



DESARROLLADOR
WEB FULL-STACK

13 de septiembre de 2024

por:

MARIO TOMAS ISLAS CASTRO

Descripción General2

BackEnd.....3

FrontEnd.....8

Notas Adicionales.....14

Descripción General y generación de Base de datos

El proyecto **CRUD-ENTERSOL** es una aplicación web que desarrollé utilizando principalmente nodejs y react. Esta aplicación está dividida en la parte del backend y del frontend,

Estoy usando la versión más actual de node, la cual es la v20.17.0

```
C:\Users\Tom\Documents\CRUD-ENTERSOL\backend>node -v
v20.17.0
```

Adicional a eso estoy usando Wampserver el cual ya tiene configurado MySQL.

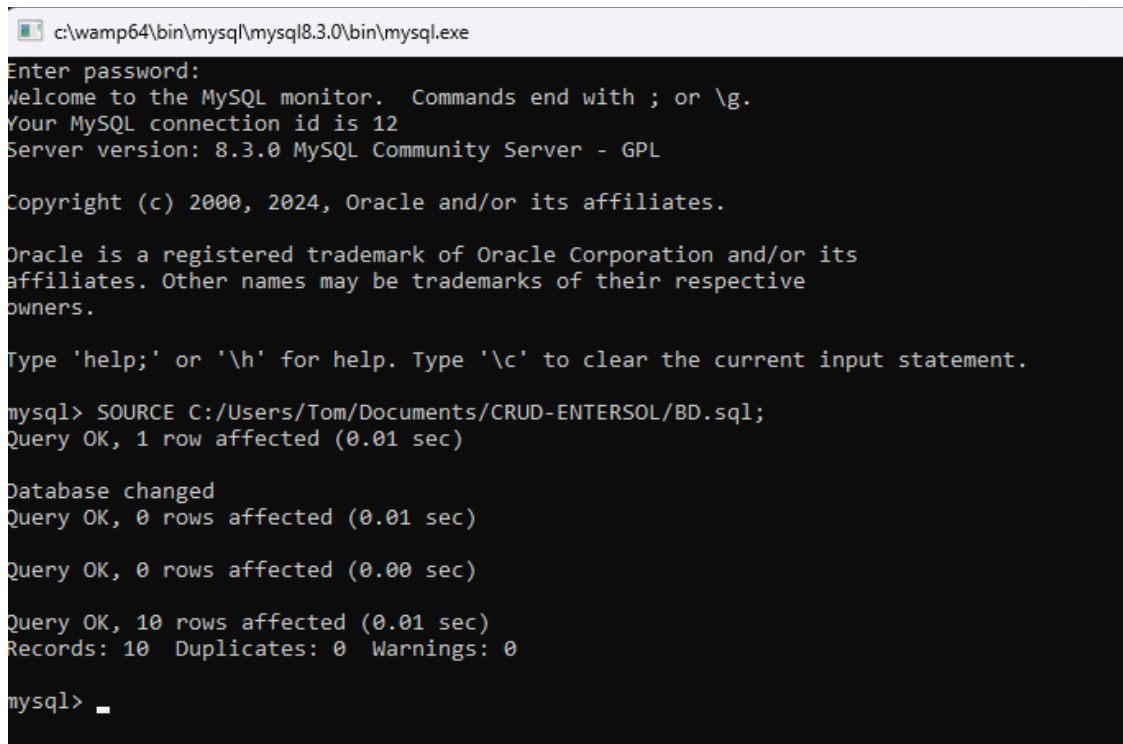
Para importar la Base de datos con algunos registros de prueba, es necesario abrir la consola de MySQL y ejecutar el comando

```
SOURCE {ruta}/CRUD-ENTERSOL/BD.sql;
```

Donde {ruta} es la carpeta donde tienen el proyecto CRUD-ENTERSOL

en mi caso el proyecto lo tengo dentro de mi carpeta de documentos, por lo que yo usé

```
SOURCE C:/Users/Tom/Documents/CRUD-ENTERSOL/BD.sql;
```



```
c:\wamp64\bin\mysql\mysql8.3.0\bin\mysql.exe
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 12
Server version: 8.3.0 MySQL Community Server - GPL

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> SOURCE C:/Users/Tom/Documents/CRUD-ENTERSOL/BD.sql;
Query OK, 1 row affected (0.01 sec)

Database changed
Query OK, 0 rows affected (0.01 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> _
```

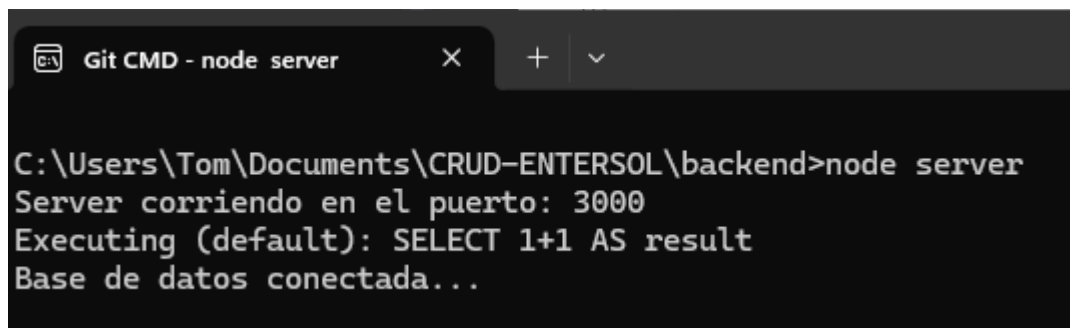
Eso genera la base de datos y 10 registros de prueba

BackEnd (Endpoints)

En mi caso uso las credenciales que tenia por defecto wamp server, por lo que en mi caso, en el archivo situado en backend/config/config.js tengo estos datos,

```
module.exports = {
  development: {
    username: 'root',
    password: null,
    database: 'entersol',
    host: '127.0.0.1',
    dialect: 'mysql'
  }
};
```

Una vez configurado el archivo (y ya instaladas las dependencias con **npm install**) porcedemos a utilizar el comando **node server** para poder ejecutar la parte del backend:



```
Git CMD - node server
C:\Users\Tom\Documents\CRUD-ENTERSOL\backend>node server
Server corriendo en el puerto: 3000
Executing (default): SELECT 1+1 AS result
Base de datos conectada...
```

Esto nos habilitará los endpoints, los cuales son:

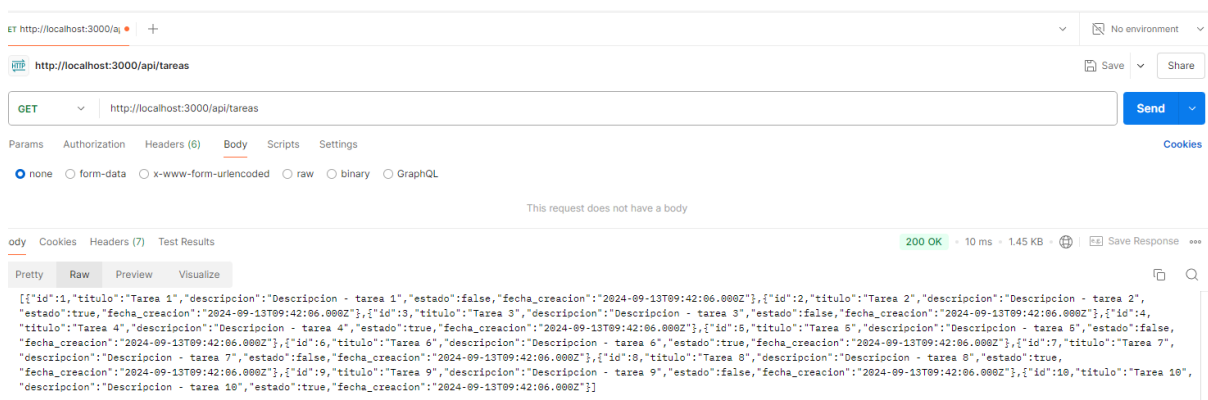
1. GET /tareas

Para obtener todas las tareas:

Método: GET

URL: <http://localhost:3000/api/tareas>

Acción: Devuelve todas las tareas almacenadas en la base de datos.



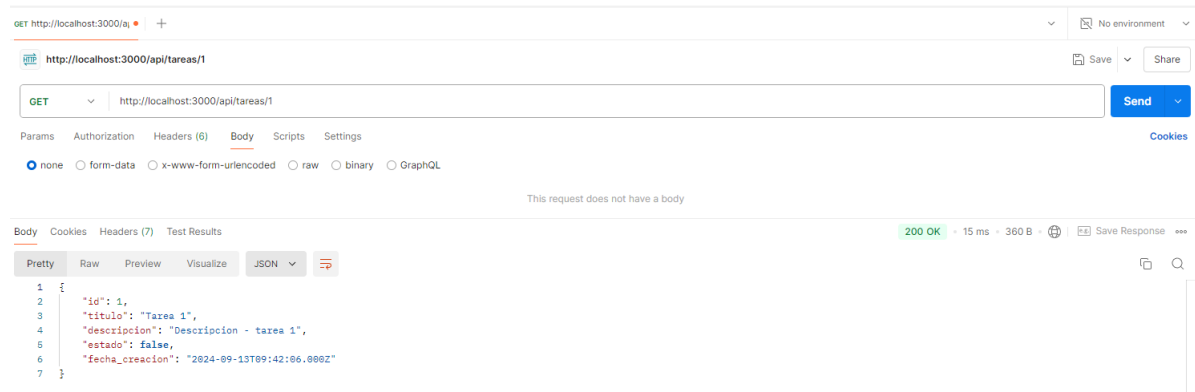
2. GET /tareas/

Para obtener una tarea específica por su ID:

Método: GET

URL: <http://localhost:3000/api/tareas/id> (sustituye id con el ID real de la tarea que quieres obtener)

Acción: Devuelve los detalles de una tarea específica.



3. POST /tareas

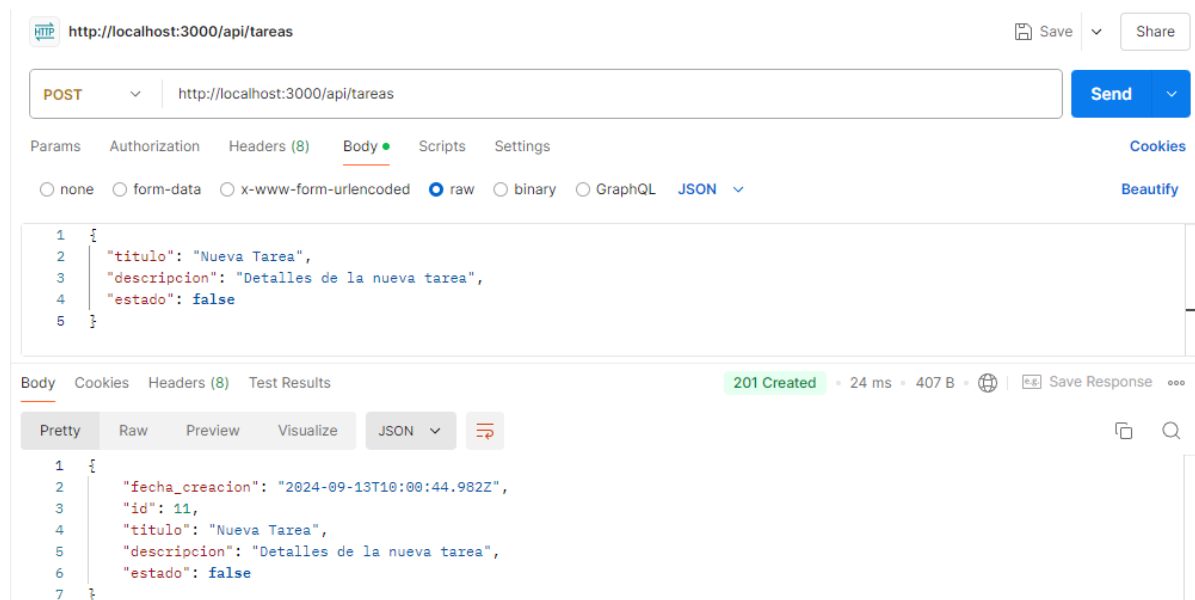
Para crear una nueva tarea:

Método: POST

URL: <http://localhost:3000/api/tareas>

Body: (formato JSON)

```
{
  "titulo": "Nueva Tarea",
  "descripcion": "Detalles de la nueva tarea",
  "estado": false
}
```



4. PUT /tareas/

Para actualizar una tarea existente:

Método: PUT

URL: <http://localhost:3000/api/tareas/id> (sustituye id con el ID real de la tarea que quieres actualizar)

Body: (formato JSON)

```
{
  "titulo": "Tarea Actualizada",
  "descripcion": "Detalles actualizados de la tarea",
  "estado": true
}
```

The screenshot shows a REST client interface with the following details:

- Method:** PUT
- URL:** <http://localhost:3000/api/tareas/1>
- Body (JSON):**

```
{
  "titulo": "Tarea Actualizada",
  "descripcion": "Detalles actualizados de la tarea",
  "estado": true
}
```
- Response:** 200 OK, 20 ms, 298 B. The response body is:

```
{
  "message": "Tarea actualizada"
}
```

5. DELETE /tareas/

Para eliminar una tarea:

Método: DELETE

URL: <http://localhost:3000/api/tareas/id> (sustituye id con el ID real de la tarea que quieres eliminar)

Acción: Elimina la tarea especificada y devuelve un mensaje de confirmación.

The screenshot shows a REST client interface with the following details:

- Method:** DELETE
- URL:** <http://localhost:3000/api/tareas/4>
- Response:** 200 OK, 18 ms, 296 B. The response body is:

```
{
  "message": "Tarea eliminada"
}
```

Adicional a eso, se realiza la validación de que el titulo no puede ser null y de que en caso de que el estado no venga definido, sea false y esa estructura se realiza directamente en el modelo del objeto de sequelize, para que lo haga en automatico al analizar el json que enviamos.

```
const Tarea = sequelize.define('Tarea', {
  titulo: {
    type: DataTypes.STRING,
    allowNull: false
  },
  descripcion: {
    type: DataTypes.TEXT
  },
  estado: {
    type: DataTypes.BOOLEAN,
    defaultValue: false
  },
  fecha_creacion: {
    type: DataTypes.DATE,
    defaultValue: DataTypes.NOW
  }
}, {
  tableName: 'tarea',
  timestamps: false
});
```

HTTP http://localhost:3000/api/tareas

POST http://localhost:3000/api/tareas

Params Authorization Headers (8) Body ● Scripts Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL JSON ▾

```
1 {
2   "titulo": null,
3   "descripcion": "Detalles de la nueva tarea",
4   "estado": false
5 }
```

Body Cookies Headers (8) Test Results 400 Bad Request

Pretty Raw Preview Visualize JSON ▾

```
1 {
2   "message": "notNull Violation: Tarea.titulo cannot be null"
3 }
```



http://localhost:3000/api/tareas

POST



http://localhost:3000/api/tareas

Params Authorization Headers (8) **Body** Scripts Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL [JSON](#)

```
1 {  
2   "titulo": "tarea sin estatus",  
3   "descripcion": "Detalles de la nueva tarea"  
4 }
```

Body Cookies Headers (8) Test Results

Pretty

Raw

Preview

Visualize

JSON



```
1 {  
2   "estado": false,  
3   "fecha_creacion": "2024-09-13T10:08:11.632Z",  
4   "id": 13,  
5   "titulo": "tarea sin estatus",  
6   "descripcion": "Detalles de la nueva tarea"  
7 }
```


FrontEnd

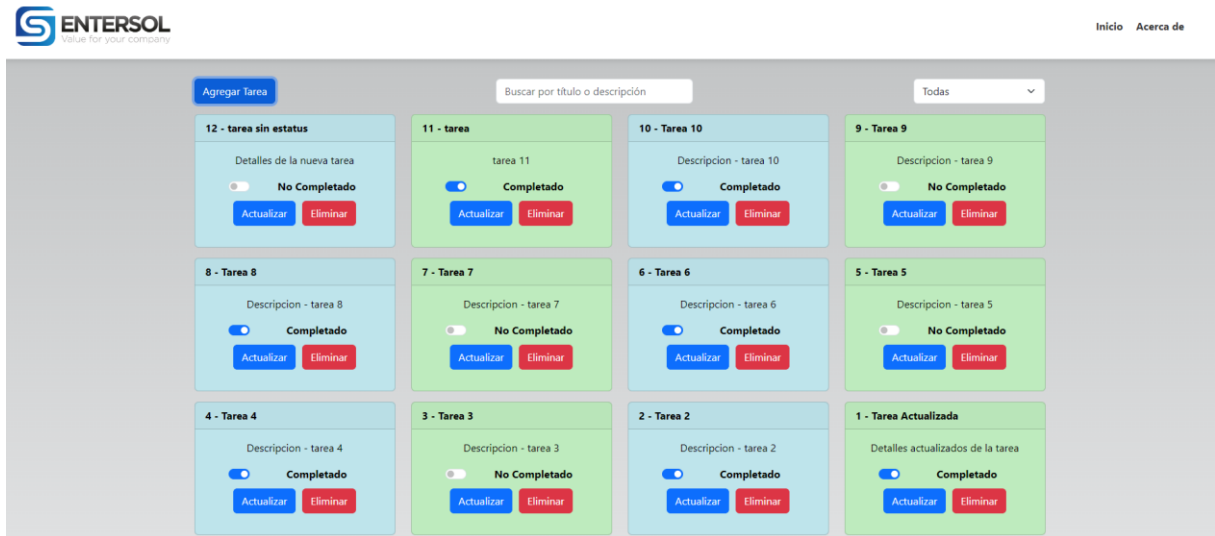
Para esta parte es necesario ir a la carpeta frontend

Utilizando el comando: **cd frontend**

Y posteriormente instalaremos las dependencias con: **npm install**

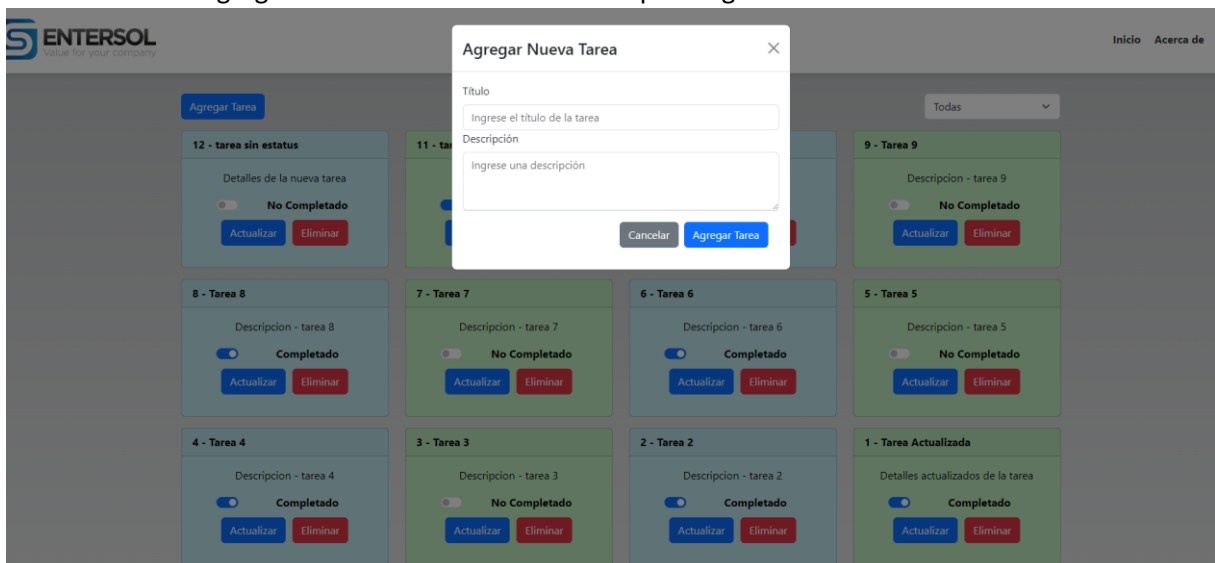
```
PS C:\Users\Tom\Documents\CRUD-ENTERSOL> cd .\frontend\  
PS C:\Users\Tom\Documents\CRUD-ENTERSOL\frontend> npm install
```

Una vez que se instalen simplemente lo inicializan con el comando: **npm start**

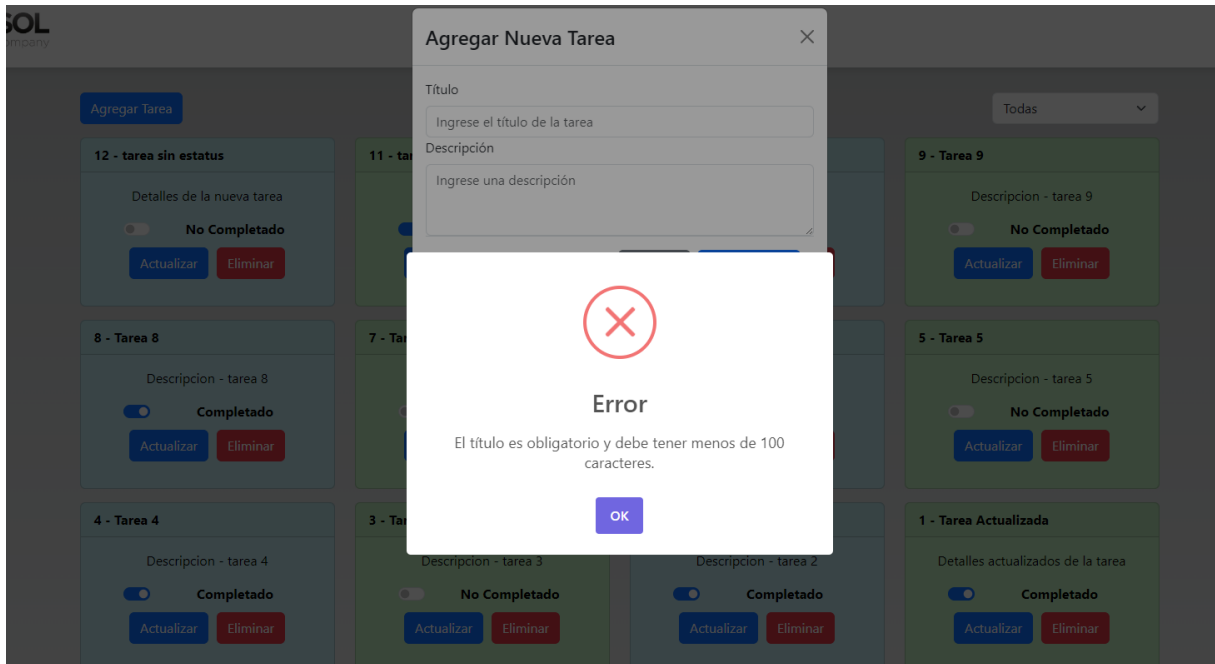


Eso lo que hará es directamente poner la pagina de inicio con las tareas cargadas previamente en la base de datos.

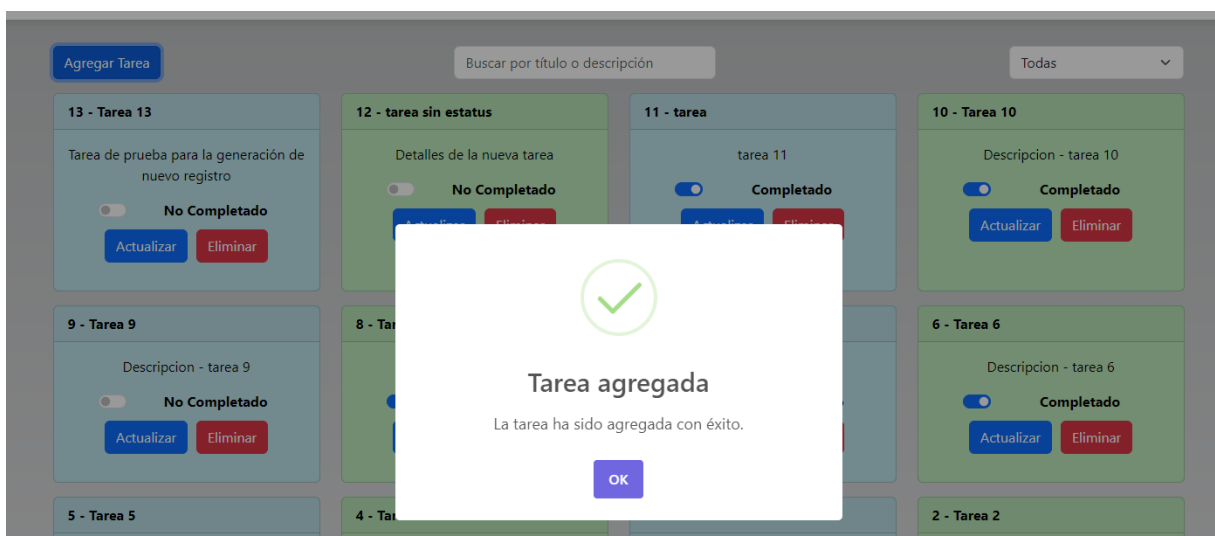
Con el botón de agregar Tarea se abrirá el formulario para ingresar una nueva tarea.



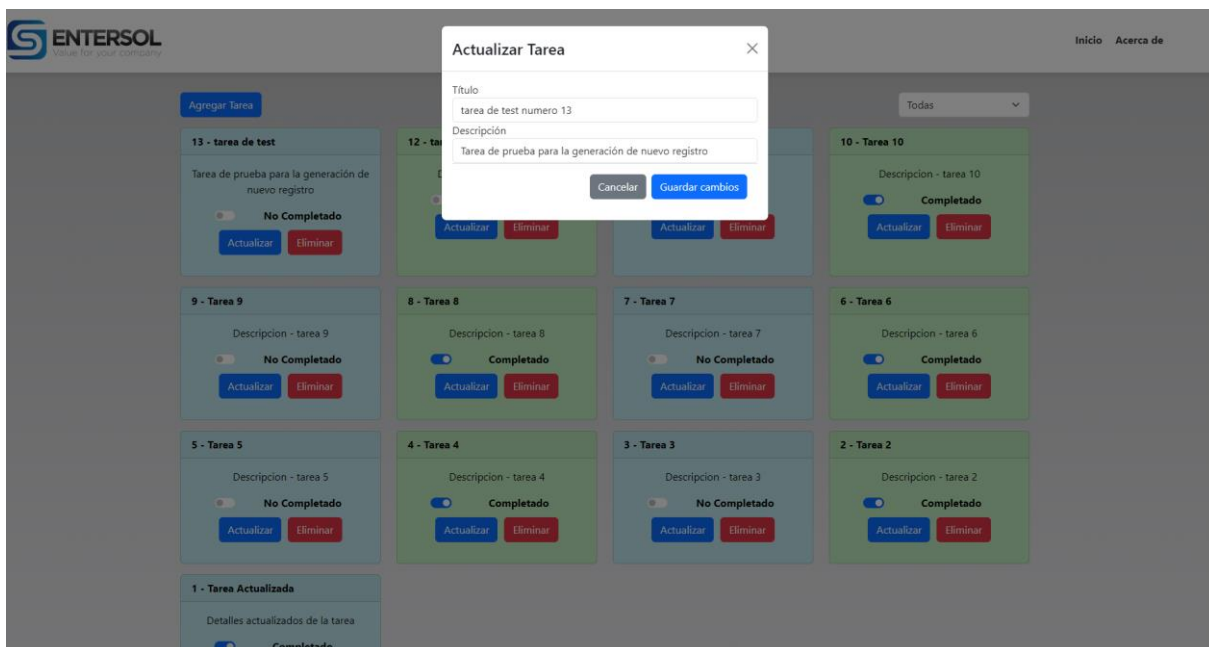
De igual manera se validó la parte de que no debe estar vacío el título y que debe tener menos de 100 caracteres (cosa que de igual manera se valida en el endpoint).



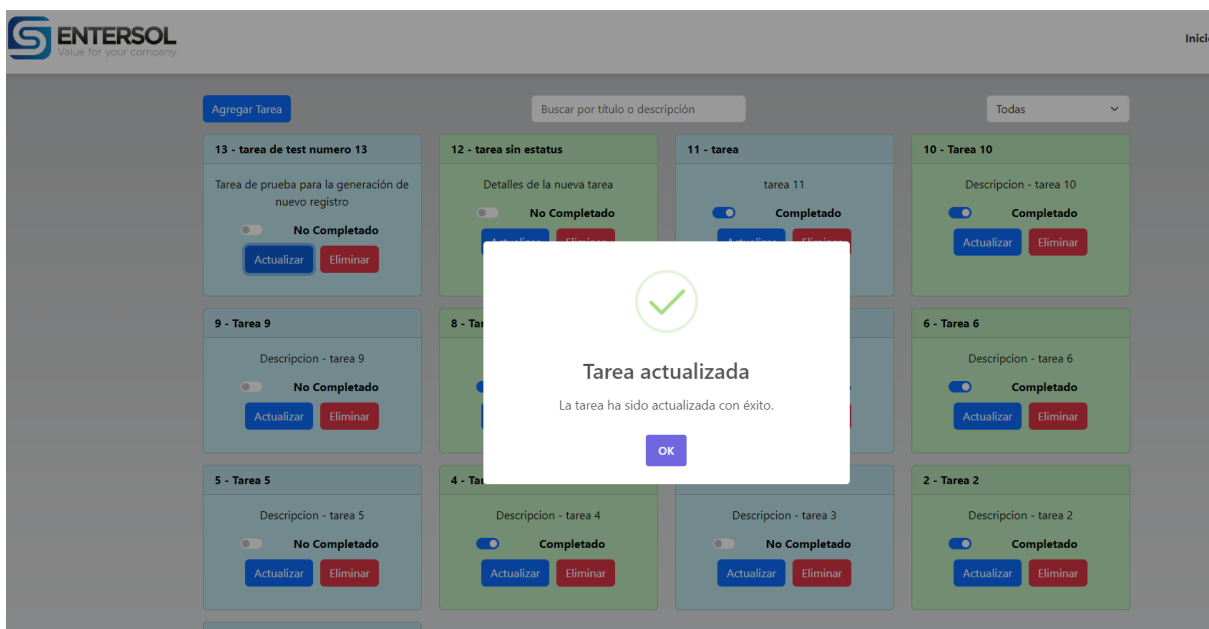
Una vez que el registro es valido, directamente se agrega y se recarga la lista, por lo que las tareas nuevas siempre irán hasta arriba y las más viejas se iran desplazando hacia abajo cada que se vaya llenando una fila extra.



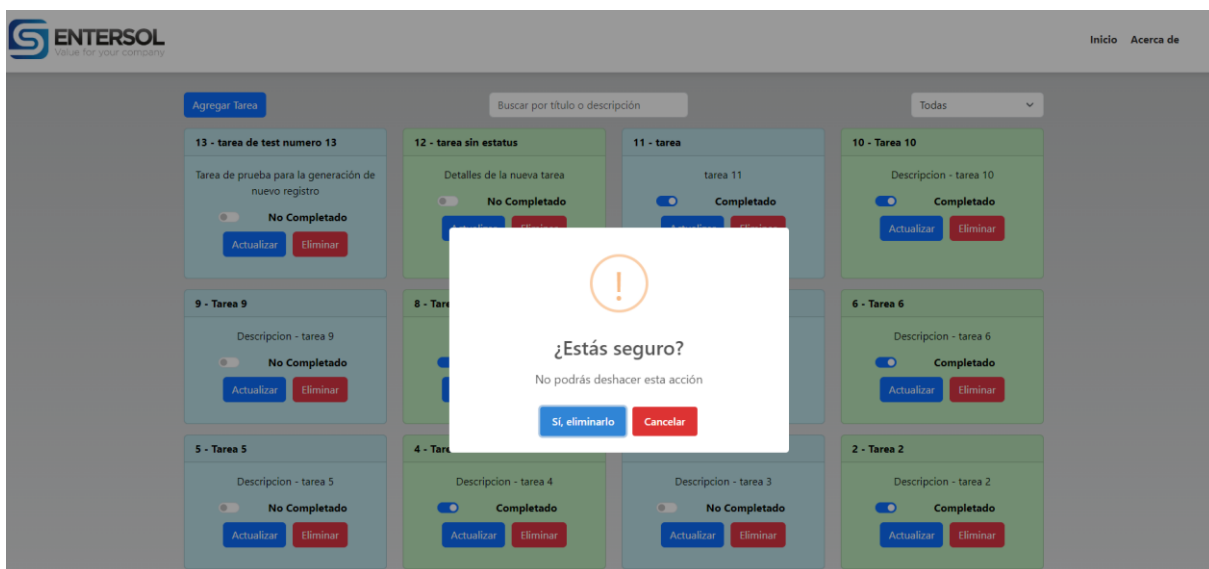
De igual manera el botón de Actualizar Tarea trae los datos previamente llenados de la Tarea que seleccionemos y al momento de guardar igual se actualiza la lista



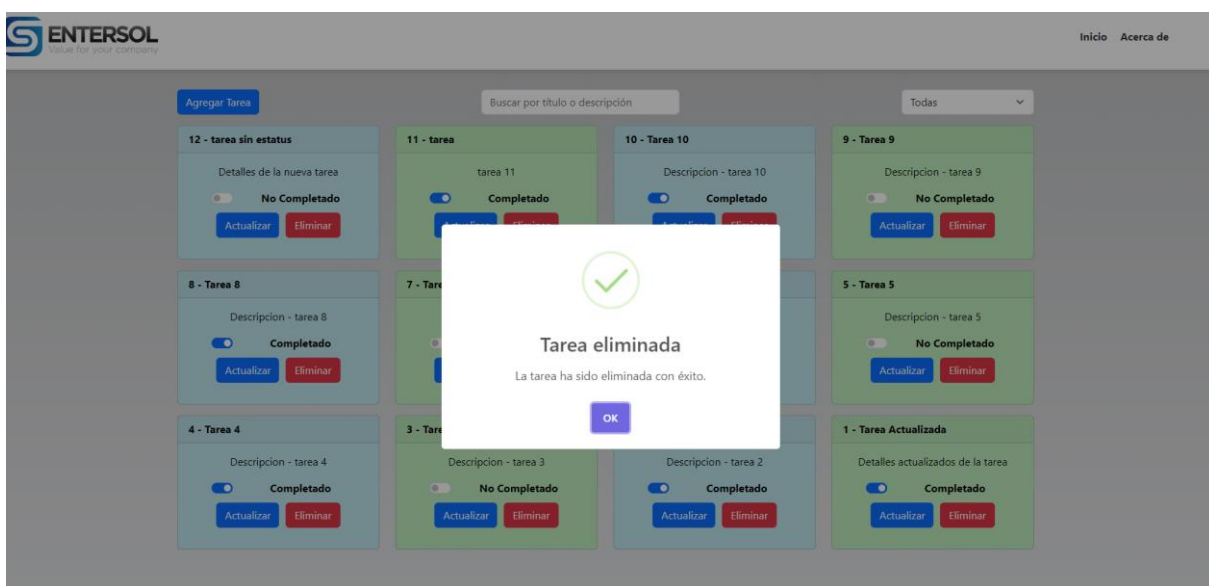
Y de la misma forma nos arroja una alerta de que se actualizó correctamente



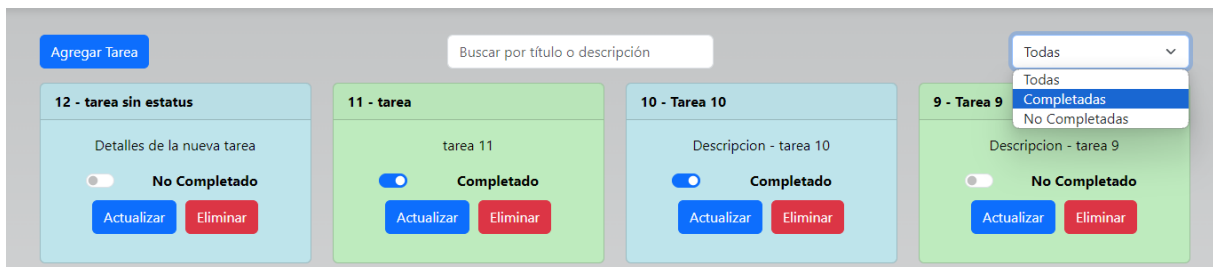
Al momento de eliminar una Tarea, nos pedirá confirmación



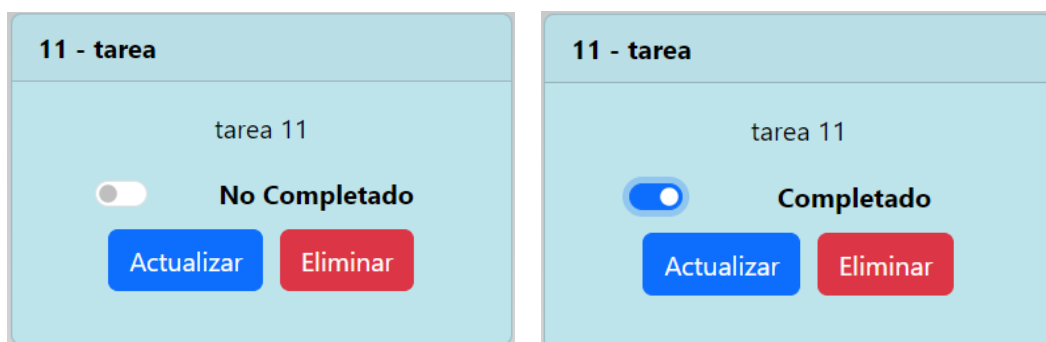
Y en caso de eliminarla, se actualizará la lista automáticamente



Adicional a Eso se pusieron filtros Por Titulo o Descripción y a parte el filtro de las tareas completadas y no completadas y eso va filtrando todas en tiempo real



A parte se integró un toggle button para cada tarea, que marca la tarea como completada o como no completada y al momento de cambiarle el estatus ahí, en automático se cambia en la base de datos.



Y de igual manera se utilizó Bootstrap para manejar la parte responsiva en caso de que la pantalla se haga más angosta y no quepan los 4 elementos por cada fila

Agregar Tarea

Buscar por título o descripción

Todas

12 - tarea sin estatus

Detalles de la nueva tarea

No Completado

Actualizar

Eliminar

11 - tarea

tarea 11

Completado

Actualizar

Eliminar

10 - Tarea 10

Descripcion - tarea 10

Completado

Actualizar

Eliminar

9 - Tarea 9

Descripcion - tarea 9

Notas adicionales

Se eligieron las siguientes tecnologías por las siguientes razones:

- **Redux:** Para manejar el estado global de la aplicación de manera eficiente, facilitando la sincronización de los cambios de estado entre componentes sin necesidad de prop-drilling. Permite una mejor organización y escalabilidad de la lógica de la aplicación.
- **Express:** Un framework minimalista para Node.js que permite construir una API REST de manera rápida y eficiente. Su flexibilidad lo hace ideal para proyectos donde se busca control total sobre la estructura del backend.
- **Sequelize:** Un ORM para Node.js que simplifica la interacción con bases de datos relacionales como MySQL. Facilita la creación y gestión de modelos y proporciona funcionalidades avanzadas como migraciones y validaciones de datos.
- **Bootstrap:** Un framework CSS que acelera el desarrollo frontend con estilos predeterminados y componentes responsivos, asegurando un diseño coherente sin necesidad de escribir CSS desde cero.
- **Hooks (useState, useEffect):** Utilizados para gestionar el estado y los efectos secundarios en los componentes de React de forma declarativa. useState facilita la gestión del estado local y useEffect permite ejecutar efectos secundarios como la obtención de datos, mejorando el ciclo de vida de los componentes y optimizando la reactividad.

Repositorio de Código

El código fuente del proyecto está disponible en GitHub en el siguiente enlace:

<https://github.com/RayEmer/CRUD-ENTERSOL>

Video demostrativo

https://youtu.be/od1j_9b-FOM