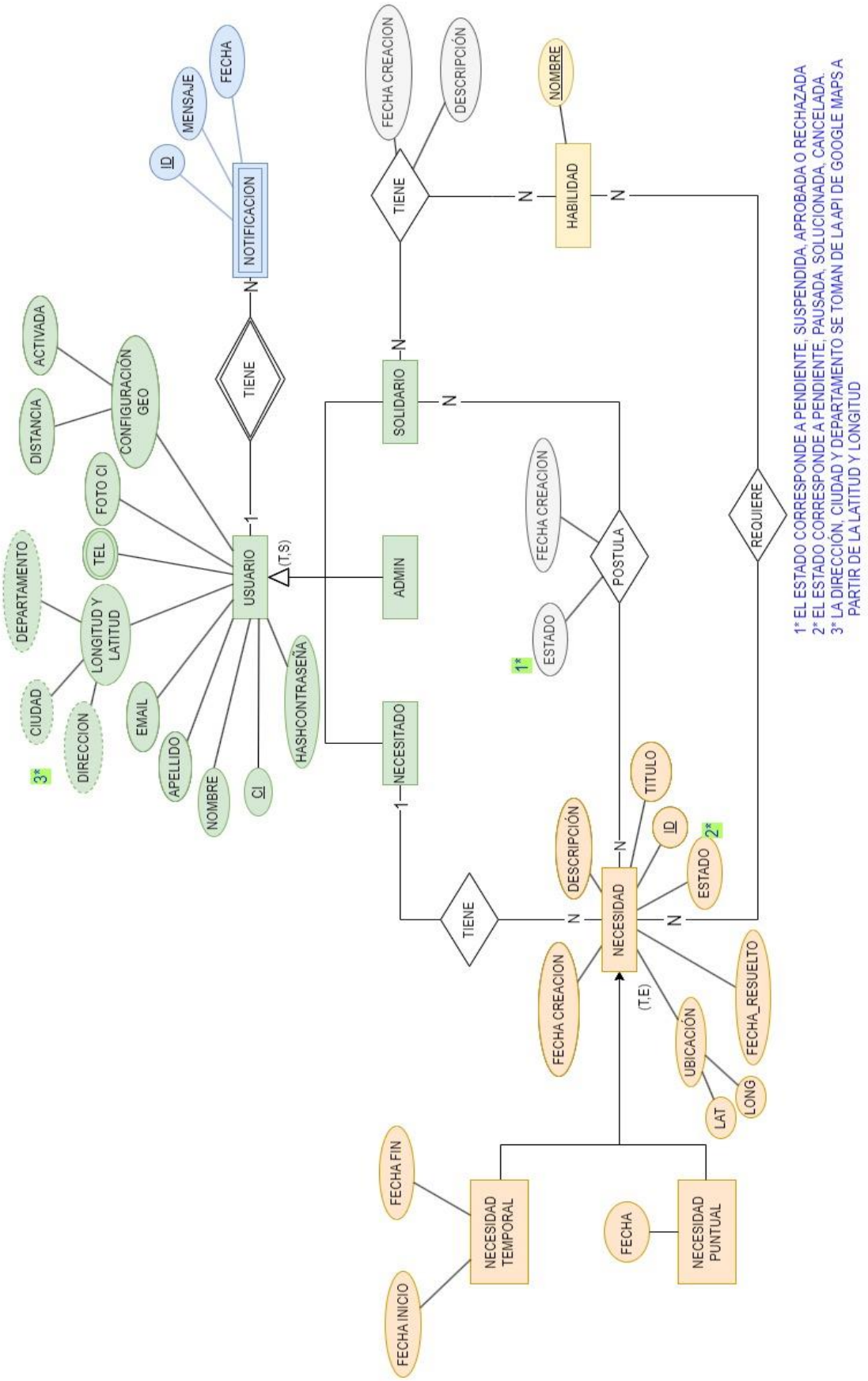


Imagen 3 Diagrama Entidad-Relación, realizado en Draw.io

Explicación del Diseño Entidad Relación Final

Después de un proceso de trabajo conjunto, el grupo logró finalizar con éxito el modelo de diseño entidad-relación que se muestra en la parte superior de esta sección. Durante esta etapa, se llevó a cabo la tarea de compartir y analizar individualmente los modelos de diseño entidad-relación propuestos por cada integrante del grupo. El objetivo principal era unificar estos diferentes modelos en uno solo que fuera lo más completo posible y que cumpliera con los requisitos funcionales y no funcionales establecidos previamente en el presente documento.

Durante la revisión y comparación de los modelos existentes, surgieron algunas diferencias y discrepancias entre ellos. Estas diferencias representaron desafíos que debieron abordarse y resolver de manera colaborativa. Para lograr un modelo final coherente y completo, fue necesario realizar ajustes y, en algunos casos, incluso rediseñar partes del flujograma. El objetivo era garantizar que el modelo resultante fuera adecuado para abordar de manera efectiva y precisa el problema planteado.

Este proceso de unificación y resolución de diferencias permitió al grupo generar un modelo de diseño entidad-relación que consideramos apropiado y óptimo para la solución del problema en cuestión. Se prestó especial atención a la integridad y la exhaustividad del modelo, asegurándose de que cumpliera con todos los requisitos y necesidades identificados previamente. La colaboración y el esfuerzo conjunto del grupo fueron fundamentales para alcanzar un resultado final sólido y satisfactorio.

Modelo Lógico

Considerando que la base de datos es relacional, describimos las transformaciones y decisiones de diseño que debemos aplicar al modelo conceptual para producir un modelo lógico adecuado para su implementación directa como una base de datos relacional. Es posible que sea necesario realizar algunos cambios en este modelo relacional inicial para alcanzar los objetivos de rendimiento; con este fin, crearemos un modelo de datos físico.

Las ventajas de producir un modelo de datos lógico como un entregable intermedio en lugar de proceder directamente al modelo de datos físico son las siguientes:

1. Dado que se ha generado mediante un conjunto de transformaciones bien definidas a partir del modelo de datos conceptual, el modelo de datos lógico refleja los requisitos de información empresarial sin verse afectado por los cambios necesarios para mejorar el rendimiento. En particular, incorpora reglas sobre las propiedades de los datos, como las dependencias funcionales (descritas en la Sección 2.8.1). Estas reglas no siempre se pueden

deducir de un modelo de datos físico, que puede haber sido desnormalizado u comprometido de alguna manera.

2. Si la base de datos se migra a otro sistema de gestión de bases de datos (DBMS) que admita estructuras similares (por ejemplo, otro DBMS relacional o una nueva versión del mismo DBMS con diferentes propiedades de rendimiento), el modelo de datos lógico se puede utilizar como referencia para el nuevo modelo de datos físico.

La tarea de transformar el modelo de datos conceptual a un modelo lógico relacional es bastante sencilla, ciertamente más que la etapa de modelado conceptual, incluso para modelos grandes. Durante este proceso, realizamos varias transformaciones, algunas de las cuales admiten alternativas y requieren tomar decisiones, mientras que otras son puramente mecánicas.

Modelo Lógico Final del Proyecto

USUARIO (CI, NOMBRE, APELLIDO, FOTO_CI, HASHPWD, EMAIL, GEO_DISTANCIA, GEO_ACTIVADO, ES_ADMIN, LATITUD, LONGITUD)
FK (DEPARTAMENTO --> DEPARTAMENTO(NOMBRE)

USUARIO_TELEFONO (CI_USUARIO, TELÉFONO)
FK (CI_USUARIO --> USUARIO(CI)

HABILIDAD (NOMBRE)

USUARIO_TIENE_HABILIDAD (NOMBRE_HAB, CI, FECHA_CREACIÓN, DESCRIPCION)
FK (CI --> USUARIO(CI), NOMBRE_HAB --> HABILIDAD(NOMBRE))

ESTADO_NECESIDAD(NOMBRE)

NECESIDAD (ID, CI_CREADOR, TITULO, DESCRIPCION, ESTADO, FECHA_CREACIÓN, LATITUD, LONGITUD, FECHA_INICIO, FECHA_FIN, FECHA_SOLUCIONADA)
FK (CI_ CREADOR --> USUARIO(CI), ESTADO --> ESTADO_NECESIDAD(NOMBRE))

NECESIDAD_REQUIERE_HABILIDAD (NOMBRE_HABILIDAD, ID_NECESIDAD)
FK (NOMBRE_HABILIDAD --> HABILIDAD(NOMBRE), ID_NECESIDAD –> NECESIDAD(ID))

ESTADO_POSTULACION(NOMBRE)

POSTULACION (ID_NECESIDAD, CI_POSTULANTE, ESTADO, FECHA_CREACION)
FK (ID_NECESIDAD –> NECESIDAD(ID), CI_POSTULANTE --> USUARIO(CI), ESTADO --> ESTADO_POSTULACION(NOMBRE))

NOTIFICACIÓN (ID, CI_USUARIO, MENSAJE, FECHA)
FK (CI_USUARIO --> USUARIO(CI))

Aclaraciones:

- El atributo DIRECCIÓN de la tabla USUARIO es número de puerta, calle, y número de apartamento si necesario.
- El atributo MENSAJE de la tabla NOTIFICACIÓN corresponde al resultado de una operación o recordatorios del sistema.
- Los registros de la tabla ESTADO_POSTULACIÓN son precargados y corresponden a los siguientes valores:
  - Pendiente,
  - Suspendida,
  - Aprobada,
  - Rechazada
- Los registros de la tabla ESTADO_NECESIDAD son precargados y corresponden a los siguientes valores:
  - Pendiente,
  - Pausada,
  - Solucionada,
  - Cancelada
- El atributo FECHA_SOLUCIONADA de la tabla NECESIDAD se autocompleta cuando el estado de esta pasa a ser “Solucionada”.

Diseño del Modelo Lógico

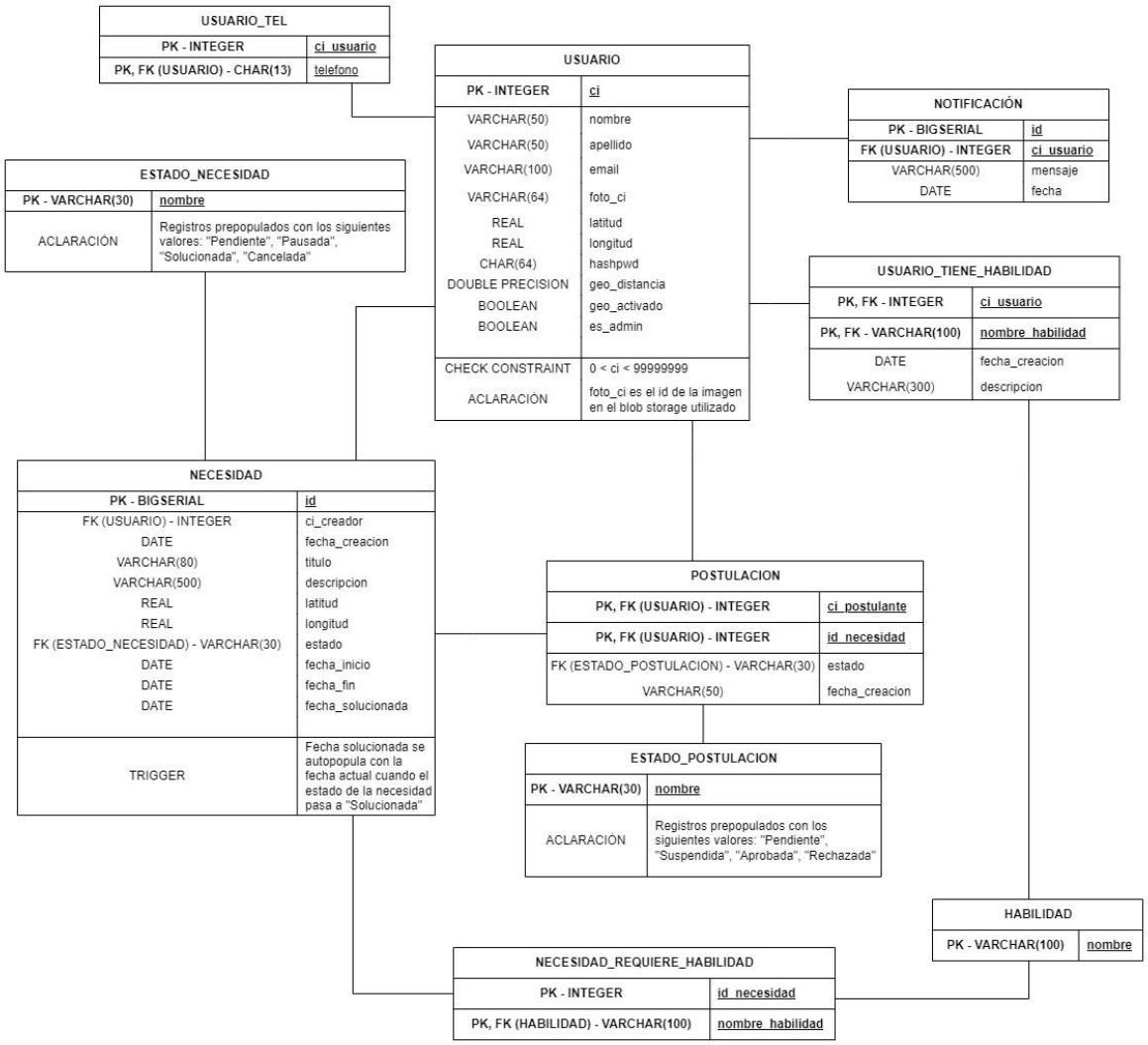
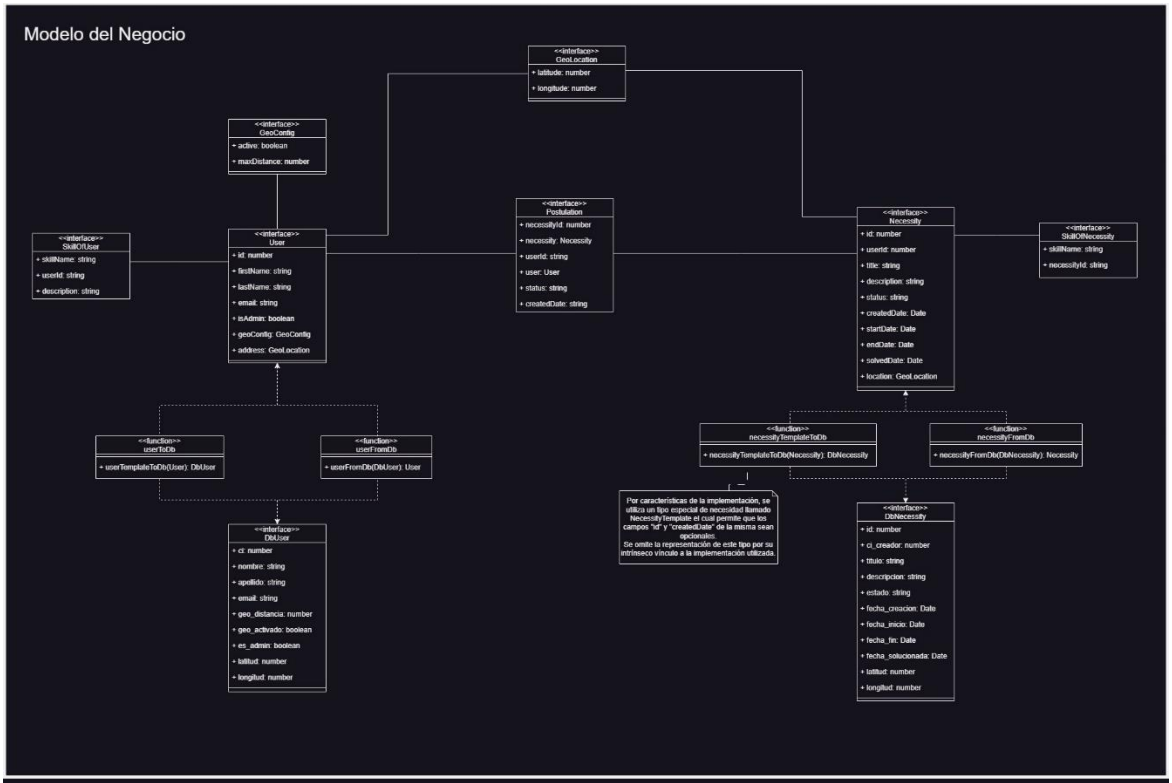
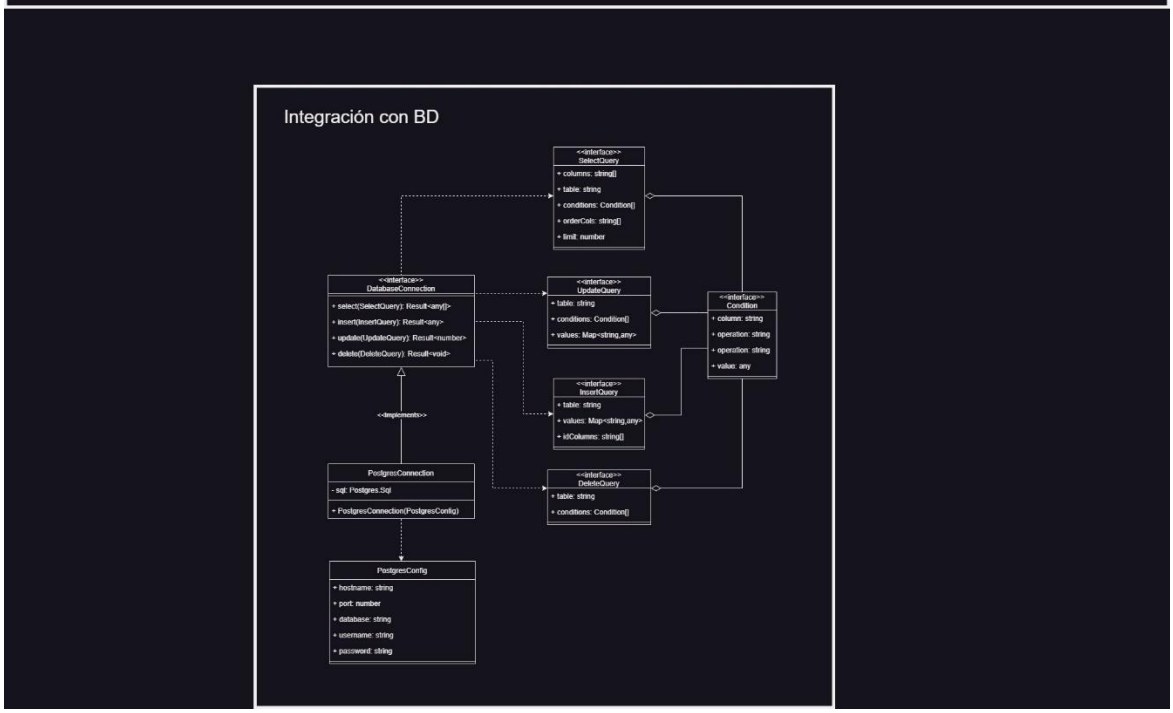


Imagen 4 Representación del modelo lógico, realizado en Draw.io

Diagrama de clases

A continuación, se adjunta un diagrama de clases con las partes relevantes de la aplicación para el modelado del negocio y la integración con la base de datos.





Integración con la base de datos

Siguiendo buenas prácticas del diseño de aplicaciones, realizamos una abstracción de la lógica para la conexión con la base de datos, para no depender de la librería de PostgreSQL y así permitir la adaptabilidad a otros DBMS para nuestra aplicación.

La interfaz central de dicha abstracción es "*DatabaseConnection*", la cual tiene cuatro métodos (*select*, *insert*, *update*, y *delete*) para representar las cuatro operaciones del *CRUD* que se pueden hacer a una tabla: *Create* (*INSERT*), *Read* (*SELECT*), *Update*, y *Delete*.

Cada método en nuestro sistema recibe un único objeto como parámetro, el cual corresponde a una implementación de una interfaz que contiene toda la información necesaria para llevar a cabo la consulta. Esta abstracción nos brinda la capacidad de realizar diversos tipos de consultas que requieren lógica específica en otros sistemas de bases de datos.

Por ejemplo, el método "*select*" recibe un parámetro de tipo "*SelectQuery*", el cual almacena las columnas, la tabla, los filtros, los criterios de orden y el límite. Estos datos pueden utilizarse tal como se presentan en el caso de la librería de Postgres que estamos utilizando. De esta manera, podemos adaptar y gestionar consultas de manera flexible y modular, permitiendo una mayor compatibilidad con diferentes sistemas de bases de datos.

El retorno de cada método es:

- Para la operación de *SELECT*: Un arreglo de objetos que representan las filas seleccionadas.
- Para la operación de *INSERT*: Un objeto que representa la clave primaria de la fila insertada.
- Para la operación de *UPDATE*: El número de filas modificadas, o 0 si ninguna fila se ve afectada.
- Para la operación de *DELETE*: El número de filas eliminadas, o 0 si ninguna fila se ve afectada.

Script para creación de la BASE DE DATOS

```
-- PostgreSQL database dump

SET statement_timeout = 0;

SET lock_timeout = 0;

SET idle_in_transaction_session_timeout = 0;

SET client_encoding = 'UTF8';

SET standard_conforming_strings = on;

SELECT pg_catalog.set_config('search_path', '', false);

SET check_function_bodies = false;

SET xmloption = content;

SET client_min_messages = warning;

SET row_security = off;


-- Name: public; Type: SCHEMA; Schema: -; Owner: postgres

CREATE SCHEMA public;

ALTER SCHEMA public OWNER TO postgres;

-- Name: SCHEMA public; Type: COMMENT; Schema: -; Owner: postgres

COMMENT ON SCHEMA public IS 'standard public schema';


SET default_tablespace = '';

SET default_table_access_method = heap;


-- Name: estado_necesidad; Type: TABLE; Schema: public; Owner: postgres

CREATE TABLE public.estado_necesidad (

    nombre character varying(20) NOT NULL

);

ALTER TABLE public.estado_necesidad OWNER TO postgres;


-- Name: estado_postulacion; Type: TABLE; Schema: public; Owner: postgres

CREATE TABLE public.estado_postulacion (
```

```
    nombre character varying(20) NOT NULL
);
ALTER TABLE public.estado_postulacion OWNER TO postgres;

-- Name: habilidad; Type: TABLE; Schema: public; Owner: postgres
CREATE TABLE public.habilidad (
    nombre character varying(20) NOT NULL
);
ALTER TABLE public.habilidad OWNER TO postgres;

-- Name: necesidad_id_seq; Type: SEQUENCE; Schema: public; Owner: postgres
CREATE SEQUENCE public.necesidad_id_seq
    START WITH 1
    INCREMENT BY 1
    NO MINVALUE
    NO MAXVALUE
    CACHE 1;
ALTER TABLE public.necesidad_id_seq OWNER TO postgres;

-- Name: necesidad; Type: TABLE; Schema: public; Owner: postgres
CREATE TABLE public.necesidad (
    id integer DEFAULT nextval('public.necesidad_id_seq'::regclass) NOT NULL,
    ci_creador integer NOT NULL,
    titulo character varying(80) NOT NULL,
    descripcion character varying(100),
    estado character varying(20) NOT NULL,
    fecha_creacion date DEFAULT now() NOT NULL,
    latitud integer NOT NULL,
    longitud integer NOT NULL,
    fecha_inicio date NOT NULL,
    fecha_fin date,
```

```
    fecha_solucionada date
);
ALTER TABLE public.necesidad OWNER TO postgres;

-- Name: necesidad_req_habilidad; Type: TABLE; Schema: public; Owner: postgres
CREATE TABLE public.necesidad_req_habilidad (
    nombre_habilidad character varying(20) NOT NULL,
    id_necesidad integer NOT NULL
);
ALTER TABLE public.necesidad_req_habilidad OWNER TO postgres;

-- Name: notificacion; Type: TABLE; Schema: public; Owner: postgres
CREATE TABLE public.notificacion (
    id integer NOT NULL,
    ci_usuario integer NOT NULL,
    mensaje character varying(100) NOT NULL,
    fecha date DEFAULT now() NOT NULL
);
ALTER TABLE public.notificacion OWNER TO postgres;

-- Name: postulacion; Type: TABLE; Schema: public; Owner: postgres
CREATE TABLE public.postulacion (
    id_necesidad integer NOT NULL,
    ci_postulante integer NOT NULL,
    estado character varying(20) DEFAULT 'Pendiente'::character varying NOT NULL,
    fecha_creacion date DEFAULT now() );
ALTER TABLE public.postulacion OWNER TO postgres;

-- Name: usuario; Type: TABLE; Schema: public; Owner: postgres
CREATE TABLE public.usuario (
    ci integer NOT NULL,
```

```
    nombre character varying(50) NOT NULL,
    apellido character varying(50) NOT NULL,
    hashpwd character varying(64) NOT NULL,
    email character varying(40) NOT NULL,
    geo_distancia integer DEFAULT 0 NOT NULL,
    geo_activado boolean DEFAULT false NOT NULL,
    es_admin boolean DEFAULT false NOT NULL,
    latitud integer NOT NULL,
    longitud integer NOT NULL );
ALTER TABLE public.usuario OWNER TO postgres;

-- Name: usuario_tiene_habilidad; Type: TABLE; Schema: public; Owner: postgres
CREATE TABLE public.usuario_tiene_habilidad (
    nombre_habilidad character varying(20) NOT NULL,
    ci integer NOT NULL,
    fecha_creacion date DEFAULT now(),
    descripcion character varying(100) );
ALTER TABLE public.usuario_tiene_habilidad OWNER TO postgres;

-- Name: usuario_habilidad; Type: VIEW; Schema: public; Owner: postgres
CREATE VIEW public.usuario_habilidad AS
 SELECT u.ci,
    u.nombre,
    u.apellido,
    u.hashpwd,
    u.email,
    u.geo_distancia,
    u.geo_activado,
    u.es_admin,
    u.latitud,
    u.longitud,
```

```
    uh.nombre_habilidad
   FROM (public.usuario u
     JOIN public.usuario_tiene_habilidad uh ON ((u.ci = uh.ci)));
ALTER TABLE public.usuario_habilidad OWNER TO postgres;

-- Name: usuario_telefono; Type: TABLE; Schema: public; Owner: postgres
CREATE TABLE public.usuario_telefono (
    ci_usuario integer NOT NULL,
    telefono integer NOT NULL);
ALTER TABLE public.usuario_telefono OWNER TO postgres;

-- Data for Name: estado_necesidad; Type: TABLE DATA; Schema: public; Owner: postgres
COPY public.estado_necesidad (nombre) FROM stdin;
Pendiente
Pausada
Solucionada
Cancelada
\.

-- Data for Name: estado_postulacion; Type: TABLE DATA; Schema: public; Owner: postgres
COPY public.estado_postulacion (nombre) FROM stdin;
Pendiente
Suspendida
Aprobada
Rechazada
\.

-- Data for Name: habilidad; Type: TABLE DATA; Schema: public; Owner: postgres
COPY public.habilidad (nombre) FROM stdin;
Carpinteria
Herreria
```

\.

-- Data for Name: necesidad; Type: TABLE DATA; Schema: public; Owner: postgres

COPY public.necesidad (id, ci_creador, titulo, descripcion, estado, fecha_creacion, latitud, longitud, fecha_inicio, fecha_fin, fecha_solucionada) FROM stdin;

2 33333323 titulo1 d1 Pendiente 2023-06-15 0 2 2023-06-14 \N \N

3 33333323 titulo1 d1 Pendiente 2023-06-15 0 2 2023-06-14 \N \N

\.

-- Data for Name: necesidad_req_habilidad; Type: TABLE DATA; Schema: public; Owner: postgres

COPY public.necesidad_req_habilidad (nombre_habilidad, id_necesidad) FROM stdin;

\.

-- Data for Name: notificacion; Type: TABLE DATA; Schema: public; Owner: postgres

COPY public.notificacion (id, ci_usuario, mensaje, fecha) FROM stdin;

\.

-- Data for Name: postulacion; Type: TABLE DATA; Schema: public; Owner: postgres

COPY public.postulacion (id_necesidad, ci_postulante, estado, fecha_creacion) FROM stdin;

\.

-- Data for Name: usuario; Type: TABLE DATA; Schema: public; Owner: postgres

COPY public.usuario (ci, nombre, apellido, hashpwd, email, geo_distancia, geo_activado, es_admin, latitud, longitud) FROM stdin;

33333323 n3 a3 $2a$10$YCxuAA8ba54/4Vn0kbbBdOgqNIt7HWtlm2LR7T.q268pAVqObWKki e3 0 f f 234 234

33333333 n3 a3 $2a$10$YCxuAA8ba54/4Vn0kbbBdOgqNIt7HWtlm2LR7T.q268pAVqObWKki e3 0 f f 234 234

\.

-- Data for Name: usuario_telefono; Type: TABLE DATA; Schema: public; Owner: postgres

COPY public.usuario_telefono (ci_usuario, telefono) FROM stdin;

\.

-- Data for Name: usuario_tiene_habilidad; Type: TABLE DATA; Schema: public; Owner: postgres

COPY public.usuario_tiene_habilidad (nombre_habilidad, ci, fecha_creacion, descripcion) FROM stdin;

Carpinteria	33333323	2023-06-17	\N

Herreria	33333323	2023-06-17	\N

Herreria	33333333	2023-06-17	\N

\.

-- Name: necesidad_id_seq; Type: SEQUENCE SET; Schema: public; Owner: postgres

SELECT pg_catalog.setval('public.necesidad_id_seq', 4, true);

-- Name: estado_necesidad estado_necesidad_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

ALTER TABLE ONLY public.estado_necesidad

    ADD CONSTRAINT estado_necesidad_pkey PRIMARY KEY (nombre);

-- Name: estado_postulacion estado_postulacion_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

ALTER TABLE ONLY public.estado_postulacion

    ADD CONSTRAINT estado_postulacion_pkey PRIMARY KEY (nombre);

-- Name: habilidad habilidad_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

ALTER TABLE ONLY public.habilidad

    ADD CONSTRAINT habilidad_pkey PRIMARY KEY (nombre);

-- Name: necesidad necesidad_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

```
ALTER TABLE ONLY public.necesidad
    ADD CONSTRAINT necesidad_pkey PRIMARY KEY (id);

-- Name: necesidad_req_habilidad necesidad_req_habilidad_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

ALTER TABLE ONLY public.necesidad_req_habilidad
    ADD CONSTRAINT necesidad_req_habilidad_pkey PRIMARY KEY (nombre_habilidad, id_necesidad);

-- Name: notificacion notificacion_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

ALTER TABLE ONLY public.notificacion
    ADD CONSTRAINT notificacion_pkey PRIMARY KEY (id);

-- Name: postulacion postulacion_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

ALTER TABLE ONLY public.postulacion
    ADD CONSTRAINT postulacion_pkey PRIMARY KEY (id_necesidad, ci_postulante);

-- Name: usuario usuario_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

ALTER TABLE ONLY public.usuario
    ADD CONSTRAINT usuario_pkey PRIMARY KEY (ci);

-- Name: usuario_telefono usuario_telefono_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

ALTER TABLE ONLY public.usuario_telefono
    ADD CONSTRAINT usuario_telefono_pkey PRIMARY KEY (ci_usuario, telefono);

-- Name: usuario_tiene_habilidad usuario_tiene_habilidad_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres

ALTER TABLE ONLY public.usuario_tiene_habilidad
    ADD CONSTRAINT usuario_tiene_habilidad_pkey PRIMARY KEY (nombre_habilidad, ci);

-- Name: necesidad necesidad_ci_creador_fkey; Type: FK CONSTRAINT; Schema: public; Owner: postgres
```

```
ALTER TABLE ONLY public.necesidad
    ADD CONSTRAINT necesidad_ci_creador_fkey FOREIGN KEY (ci_creador) REFERENCES public.usuario(ci);


-- Name: necesidad necesidad_estado_fkey; Type: FK CONSTRAINT; Schema: public; Owner: postgres

ALTER TABLE ONLY public.necesidad
    ADD CONSTRAINT necesidad_estado_fkey FOREIGN KEY (estado) REFERENCES public.estado_necesidad(nombre);


-- Name: necesidad_req_habilidad necesidad_req_habilidad_id_necesidad_fkey; Type: FK CONSTRAINT; Schema: public; Owner: postgres

ALTER TABLE ONLY public.necesidad_req_habilidad
    ADD CONSTRAINT necesidad_req_habilidad_id_necesidad_fkey FOREIGN KEY (id_necesidad) REFERENCES public.necesidad(id);


-- Name: necesidad_req_habilidad necesidad_req_habilidad_nombre_habilidad_fkey; Type: FK CONSTRAINT; Schema: public; Owner: postgres

ALTER TABLE ONLY public.necesidad_req_habilidad
    ADD CONSTRAINT necesidad_req_habilidad_nombre_habilidad_fkey FOREIGN KEY (nombre_habilidad) REFERENCES public.habilidad(nombre);


-- Name: notificacion notificacion_ci_usuario_fkey; Type: FK CONSTRAINT; Schema: public; Owner: postgres

ALTER TABLE ONLY public.notificacion
    ADD CONSTRAINT notificacion_ci_usuario_fkey FOREIGN KEY (ci_usuario) REFERENCES public.usuario(ci);


-- Name: postulacion postulacion_ci_postulante_fkey; Type: FK CONSTRAINT; Schema: public; Owner: postgres

ALTER TABLE ONLY public.postulacion
    ADD CONSTRAINT postulacion_ci_postulante_fkey FOREIGN KEY (ci_postulante) REFERENCES public.usuario(ci);
```

-- Name: postulacion postulacion_estado_fkey; Type: FK CONSTRAINT; Schema: public; Owner: postgres

ALTER TABLE ONLY public.postulacion

    ADD CONSTRAINT postulacion_estado_fkey FOREIGN KEY (estado) REFERENCES public.estado_postulacion(nombre);

-- Name: postulacion postulacion_id_necesidad_fkey; Type: FK CONSTRAINT; Schema: public; Owner: postgres

ALTER TABLE ONLY public.postulacion

    ADD CONSTRAINT postulacion_id_necesidad_fkey FOREIGN KEY (id_necesidad) REFERENCES public.necesidad(id);

-- Name: usuario_telefono usuario_telefono_ci_usuario_fkey; Type: FK CONSTRAINT; Schema: public; Owner: postgres

ALTER TABLE ONLY public.usuario_telefono

    ADD CONSTRAINT usuario_telefono_ci_usuario_fkey FOREIGN KEY (ci_usuario) REFERENCES public.usuario(ci);

-- Name: usuario_tiene_habilidad usuario_tiene_habilidad_ci_fkey; Type: FK CONSTRAINT; Schema: public; Owner: postgres

ALTER TABLE ONLY public.usuario_tiene_habilidad

    ADD CONSTRAINT usuario_tiene_habilidad_ci_fkey FOREIGN KEY (ci) REFERENCES public.usuario(ci);

-- Name: usuario_tiene_habilidad usuario_tiene_habilidad_nombre_habilidad_fkey; Type: FK CONSTRAINT; Schema: public; Owner: postgres

ALTER TABLE ONLY public.usuario_tiene_habilidad

    ADD CONSTRAINT usuario_tiene_habilidad_nombre_habilidad_fkey FOREIGN KEY (nombre_habilidad) REFERENCES public.habilidad(nombre);

-- Name: SCHEMA public; Type: ACL; Schema: -; Owner: postgres

REVOKE USAGE ON SCHEMA public FROM PUBLIC;

GRANT ALL ON SCHEMA public TO PUBLIC;

-- PostgreSQL database dump complete

Conclusión

Al plantear los objetivos de este proyecto, desarrollamos una visión de la aplicación propuesta en la consigna, con la cual podríamos ayudar a que distintas personas con distintas habilidades y necesidades pudieran encontrarse para apoyarse mutuamente. En la creación de una aplicación que satisficiera esta visión, nos propusimos también la creación de una base de datos extensible, una API adaptable y una aplicación web accesible, responsiva y extensible.

Siguiendo la metodología y conceptos discutidos en clase y expandiendo sobre estos con nuestra propia investigación, concluimos que llegamos a un modelo de datos relativamente básico, que cubre todas las necesidades esenciales del sistema, pero que está abierto a la extensión para integrar nuevas funcionalidades.

Utilizando los conceptos conocidos por el equipo, complementando con investigación e iteración, se consiguió el desarrollo de una API RESTful que funciona para integrar la aplicación web con el servidor y la base de datos. Además, podría ser utilizada por otros sistemas para interactuar con el nuestro, ya que se utilizan patrones de diseño estándares en la industria para facilitar una API clara y abierta.

Si bien implementamos un sistema de autenticación estándar en la industria utilizando JSON Web Token (JWT), e incluimos la verificación de identidad de usuario a través del envío del documento de identidad en el registro, nos gustaría haber podido implementar también una integración con el sistema de identidad digital de Antel TuID. Aun así, consideramos que este objetivo fue parcialmente satisfecho.

Finalmente, combinando todos los objetivos previamente mencionados, consideramos que conseguimos un producto válido, altamente extensible y capaz de satisfacer los requisitos planteados inicialmente en la consigna de este proyecto.

Los próximos pasos para nuestra aplicación, serían la inclusión de autenticación con sistemas de terceros por medio del protocolo OAuth 2.0 junto a la verificación de identidad utilizando servicios de firma digital como TuID. Además, podríamos fortificar los aspectos de red social del producto, agregando una red de amistades, la posibilidad de comentar en necesidades de otros usuarios e incluyendo un chat en tiempo real dentro de la aplicación.

Glosario:

API RESTful: es una API web que sigue las pautas REST según sea necesario para solucionar un problema.

Backend: es la parte lógica y de funcionamiento detrás de una aplicación o sitio web.

Brainstorming: es una técnica grupo o individual de creación de contenido creativo, donde todas las soluciones a un problema cuentan y se registran para ser evaluadas posteriormente.

Código abierto: es aquel código fuente que esta disponible y puede ser utilizado por cualquier persona.

Colaboradores: son las personas que poseen una habilidad y participan activamente en ayudar a otros.

CRUD: son el conjunto de operaciones básicas para el manejo de datos (crear, leer, actualizar y borrar, respectivamente)

Distancia de geolocalización: distancia hasta la cual un usuario es visible desde la funcionalidad de búsqueda de usuarios.

DBMS: es un software que permite administrar y gestionar bases de datos.

Framework: es un conjunto de herramientas, bibliotecas y estructuras de código que permiten el desarrollo de aplicaciones de Software.

GoldPlating: es un término que refiere a agregar funcionalidades innecesarias en un proyecto.

Habilidades: es el nombre que recibe la capacidad o competencia que un usuario del sistema posee, que le permite realizar tareas, cierta actividad o resolver problemas.

JSON Web Token (JWT): es un estándar abierto de la industria de software definido en el RFC-7519 para verificar la identidad e integridad de un mensaje enviado a través de la web.

Modelo de entidad-relación: es una herramienta para el diseño de Bases de Datos, que permite representar y describir estructuras y relaciones.

Necesidades: es el nombre la carencia de un usuario del sistema de una habilidad.

Necesitados: son las personas que indicaron poseer una necesidad.

Networking: es el proceso de establecer y gestionar conexiones entre dispositivos y sistemas informáticos.

PGDG: es el nombre de la comunidad de desarrolladores y colaboradores de PostgreSQL.

REST: es un conjunto de pautas de arquitectura de software para el desarrollo de APIs web extensibles, legibles, uniformes y abstraídas desde el punto de vista del cliente.

Volere: es el nombre de la empresa que utilizamos como referencia para el modelado de los requisitos.

Referencias bibliográficas

AirTasker. (s.f.). How it works. Recuperado de https://www.airtasker.com/au/how-it-works/

Thumbtack. (s.f.). Home - Thumbtack. Recuperado de https://www.thumbtack.com/

Meetup. (s.f.). Acerca de nosotros. Recuperado de https://www.meetup.com/es/about/

Nextdoor. (s.f.). Aprovecha todo lo que te ofrece tu barrio con Nextdoor. Recuperado de https://nextdoor.com/

TaskRabbit. (s.f.). Revolutionizing Everyday Work. Recuperado de https://www.taskrabbit.com/

LinkedIn. (s.f.). Acerca de LinkedIn. Recuperado de https://about.linkedin.com/es-es?trk=seo-authwall-base_footer-about&lr=1

Robertson, S. y Robertson, J. (07/2014). *Business Events.* Mastering the requirement process. Getting requirements right. Third Edition. (pp. 73-74) Ed. Pearson Education, Inc.

Cockburn, A. (2000). Writing Effective Use Cases. Addison-Wesley Professional.

Ripollet, A. (31/07/2021). *10 maneras de verificar la identidad de una persona.* Barcelona hoy. Recuperado de https://www.barcelonahoy.es/verificar-la-identidad-de-una-persona-de-forma-facil-en-10-pasos#:~:text=Para%20verificar%20la%20identidad%20de,siempre%20debe%20llevar%20consigo%20legalmente.

Identidad Digital. (s.f.). Elegí la forma de identificarte digitalmente. Recuperado de https://mi.iduruguay.gub.uy/

Google. (s.f.). Google Maps Platform. API de Mapas de Google. Recuperado de https://developers.google.com/maps/documentation?hl=es-419

Silberschatz, A., Korth, H. y Sudershan, S. (2002). Primera Parte, Capitulo 2 Modelo Entidad-Relación. Fundamentos de Bases de Datos, Cuarta edición. (pp. 19-52). Editorial McGraw Hill Inc.

Simsion, G., Witt, G. (2005). *Logical Database Design*. Data Modeling Essentials. (pp. 321-356). Morgan Kaufmann Publishers

Angular. (28/02/2022). ¿Qué es Angular?. Recuperado de https://angular.io/guide/what-is-angular

Node.js. (s.f.). Acerca de Node.js. Recuperado de https://nodejs.org/es/about

Red Hat. (2020). ¿Qué es una API de REST?. Recuperado de https://www.redhat.com/es/topics/api/what-is-a-rest-api

Internet Engineering Task Force (IETF). (2015). JSON Web Token (JWT). Recuperado de https://datatracker.ietf.org/doc/html/rfc7519

Anexo

Repositorios de Github

Aplicación Cliente o Front-End:
https://github.com/santiagodeolivera/UCU-1-2023-BD-Obligatorio

Aplicación Servidor o Back-End:
https://github.com/WalterTano/UCU-1-2023-BD-Obligatorio-Backend