Cens

Colaboradores:

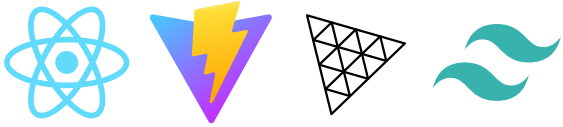
Eliam Rodríguez Pérez (@Prz-Ellam) - **Back-End**
Jan Anthony Ochoa Retta (@JanOchoa10) - **Front-End**
Ian Brandon Palacios (@IanPalace) - **DBA**
Josue Alonso Moncada Balderas (@CochueAMB) - **Front-End**
Gerardo Octavio Arias Hernández (@geohwnd) - **Front-End**

Cens

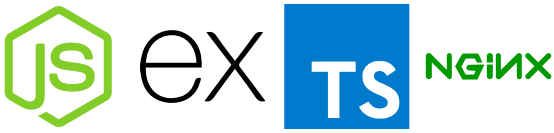
Cens es una aplicacion web enfocada en la creacion de encuestas, los usuarios de la aplicación podrán crear y compartir sus encuestas de una amplia variedad de temas, votar en encuestas y dar su opinión de las mismas. Todos los usuarios tienen un perfil y pueden seguir o ser seguidos por otros usuarios para poder ver las últimas encuestas que publiquen.
Se cuenta con un chat donde los usuarios podrán mantener contacto directo.

Tecnologías utilizadas

Frontend:



Backend:



Base de datos:

Otros:

Carpetas

- `client`: Esta carpeta la aplicación frontend utilizando React + Vite
  - `dist`: Se utiliza para almacenar los archivos generados después de la transpilación y construcción del código fuente. Este directorio contiene la versión optimizada y lista para producción de la aplicación.
  - `node_modules`: Contiene todas las dependencias necesarias para ejecutar la aplicación. Estas dependencias incluyen bibliotecas y módulos de terceros que son utilizados por el código fuente.
  - `src`: Contiene el código fuente principal de la aplicación
    - `assets`: Contiene archivos estáticos como imágenes, fuentes, archivos CSS, y otros recursos necesarios para la interfaz de usuario.
    - `components`: Es donde se almacenan los componentes reutilizables de la aplicación. Estos componentes pueden ser elementos de interfaz de usuario como botones, encabezados, formularios, etc.
    - `context`: Se utiliza para organizar y gestionar los contextos de la aplicación. Los contextos en React son una forma de compartir información entre componentes sin necesidad de pasar props a través de múltiples niveles de la jerarquía de componentes.
    - `hooks`: Se utiliza para almacenar custom hooks. Los custom hooks son funciones que encapsulan la lógica compartida entre componentes, lo que permite una reutilización efectiva del código.
    - `layouts`: Se ubican los diseños de página o componentes de alto nivel que estructuran la disposición de otros componentes.
    - `pages`: Contiene componentes que representan páginas individuales de la aplicación.
    - `services`: Se usa para almacenar módulos que interactúan con servicios externos o API, como llamadas a backend, autenticación, o cualquier otra comunicación con el servidor.
    - `utils`: Se ubican funciones y utilidades compartidas que se utilizan en varios lugares de la aplicación. Esto puede incluir funciones de ayuda, formateo de datos, validación, etc.
    - `validators`: Contiene reglas de validación para los formularios de la aplicación.
- `docs`: Contiene documentación técnica del uso de la aplicación
  - `api`: Contiene la documentación de la REST API incluyendo los endpoints, cuerpos de solicitud, respuesta y códigos de respuesta
- `server`: Contiene la aplicación de NodeJS + Typescript
  - `database`: Código fuente de la base de datos
    - `migrations`: Archivos de migración para la creación de la base de datos
    - `seeders`: Datos y valores iniciales para la base de datos
  - `dist`: Código fuente transpilado a javascript
  - `logs`: Se almacen los logs de la aplicación
  - `src`: Contiene el código fuente principal de la aplicación
    - `config`: Contiene archivos de configuración para la aplicación, como la configuración de la base de datos, variables de entorno, o cualquier otra configuración necesaria.
    - `controllers`: Se utiliza para almacenar los controladores de la aplicación. Los controladores son responsables de manejar las solicitudes HTTP y ejecutar la lógica de negocio correspondiente.
    - `middlewares`: Se guardan las funciones de middleware que pueden procesar o modificar las solicitudes HTTP antes de llegar a los controladores. Esto es útil para la validación, la

autenticación y otras tareas intermedias.
        - `models`: Contiene las definiciones de modelos de datos que representan las estructuras de la base de datos. Estos modelos son utilizados para interactuar con la base de datos utilizando un ORM (Object-Relational Mapping).
        - `routes`: Se definen las rutas de la aplicación. Aquí se mapean las solicitudes HTTP a los controladores correspondientes y se establecen las rutas de la API.
        - `services`: Alberga módulos que encapsulan la lógica de negocio de la aplicación.
        - `utils`: Se ubican funciones y utilidades compartidas que se utilizan en varios lugares de la aplicación. Esto puede incluir funciones de ayuda, formateo de datos, validación, etc.
        - `validators`: Se utiliza para almacenar validaciones de datos. Esto puede incluir validaciones de entrada de usuario, validaciones de formularios y otras validaciones personalizadas.
    - `tests`: Se encuentra el código fuente de los tests
        - `feature`: Se encuentran tests de tipo feature
        - `unit`: Se encuentran tests de tipo unit
- `web`: Contiene la información necesaria para levantar el servidor web (Nginx)
    - `certs`: Contiene los certificados SSL
    - `conf`: Contiene el archivo de configuración de Nginx