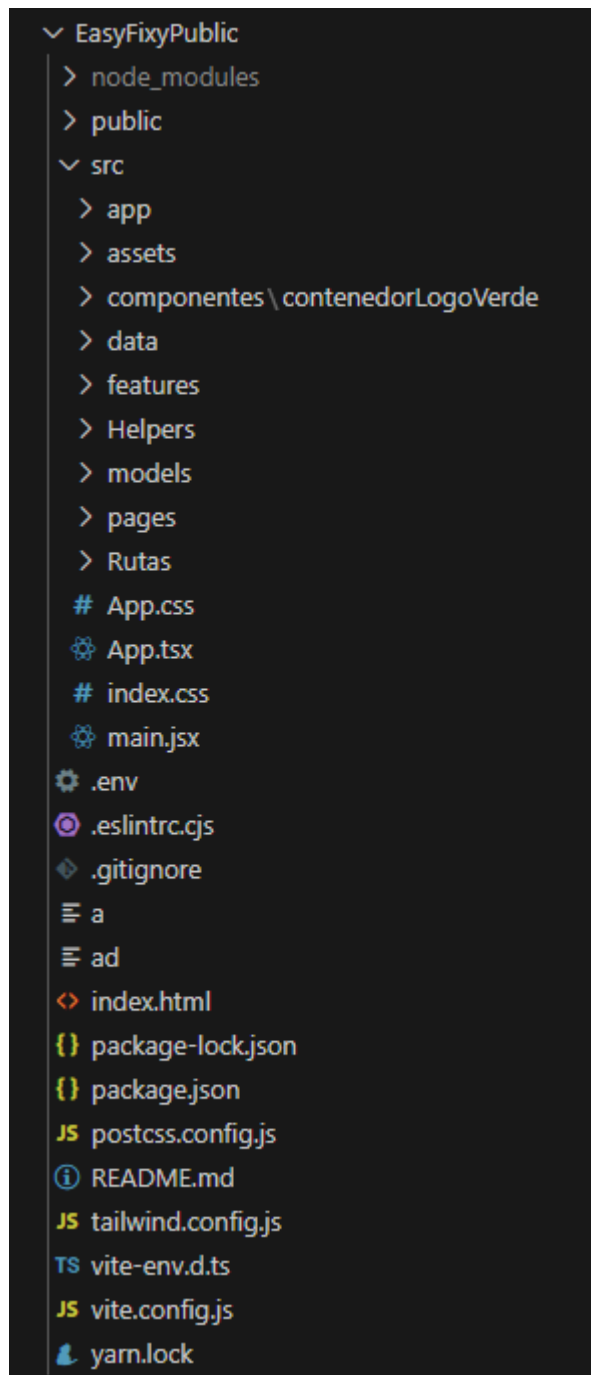


# Manual técnico EasyFixy app

<b>Frontend</b>	<b>2</b>
Distribución	2
Carpeta src	2
Otros Archivos y Carpetas	4
Librerías	5
Dependencias de Producción	5
Dependencias de Desarrollo	7
Repositorio	8
Gitignore	8
Despliegue del Front	8
<b>Backend</b>	<b>10</b>
Distribución	10
Carpeta src	10
Otros Archivos	11
Librerías	12
Repositorio	13
Gitignore	13
Despliegue del Back y Base de datos	13
<b>Base de datos</b>	<b>15</b>
Diagrama entidad Relación	16

# Frontend

## Distribución



## Carpeta src

- **app:**  
Contiene el código relacionado con la configuración y gestión global de la aplicación, como el estado global, y otras configuraciones generales.

- **assets:**

Almacena recursos estáticos como imágenes, fuentes, y archivos que no cambian en el tiempo de ejecución. Estos recursos se pueden importar y usar en los componentes de React.
- **componentes:**

Almacena los componentes React reutilizables que se utilizan en diferentes partes de la aplicación. Por ejemplo, el componente `contenedorLogoHorizontal` es un componente visual específico que se usa en varias páginas.
- **data:**

Esta carpeta almacena datos estáticos y configuraciones que se utilizan en la aplicación, como archivos JSON, constantes o configuraciones predeterminadas, en la aplicación almacena los datos de los países para los select de entradas de teléfono.
- **features:**

Contiene funcionalidades o módulos específicos de la aplicación. Cada carpeta dentro de `features` podría corresponder a una funcionalidad principal, como autenticación, gestión de usuarios, etc. para la aplicación se usó para guardar las funciones de React Redux.
- **Helpers:**

Almacena funciones utilitarias y helpers que son reutilizados en diferentes partes de la aplicación para realizar tareas comunes, como el formateo de fechas, manejo de datos y demás.
- **models:**

Esta carpeta contiene definiciones de modelos de datos o clases que representan las estructuras de datos TypeScript utilizadas en la aplicación.
- **pages:**

Contiene los componentes React que representan las diferentes páginas o vistas de la aplicación. Cada archivo dentro de `pages` generalmente corresponde a una ruta o vista específica en la aplicación.
- **Rutas:**

Almacena la configuración de las rutas de la aplicación. Aquí se define cómo se mapean las URL a los componentes de las páginas en `pages` cuando el usuario está logueado.
- **App.css:**

Archivo CSS que contiene estilos globales o específicos para el componente `App`.
- **App.tsx:**

El componente principal de la aplicación donde se pueden configurar el enrutamiento y la estructura general de la interfaz.

- `index.css`:  
Archivo CSS global donde se pueden definir estilos base que se aplican en toda la aplicación.
- `main.jsx`:  
El punto de entrada de la aplicación React. Aquí es donde el componente principal App es montado en el DOM.

## Otros Archivos y Carpetas

- `.env`:  
Archivo de configuración que almacena variables de entorno que se utilizan en la aplicación, como claves API o configuraciones sensibles.
- `.eslintrc.cjs`:  
Archivo de configuración de ESLint, una herramienta para analizar el código y asegurar que sigue ciertas reglas de estilo y buenas prácticas.
- `.gitignore`:  
Archivo que especifica qué archivos o carpetas deben ser ignorados por Git, como `node_modules` o archivos de configuración locales.
- `index.html`:  
El archivo HTML principal donde se monta la aplicación React. Vite lo utiliza como plantilla durante el proceso de construcción.
- `package.json`:  
Archivo de configuración del proyecto que lista las dependencias, scripts, y otra información sobre la aplicación.
- `postcss.config.js`:  
Configuración de PostCSS, una herramienta que transforma CSS con plugins, como autoprefixer.
- `README.md`:  
Documento que contiene información sobre el proyecto, como cómo instalarlo y usarlo.
- `tailwind.config.js`:  
Archivo de configuración de Tailwind CSS, donde se personalizan temas, colores, y otros aspectos del framework CSS.
- `vite-env.d.ts`:  
Archivo de tipos de TypeScript que proporciona definiciones para las variables de entorno de Vite, ayudando a mejorar la experiencia de desarrollo con TypeScript.

- vite.config.js:  
Archivo de configuración de Vite, donde se pueden personalizar aspectos como alias de rutas, configuraciones de plugins, y más.
- yarn.lock:  
Archivo de bloqueo de dependencias utilizado por Yarn para asegurar que las mismas versiones de las dependencias se instalen en diferentes entornos.

## Librerías

### Dependencias de Producción

@emotion/react - Versión: ^11.11.4

Proporciona utilidades para escribir estilos CSS en JavaScript con soporte para estilos dinámicos y temáticos la aplicación.

@emotion/styled - Versión: ^11.11.5

Permite crear componentes React con estilos embebidos utilizando una sintaxis similar a styled-components, permitiendo estilos dinámicos basados en props.

@fortawesome/free-solid-svg-icons - Versión: ^6.5.2

Proporciona un conjunto de íconos SVG de Font Awesome para usar en la interfaz de usuario.

@fortawesome/react-fontawesome - Versión: ^0.2.0

Integración de Font Awesome con React para facilitar la inclusión de íconos en componentes React.

@mui/icons-material - Versión: ^5.15.18

Proporciona un conjunto de íconos Material Design para su uso en la aplicación con Material-UI.

@mui/material - Versión: ^5.15.18

Biblioteca de componentes de Material-UI para construir interfaces de usuario con diseño Material en la aplicación.

@reduxjs/toolkit - Versión: ^2.2.2

Herramientas y utilidades para trabajar con Redux de manera más sencilla y eficiente, facilitando la gestión del estado global de la aplicación.

axios - Versión: ^1.6.7

Realizar solicitudes HTTP desde el front-end, como obtener datos de APIs o enviar datos a un servidor.

bootstrap - Versión: ^5.3.3

Biblioteca de CSS para crear interfaces de usuario responsivas y estéticamente atractivas con componentes predefinidos.

jsonwebtoken - Versión: ^9.0.2

Validar y decodificar tokens JWT en el cliente, especialmente para manejar autenticación.

react - Versión: ^18.2.0

La biblioteca principal para construir interfaces de usuario en componentes.

react-dom - Versión: ^18.2.0

Paquete que trabaja junto con React para renderizar componentes en el DOM.

react-flags-select - Versión: ^2.2.3

Componente para seleccionar países utilizando banderas en la interfaz de usuario.

react-loader-spinner - Versión: ^6.1.6

Mostrar animaciones de carga en la interfaz de usuario mientras se realizan solicitudes o procesos en segundo plano.

react-redux - Versión: ^9.1.0

Proporcionar la integración entre React y Redux para conectar componentes a la tienda de estado global.

react-router-dom - Versión: ^6.23.0

Gestionar la navegación y el enrutamiento en una aplicación React, permitiendo la creación de rutas dinámicas y la navegación entre diferentes vistas.

react-select - Versión: ^5.8.0

Crear menús desplegables con soporte para búsqueda y selección múltiple en React.

react-spinners - Versión: ^0.13.8

Proporcionar una variedad de componentes de animación de carga (spinners) para mostrar mientras los datos o componentes se están cargando.

react-toastify - Versión: ^10.0.5

Mostrar notificaciones de forma fácil y personalizable en la aplicación.

reactstrap - Versión: ^9.2.2

Biblioteca de componentes React que utilizan Bootstrap para construir interfaces de usuario.

redux - Versión: ^5.0.1

Gestión del estado global de la aplicación, permitiendo un flujo de datos predecible a través de un store centralizado.

socket.io-client - Versión: ^4.7.5

Implementar la comunicación en tiempo real desde el cliente, permitiendo que los usuarios interactúen en vivo en el chat.

## Dependencias de Desarrollo

@types/react - Versión: ^18.2.56

Proporcionar definiciones de tipo TypeScript para React, mejorando la experiencia de desarrollo con tipado estático.

@types/react-dom - Versión: ^18.2.19

Proporcionar definiciones de tipo TypeScript para react-dom.

@vitejs/plugin-react - Versión: ^4.2.1

Integrar React con Vite, un empaquetador rápido y moderno para proyectos JavaScript.

autoprefixer - Versión: ^10.4.18

Añadir automáticamente prefijos específicos del navegador a las reglas CSS para mejorar la compatibilidad entre navegadores.

eslint - Versión: ^8.56.0

Herramienta de análisis de código que ayuda a identificar y corregir problemas en el código JavaScript.

eslint-plugin-react - Versión: ^7.33.2

Proporciona reglas específicas para React en ESLint, ayudando a seguir mejores prácticas y evitar errores comunes.

eslint-plugin-react-hooks - Versión: ^4.6.0

Añade reglas específicas para asegurar el uso correcto de Hooks en React.

eslint-plugin-react-refresh - Versión: ^0.4.5

Facilita la recarga automática de componentes React durante el desarrollo para mejorar la experiencia de desarrollo.

postcss - Versión: ^8.4.35

Herramienta para transformar CSS con plugins como autoprefixer.

react-scripts - Versión: ^5.0.1

Proporciona scripts y configuración para crear la aplicación, principalmente con Create React App.

tailwindcss - Versión: ^3.4.1

Un framework de CSS utilitario para construir interfaces de usuario personalizadas de manera rápida.

vite - Versión: ^5.2.7

Un empaquetador rápido y ligero para proyectos modernos de frontend, optimizado para un rendimiento rápido en el desarrollo.

## Repositorio

El proyecto está alojado en el repositorio **EasyFixyPublic** (<https://github.com/EasyFixy/EasyFixyPublic.git>) de la organización **EasyFixy**. Durante el desarrollo, se creó una rama dedicada para cada tarea, utilizando como nombre el título correspondiente en Jira.

## Gitignore

```
1  # Logs
2  logs
3  *.log
4  npm-debug.log*
5  yarn-debug.log*
6  yarn-error.log*
7  pnpm-debug.log*
8  lerna-debug.log*
9
10 node_modules
11 dist
12 dist-ssr
13 *.local
14
15 # Editor directories and files
16 .vscode/*
17 !.vscode/extensions.json
18 .idea
19 .DS_Store
20 *.suo
21 *.ntvs*
22 *.njsproj
23 *.sln
24 *.sw?
25
```

## Despliegue del Front

Para el despliegue del Front hicimos uso de una Web as Services llamada Vercel, la cual facilita bastante el despliegue ya que se encuentra integrada con gitHub, por ende para el despliegue simplemente se conectó la misma cuenta de gitHub con Vercel y esta automáticamente trae todos los repositorios asociados a esta, en el despliegue únicamente adicional a seleccionar qué tecnología usamos (en nuestro caso React) añadimos variables de entorno, en nuestro caso únicamente la variable que contiene el enlace de la conexión al back:



ludian74gmailcom's projects

Hobby

easyfixy

Feedback

Changelog

Help

Docs

Project

Deployments

Analytics

Speed Insights

Logs

Firewall

Storage

Settings

easyfixy

Repository

Usage

Domains

Visit


Production Deployment

Build Logs

Runtime Logs

Instant Rollback

The deployment that is available to your visitors.



Deployment

easyfixy-ianiuky8d-ludian74gmailcoms-projects.vercel.app

Domains

easyfixy.vercel.app easyfixy-ludian74gmailcoms-projects.vercel.app +1

Status

Created

Ready 11d ago by lusanchezmo

Source

main

5d552af Merge pull request #74 from EasyFixy/dev

https://easyfixy.vercel.app

To update your Production Deployment, push to the "main" branch

Learn More

Variable de entorno:

Search...

All Environments

Last Updated

<>

VITE\_BASE\_URL

Development, Preview, Production

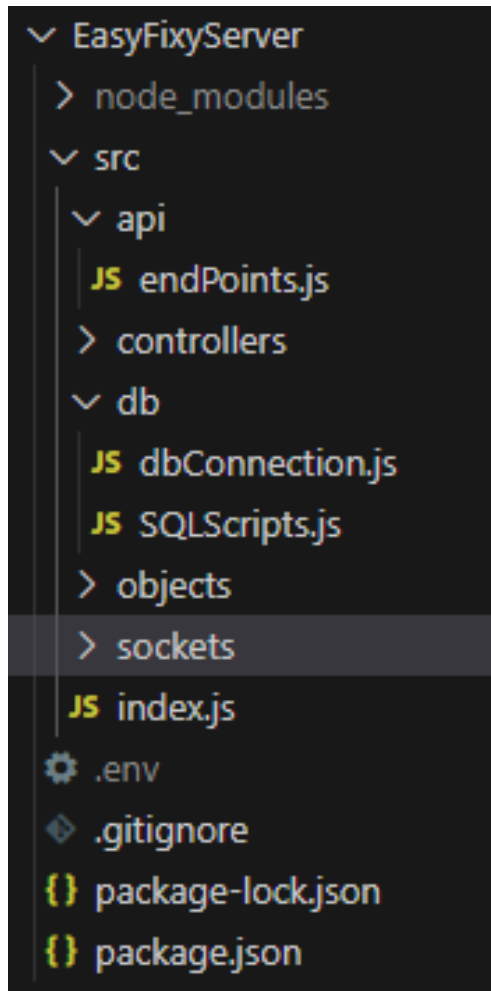
https://easyfixyserver-produc...

Updated 11d ago

...

# Backend

## Distribución



## Carpeta src

- api:
  - Almacena los puntos finales de la API que definen las rutas y métodos HTTP que pueden ser llamados desde el cliente. Estos puntos finales gestionan las solicitudes entrantes y redirigen las peticiones a los controladores correspondientes.
- endpoints.js:
  - Contiene la configuración de los diferentes puntos finales de la API. Aquí se definen las rutas y los métodos HTTP (GET, POST, PUT, DELETE) que se exponen para que el cliente interactúe con el servidor.
- controllers:
  - Almacena los controladores que contienen la lógica de negocio de la aplicación. Los controladores se encargan de procesar las solicitudes recibidas a

través de los puntos finales de la API y de interactuar con la base de datos o realizar otras operaciones necesarias.

- db:
  - Contiene todos los archivos relacionados con la conexión y las operaciones de la base de datos.
  - dbConnection.js:
    - Configura y establece la conexión con la base de datos. Este archivo se asegura de que el servidor esté correctamente conectado a la base de datos para realizar operaciones CRUD.
  - SQLScripts.js:
    - Almacena consultas SQL y scripts predefinidos que se utilizan en diferentes operaciones de base de datos, como la creación de tablas, la inserción de datos, o la ejecución de consultas específicas.
- objects:
  - Podría almacenar clases o estructuras de datos que representan objetos de dominio específicos de la aplicación. Estos objetos se pueden utilizar para mapear datos de la base de datos a estructuras en el servidor y viceversa.
- sockets:
  - Contiene la configuración y lógica para la comunicación en tiempo real a través de WebSockets. Aquí se define cómo los clientes y el servidor intercambian mensajes en tiempo real, como en el caso de un chat.
- index.js:
  - El punto de entrada principal de la aplicación del servidor. Aquí se inicializa el servidor, se configuran las rutas principales, se establece la conexión con la base de datos, y se ponen en marcha otros servicios como los sockets.

## Otros Archivos

- .env:
  - Archivo de configuración que almacena variables de entorno sensibles, como claves API, configuraciones de la base de datos, y otras configuraciones que no deberían estar expuestas en el código fuente.
- .gitignore:
  - Archivo que especifica qué archivos o carpetas deben ser ignorados por Git. Esto incluye archivos sensibles o específicos del entorno que no deberían ser versionados, como node\_modules y archivos de configuración local.
- package.json:

Archivo de configuración que lista las dependencias del proyecto, scripts de npm, y otra información relevante para el proyecto. Es esencial para la gestión de las dependencias y la automatización de tareas comunes.

- package-lock.json:

Archivo que asegura que se instalen las mismas versiones de las dependencias en todos los entornos, bloqueando las versiones específicas que fueron instaladas en el momento de la última ejecución de npm install.

## Librerías

**cors** - Versión: 2.8.5

Permitir que la API acepte solicitudes desde otros dominios, habilitando la comunicación entre el servidor y aplicaciones front-end alojadas en diferentes orígenes.

**dotenv** - Versión: 16.4.5

Cargar configuraciones sensibles, como claves API o credenciales de la base de datos, desde un archivo .env para mantenerlas separadas del código fuente.

**express** - Versión: 4.18.3

Servir como el marco de trabajo principal para construir la API y manejar las rutas y middleware en tu aplicación Node.js.

**jsonwebtoken** - Versión: 9.0.2

Implementar la autenticación basada en tokens JWT para el sistema de login, permitiendo que los usuarios inicien sesión y se mantengan autenticados en la aplicación.

**mercadopago** - Versión: 2.0.8

Integrar el sistema de pagos en línea, permitiendo a los usuarios realizar transacciones a través de la plataforma de Mercado Pago.

**mysql** - Versión: 2.18.1

Conectar y ejecutar consultas en la base de datos MySQL, manejando operaciones relacionadas con la persistencia de datos.

**mysql2** - Versión: 3.9.7

Similar a mysql, pero utilizado para obtener un mejor rendimiento y aprovechar las características adicionales como soporte para Promises y async/await en la interacción con la base de datos MySQL.

**nodemailer** - Versión: 6.9.11

Enviar correos electrónicos automatizados, como confirmaciones de registro o notificaciones a los usuarios.

**socket.io** - Versión: 4.7.5

Implementar la funcionalidad de chat en tiempo real, permitiendo la comunicación instantánea entre los usuarios a través de WebSockets.

## Repositorio

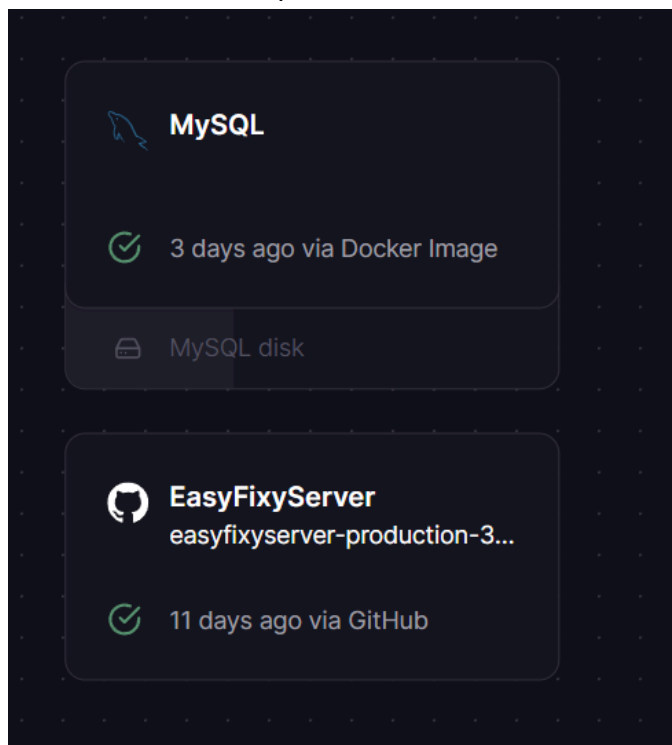
El proyecto está alojado en el repositorio **EasyFixyServer** (<https://github.com/EasyFixy/EasyFixyServer.git>) de la organización **EasyFixy**. Durante el desarrollo, se creó una rama dedicada para cada tarea, utilizando como nombre el título correspondiente en Jira.

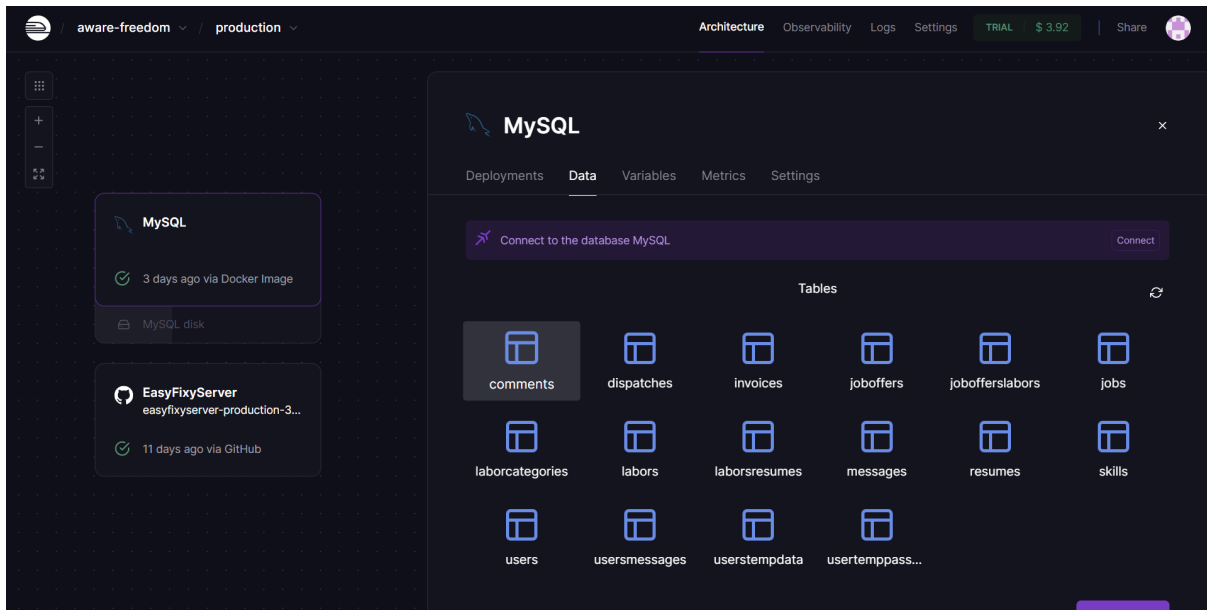
## Gitignore

```
1  # Ignorar archivo .env
2  .env
3
4  node_modules
```

## Despliegue del Back y Base de datos

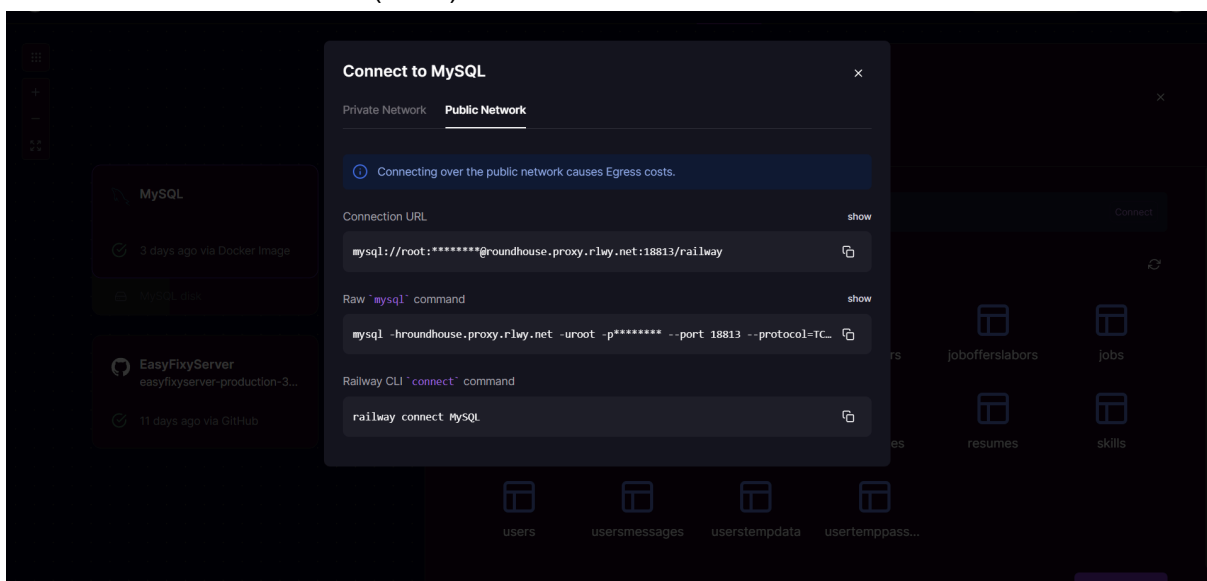
En el despliegue del backend y base de datos hicimos uso de Railway, el cual al igual que Vercel permite conectarnos a los repositorios que tengamos asociados a nuestra cuenta de GitHub, por ende el despliegue se hizo primeramente de la base de datos, la cual una vez creada dentro de railway tuvimos que conectarnos mediante SSH para añadir el esquema de la base de datos que nosotros teníamos en local:





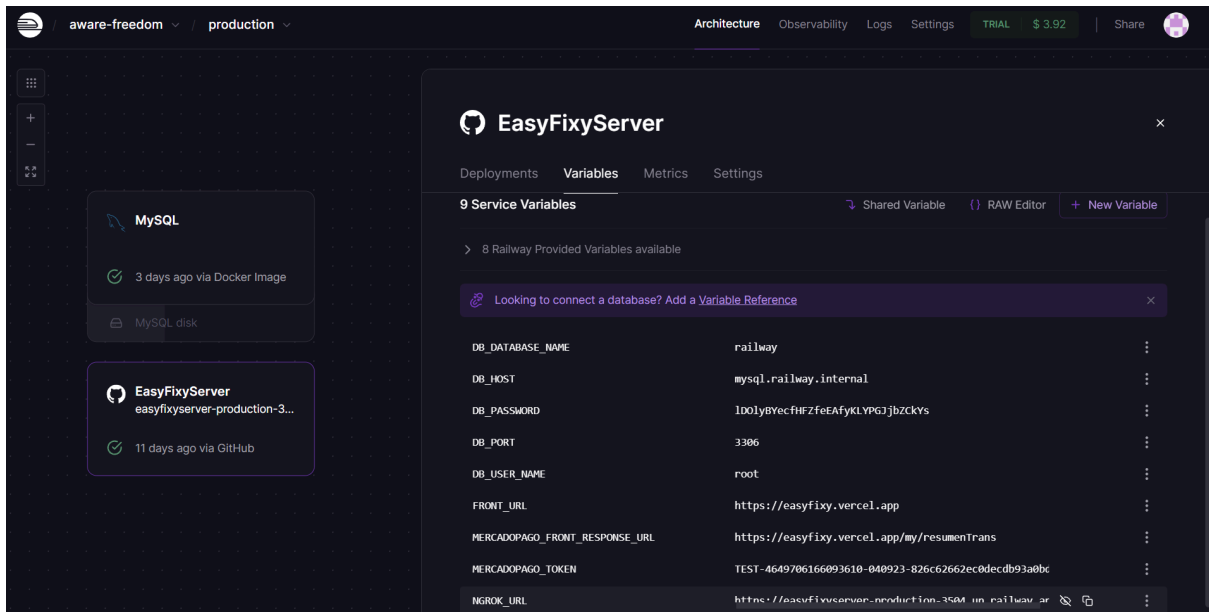
*Base de datos desplegada en Railway*

## Conexión mediante consola (CMD)



Para el despliegue del back-end dentro del mismo proyecto de railway añadimos una apartado de despliegue del proyecto en github (server-easyfixy) donde antes de hacer el despliegue permanente añadimos variables de entorno, para que funcionara absolutamente todo:

Variables de entorno:



## Base de datos

Estamos utilizando **MySQL versión 8.2.0** en un entorno **Win64** con arquitectura **x86\_64**, proporcionado por **MySQL Community Server (GPL)**. Esta versión de MySQL es robusta y eficiente, diseñada para manejar grandes volúmenes de datos y ofrecer un rendimiento optimizado. Nos permite gestionar de manera efectiva las operaciones de almacenamiento, recuperación y manipulación de datos, asegurando la integridad y consistencia de la información en nuestra aplicación.

# Diagrama entidad Relación

