Informe final

Calderón Prieto Brandon - 2125974

Hernández Agudelo Carlos - 2125652

Muñoz Gutiérrez Mauricio - 2123687

Jefferson Amado Peña Torres

Cali - Valle del Cauca, 2022

Bases de Datos

Índice general

0.1. Planteamiento

La aplicación **Mande**, como intermediario entre personas del común y trabajadores, para la contratación de servicios, propone implementar las siguientes funcionalidades:

- Contratación de trabajadores, de una forma cómoda, ofreciendo **filtros** para búsquedas personalizadas.
- Ofrecimiento de labores por parte de los trabajadores, con la posibilidad de:
  - Ofrecer varias labores a la vez (con una descripción opcional).
  - Cobrar por unidad de tiempo o trabajo.

Dichas funcionalidades fueron implementadas en nuestra solución junto con todas las restricciones declaradas en el proyecto:

0.1.1. Trabajador

- Registro
  - Foto de perfil, imagen de documento de identidad y dirección de residencia obligatoria.
  - La dirección de residencia se convertirá en una coordenada de GPS.
- Labores
  - Selección de una lista predefinida de labores.
  - Cada labor debe poner su precio por hora o unidad de labor.

0.1.2. Cliente

- Registro
  - Se pide la información personal.
  - Dirección de residencia, que se convertirá en una coordenada de GPS para la aplicación.

- Recibo de servicio público que se usará internamente para validar el lugar de residencia.
- Email y número de celular usados para **iniciar sesión**.
- Medio de pago (tarjeta, crédito o débito).

- El medio de pago debe guardarse de forma encriptada.

0.2. Modelo Entidad - Relación

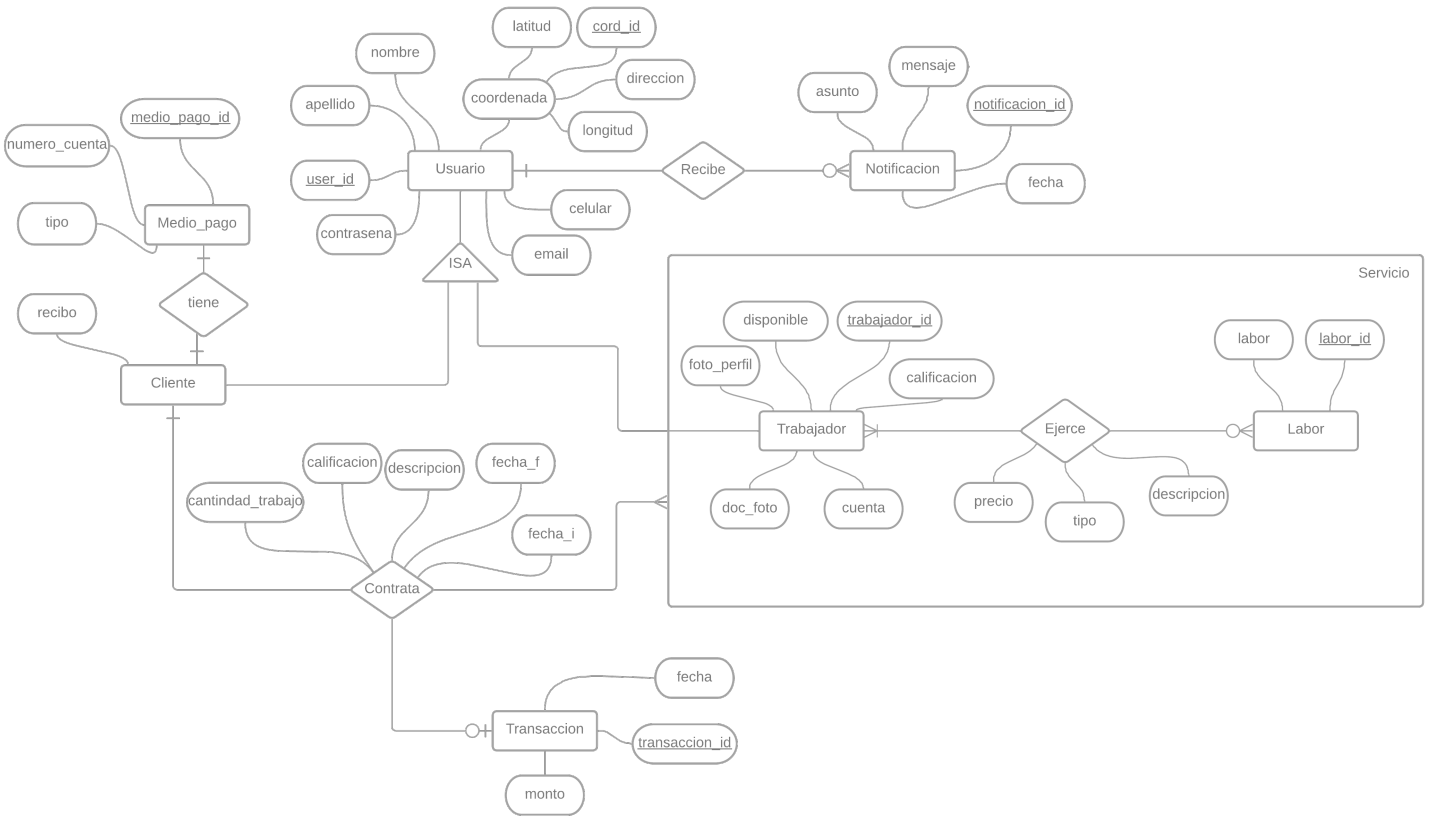

Figura 1: Modelo Entidad Relación

0.3. Modelo relacional

A partir del modelo Entidad-Relación presentado anteriormente, tras un proceso de transformación se ha logrado definir el esquema relacional para la base de datos.

- Coordenada(**coor_id**: INT, latitud: FLOAT, longitud: FLOAT, direccion: VARCHAR(255))
- Medio_pago(**medio_pago_id**: INT, **numero_cuenta**: INT, tipo: medio_pago_tipo)
- Usuario(**user_id**: INT, nombre: VARCHAR(255), apellido: VARCHAR(255), email: VARCHAR(255), contrasena: VARCHAR(255), celular: VARCHAR(255),coor_id* (Coordenada): INT)
- Cliente(**cliente_id**: INT, recibo: VARCHAR(255), numero_cuenta*: VARCHAR(255) (Medio_pago), user_id* (Usuario): INT)
- Trabajador(**trabajador_id**: INT, foto_perfil: VARCHAR(255), disponible: BOOLEAN, calificacion: FLOAT, doc_foto: VARCHAR(255), cuenta: VARCHAR(255), user_id*: INT (Usuario))
- Labor(**labor_id**: INT, labor: labor_tipo)
- Notificacion(**notificacion_id**: INT, fecha: DATE, mensaje: VARCHAR(255), asunto: asunto_tipo, user_id*: INT (Usuario))
- Transaccion(**transaccion_id**: INT, fecha: DATE, monto: FLOAT)
- Ejerce(**ejerce_id**: INT, trabajador_id*: INT (Trabajador), labor_id*: INT (Labor), tipo_trabajo: tipo_trabajo, precio: FLOAT, descripcion: VARCHAR(255))
- Contrato(**contrato_id**: INT, ejerce_id*: INT (Ejerce), cliente_id* : INT (Cliente), cantidad_trabajo: INT, calificacion: INT, descripcion: VARCHAR(255), fecha_i: DATE, fecha_f: DATE, transaccion_id* : INT (Transaccion))

0.4. Desarrollo

Esta aplicación web, permite a los usuarios contratar servicios de trabajadores a domicilio en tareas cotidianas como Carpintería, Electricidad, Plomería, Pintura y demás. Los trabajadores y

clientes pueden registrarse con sus respectivos datos, en donde la contraseña y el medio de pago de cliente se guardan de manera encriptada para asegurar la seguridad de la applicacion. Los trabajadores pueden seleccionar las labores que están dispuestos a realizar y especificar su precio por hora o unidad de trabajo, así como una descripción si lo desea. Cuando un usuario necesita un servicio, la aplicación busca los trabajadores disponibles por labor y los ordena por calificación, distancia y precio, y si este lo desea lo contratará, haciendo que le llegue una notificación al trabajador. Luego, el trabajador puede indicarle al cliente cuando finalizo el trabajo para enviarle una notificación al usuario y así este pueda calificar y pagarle al trabajador.

- Backend: Express - Node - Bcrypt
- Frontend: React - Tailwind - Chakra UI - Formik - API de opencagedata
- Database: Postgresql

0.5. Implementación en la base de datos

- **Esquema.**

  El esquema fue implementado de acuerdo a lo mencionado en la sección del modelo relacional.

- **Funciones.**

  - **verificar_email**: verificar que un email no esté en uso.
  - **verificar_login_trabajador**: verificar que el usuario (con `user_id`) tiene un trabajador asociado.
  - **verificar_login_cliente**: verificar que el usuario (con `user_id`) tiene un cliente asociado.
  - **verificar_cuenta_trabajador**: verificar que una cuenta (`numero_cuenta`) de un trabajador este en uso.
  - **verificar_cuenta_cliente**: verificar que una cuenta (`numero_cuenta`) de un cliente este en uso.

  - **verificar_celular**: verificar que un celular esté en uso.
  - **buscar_trabajadores**: buscar trabajadores según su `labor_id`, `latitudn`, `longitud_in` y `criterio`.

- **Procedimientos.**
  - **crear_trabajador**: usado para registrar un trabajador en la aplicación.
  - **crear_cliente**: usado para registrar un cliente en la aplicación.
  - **calificarServicio**: indicado un contrato, se le asigna una calificación del servicio.
  - **finalizarContrato**: usado para poner fecha de finalización a los contratos.
  - **realizarPago**: usado para indicar que se ha realizado una transacción.

- **Vistas.**
  - **infoContratoT**: muestra toda la información de los contratos para los trabajadores.
  - **infoTransaccionT**: muestra todas las transacciones relacionadas con los trabajadores.
  - **notificacionesT**: muestra todas las notificaciones que solo reciben los trabajadores.
  - **infoContratoC**: muestra toda la información necesaria de los contratos a los clientes.
  - **infoTransaccionC**: muestra todas las transacciones relacionadas con los clientes.
  - **notificacionesC**: muestra todas las notificaciones que solo reciben los clientes.
  - **laboresDisponibles**: muestra todos los labores en las cuales los trabajadores están disponibles.
  - **infoServicio**: muestra toda la información de todos los servicios.

- **Triggers.**
  - **actualiza_disponibilidad**: cuando se realiza un contrato, la disponibilidad del trabajador cambia a falso.
  - **update_trabajador_disponibilidad**: cuando se finaliza un contrato, la disponibilidad del trabajador cambia a verdadero.

- **actualizar_promedio_calificacion**: cuando se califica un contrato, calcular la calificación promedio del trabajador teniendo en cuenta la ultima calificación.
- **notificar_usuario**: cuando se realiza un contrato, se indica por medio de una notificacion al trabajador y cuando finaliza el contrato se le avisa al cliente.

0.6. Resultados

Para el desarrollo de la aplicación se utilizó un stack PERN (React en el frontend, Node y Express en el backend y PostgreSQL para almacenamiento) para desarrollar la aplicación Mande. La combinación de estas tecnologías permitió la creación de una aplicación que cumple con todos los requerimientos específicos solicitados que debe tener la aplicación, incluyendo una interfaz que funciona tanto en celular como en computador y una experiencia de usuario atractiva.

Despliegue

Hacemos uso de Docker, en específico la herramienta `docker-compose` para desplegar y conectar varios contenedores con mayor comodidad. Se definieron 3 contenedores, cada uno para almacenar una parte de la estructura de la aplicación (fronted, backend y la base de datos).

0.7. Experiencias

Nuestro grupo tuvo la experiencia de desarrollar **Mande** que brinda a los usuarios la posibilidad de conseguir trabajadores expertos y confiables para realizar tareas cotidianas en sus hogares, desde la comodidad de su celular. La aplicación utiliza tecnologías como React, Express, Node.js, y PostgreSQL, usando el stack **PERN**.

Durante el desarrollo del proyecto, nos encontramos con varios desafíos al implementar tanto la parte de front-end como la de back-end de la aplicación. Dado que fue nuestra primera vez en el desarrollo de una aplicación de este tipo, tuvimos que aprender a medida que avanzábamos. Sin embargo, esto resultó en una experiencia de aprendizaje inolvidable.

Aprendimos a utilizar herramientas como React, comprendiendo el uso de estados, formularios, componentes y cómo conectar todo esto con un back-end. Descubrimos la importancia de conectar los servicios de front-end y back-end, y aprendimos a hacerlo a través del uso de Docker

Compose. Aunque en un principio hubo dificultades, al final logramos superarlas y hacer que la aplicación funcionara correctamente.

Gracias al conocimiento adquirido en el curso de *Base de Datos* pudimos hacer uso de PostgreSQL en nuestra aplicación, garantizando un buen rendimiento en la gestión y acceso a los datos de los usuarios. La base de datos resultó ser un pilar fundamental en el desarrollo de la aplicación y su importancia radica en la capacidad de almacenar, acceder y utilizar de manera eficiente los datos necesarios para el correcto funcionamiento de la aplicación.

Además, descubrimos la importancia de la planificación y la organización en el desarrollo de un proyecto de este tipo, y aprendimos a trabajar de manera colaborativa y efectiva como equipo con herramientas que ya conocíamos previamente como GitHub. Cada dificultad superada nos brindó una nueva habilidad y una mayor confianza en nuestras capacidades como desarrolladores. En definitiva, este proyecto fue una experiencia enriquecedora y valiosa que contribuirá a nuestro crecimiento profesional a largo plazo.

0.8. Links necesarios

- Perfil de Carlos Hernández: https://github.com/Carlosher007
- Perfil de Brandon Calderon: https://github.com/Br4z
- Perfil de Mauricio Muñoz: https://github.com/MauMG03
- GitHub donde esta alojado el proyecto: https://github.com/Carlosher007/Proyecto-Base-de-Datos
- Vídeo de Youtube: https://youtu.be/H6dbJvmF2pM