

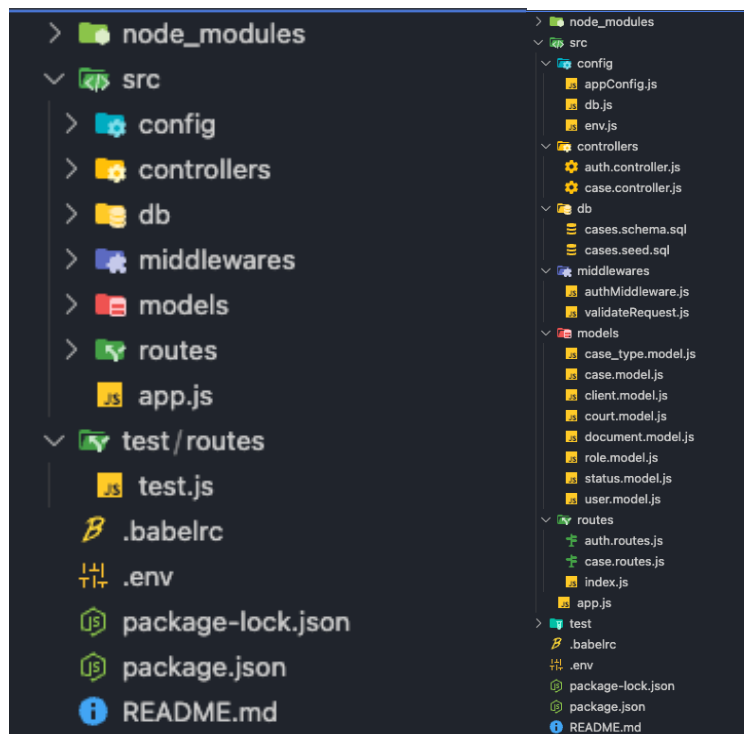
Hito 4 - Aplicación Backend en Node y Express

Alumno: Nicolás Chávez Garzón

Grupo: 1

1. **Configurar el proyecto aplicando el patrón MVC para encapsular los modelos y los controladores y endpoints en carpetas separadas.**

Se modifica el proyecto para que tenga el patrón MVC como se muestra a continuación:



En este caso el proyecto busca crear una API para administrar los casos de un estudio jurídico de abogados. Se crearon distintos modelos para manejar tanto los casos como la documentación y los roles. Los controladores se dividen en 2, uno de “autorización” que permite registrarse y hacer login para obtener un token y otro de “casos” que contiene el CRUD para administrar los casos mediante el token recibido. Además, las rutas se administran en la carpeta “routes”, teniendo las rutas de autorización y casos para ser llamadas en el index.js y posteriormente en app.js.

Por otro lado, se crearon también las siguientes carpetas para administrar mejor los recursos:

- a) **Carpeta “config”:** Contiene scripts para administrar las variables de entorno que se utilizan para la base de datos y los secretos de JWT.
- b) **Carpeta “db”:** Contiene los schemas y seeds para rellenar las tablas con información de ejemplo.
- c) **Carpeta “middlewares”:** Contiene las lógicas de los middlewares utilizados para permitir una mejor administración de ellos.

- d) **Carpeta “test”**: Se mantienen las pruebas del proceso de registro, login y consulta de casos usando el token recibido.

2. Instalar y configurar la dependencia de Sequelize para realizar los procesos de consultas a la base de datos del proyecto.

Se instalan las dependencias y se crean modelos para cada una de las tablas a utilizar para administrar los casos. En “src/config/db.js” instanciamos el Sequelize con las variables de entorno necesarias para autenticarnos con Postgres:

```
src > config > db.js > [?] default
1  import { Sequelize } from 'sequelize';
2  import getEnv from './env.js';
3
4  const sequelize = new Sequelize(getEnv('DB_NAME'), getEnv('DB_USER'), getEnv('DB_PASSWORD'), {
5    host: getEnv('DB_HOST'),
6    dialect: 'postgres',
7  });
8
9  sequelize.authenticate()
10   .then(() => console.log('Database connected'))
11   .then(() => sequelize.sync())
12   .catch(err => console.error('Error connecting to database:', err));
13
14 export default sequelize;
```

Cada tabla tiene su propio modelo definido en “src/models”. La siguiente imagen muestra el modelo de Cases:

```
import { DataTypes } from 'sequelize';
import sequelize from '../config/db.js';
import CaseType from './case_type.model.js';
import Court from './court.model.js';
import User from './user.model.js';
import Status from './status.model.js';

const Case = sequelize.define('Case', {
  title: {
    type: DataTypes.STRING,
    allowNull: false,
  },
  description: {
    type: DataTypes.TEXT,
    allowNull: true,
  },
  case_type_id: {
    type: DataTypes.INTEGER,
    references: {
      model: CaseType,
      key: 'id',
    },
    allowNull: false,
  },
  court_id: {
    type: DataTypes.INTEGER,
    references: {
      model: Court,
      key: 'id',
    },
    allowNull: false,
  },
  created_by: {
    type: DataTypes.INTEGER,
    references: {
      model: User,
      key: 'id',
    },
    allowNull: false,
  },
  status_id: {
    type: DataTypes.INTEGER,
    references: {
      model: Status,
      key: 'id',
    },
    allowNull: false,
  },
  start_date: {
    type: DataTypes.DATE,
    allowNull: false,
  },
  end_date: {
    type: DataTypes.DATE,
    allowNull: true,
  },
}, {
  tableName: 'cases',
  timestamps: true
});

export default Case;
```

3. Verificar el funcionamiento del sistema haciendo uso de ThunderClient o Postman. Considerar probar los endpoints para CRUD.

Inicializamos el servidor con la API utilizando nodemon y el ambiente dev:

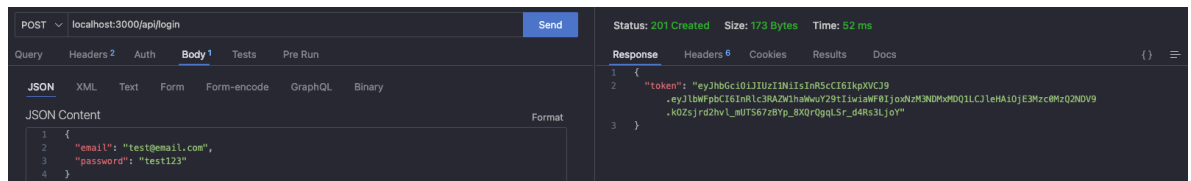
```
(base) n@n436g-hghkdjfw Mito 4 % npm run dev
> hito-lbl:0.0 dev
> nodemon src/app.js
[nodemon] 3.1.7
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting node src/app.js
Servidor encendido
Executing (default): SELECT 1+1 AS result
Database connected
Executing (default): SELECT table_name FROM information_schema.tables WHERE table_schema = 'public' AND table_name = 'roles'
Executing (default): SELECT i.relname AS name, ix.indisprimary AS primary, ix.indisunique AS unique, ix.indkey AS indkey, array_agg(a.attname) AS column_names, pg_get_indexdef(ix.indexrelid) AS definition FROM pg_class t, pg_class i, pg_index ix, pg_attribute a WHERE t.oid = ix.indexrelid AND i.oid = ix.indexrelid AND a.attrelid = t.oid AND t.relkind = 'r' and i.relname = 'roles' GROUP BY i.relname, ix.indexrelid, ix.indisprimary, ix.indisunique, ix.indkey ORDER BY i.relname
Executing (default): SELECT table_name FROM information_schema.tables WHERE table_schema = 'public' AND table_name = 'users'
Executing (default): SELECT i.relname AS name, ix.indisprimary AS primary, ix.indisunique AS unique, ix.indkey AS indkey, array_agg(a.attname) AS column_names, pg_get_indexdef(ix.indexrelid) AS definition FROM pg_class t, pg_class i, pg_index ix, pg_attribute a WHERE t.oid = ix.indexrelid AND i.oid = ix.indexrelid AND a.attrelid = t.oid AND t.relkind = 'r' and i.relname = 'users' GROUP BY i.relname, ix.indexrelid, ix.indisprimary, ix.indisunique, ix.indkey ORDER BY i.relname
Executing (default): SELECT table_name FROM information_schema.tables WHERE table_schema = 'public' AND table_name = 'case_types'
Executing (default): SELECT i.relname AS name, ix.indisprimary AS primary, ix.indisunique AS unique, ix.indkey AS indkey, array_agg(a.attname) AS column_names, pg_get_indexdef(ix.indexrelid) AS definition FROM pg_class t, pg_class i, pg_index ix, pg_attribute a WHERE t.oid = ix.indexrelid AND i.oid = ix.indexrelid AND a.attrelid = t.oid AND t.relkind = 'r' and i.relname = 'case_types' GROUP BY i.relname, ix.indexrelid, ix.indisprimary, ix.indisunique, ix.indkey ORDER BY i.relname
Executing (default): SELECT table_name FROM information_schema.tables WHERE table_schema = 'public' AND table_name = 'courts'
Executing (default): SELECT i.relname AS name, ix.indisprimary AS primary, ix.indisunique AS unique, ix.indkey AS indkey, array_agg(a.attname) AS column_names, pg_get_indexdef(ix.indexrelid) AS definition FROM pg_class t, pg_class i, pg_index ix, pg_attribute a WHERE t.oid = ix.indexrelid AND i.oid = ix.indexrelid AND a.attrelid = t.oid AND t.relkind = 'r' and i.relname = 'courts' GROUP BY i.relname, ix.indexrelid, ix.indisprimary, ix.indisunique, ix.indkey ORDER BY i.relname
Executing (default): SELECT table_name FROM information_schema.tables WHERE table_schema = 'public' AND table_name = 'status'
```

- a. **POST “/api/register”**: Nos registramos con un nuevo usuario para posteriormente hacer login y confirmamos que queda guardado en la base de datos.

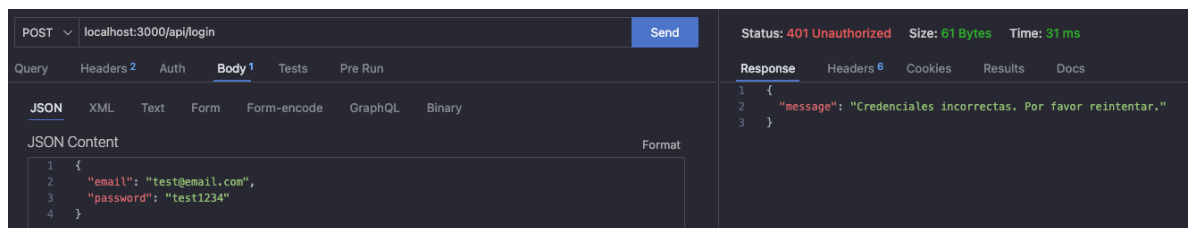


id	name	email	phone	password	role_id	createdAt	updatedAt
1	Test	test@email.com	12345678	\$2a\$08\$IDvPpXnc0S5zQLpGXKIh5kq7FSTBIK9NTeaYVnuhPpK6iOPMta	1	2025-01-21 00:34:46.356-03	2025-01-21 00:34:46.356-03

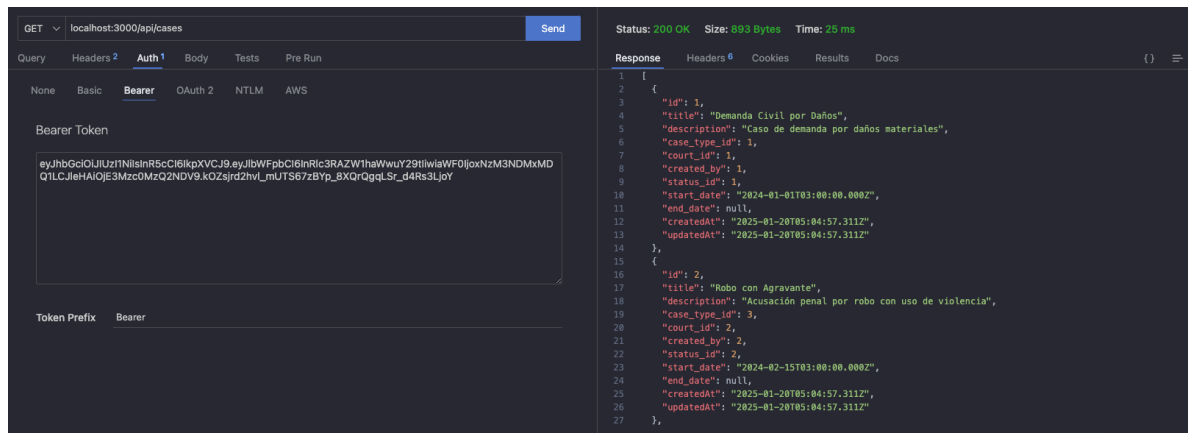
- b. **POST “/api/login”**: Enviamos las credenciales creadas anteriormente para recibir un token que nos permitirá autenticarnos para obtener los casos.



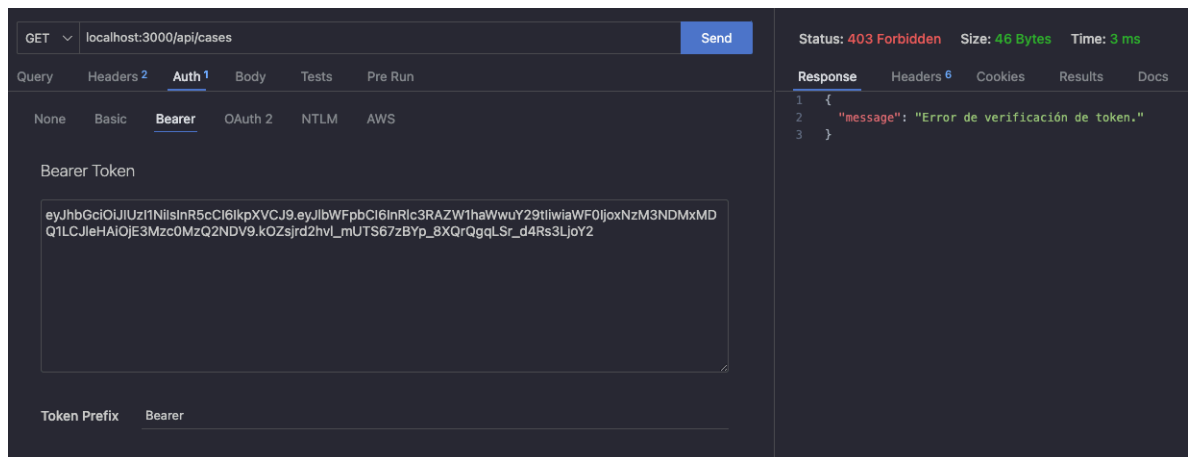
La API también maneja los casos en donde las credenciales sean incorrectas.



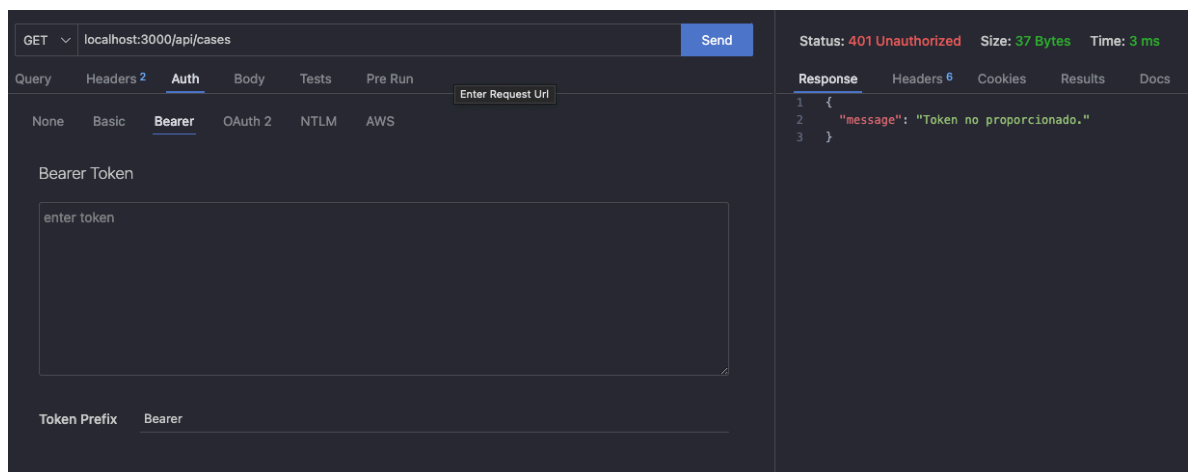
- c. **POST “/api/cases”**: Con el token recibido anteriormente, lo enviamos a través de “Auth” para recibir todos los casos del portafolio.



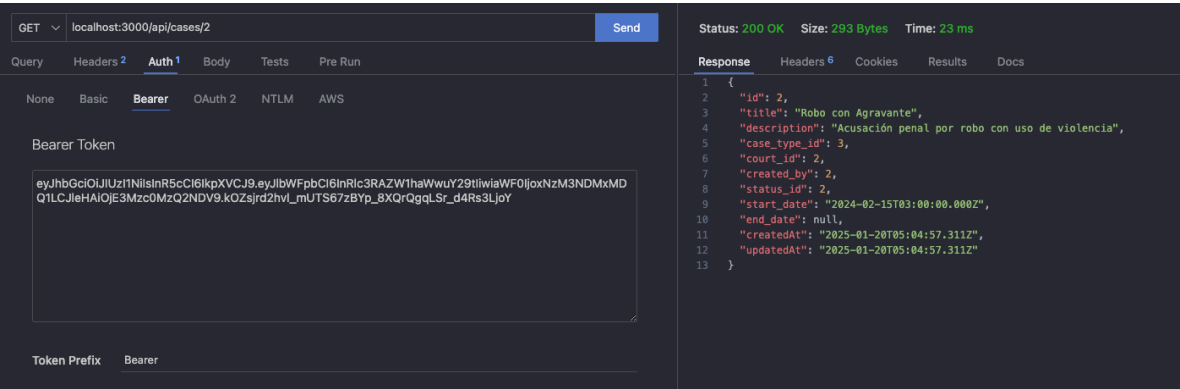
La API también maneja los casos en donde el token no es válido.



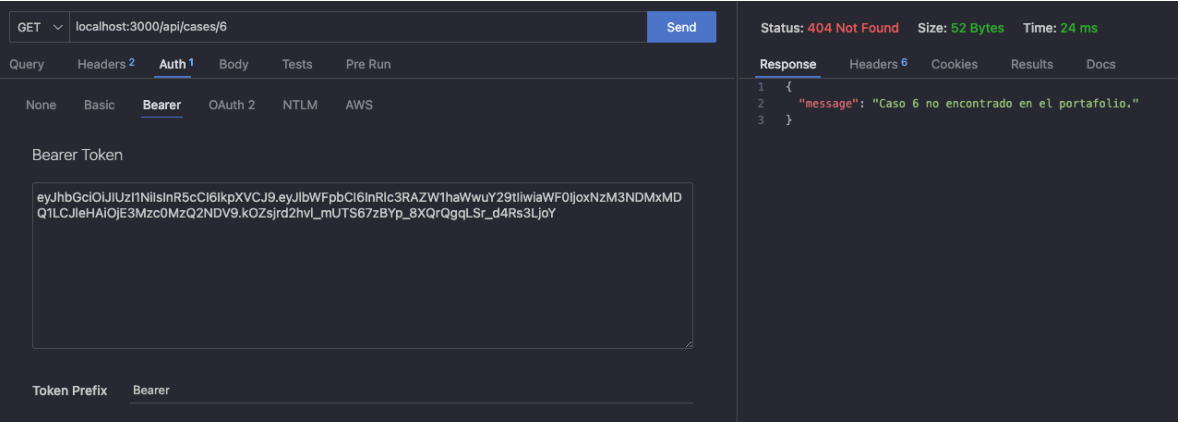
Además, también se maneja el caso cuando no se especifica el token.



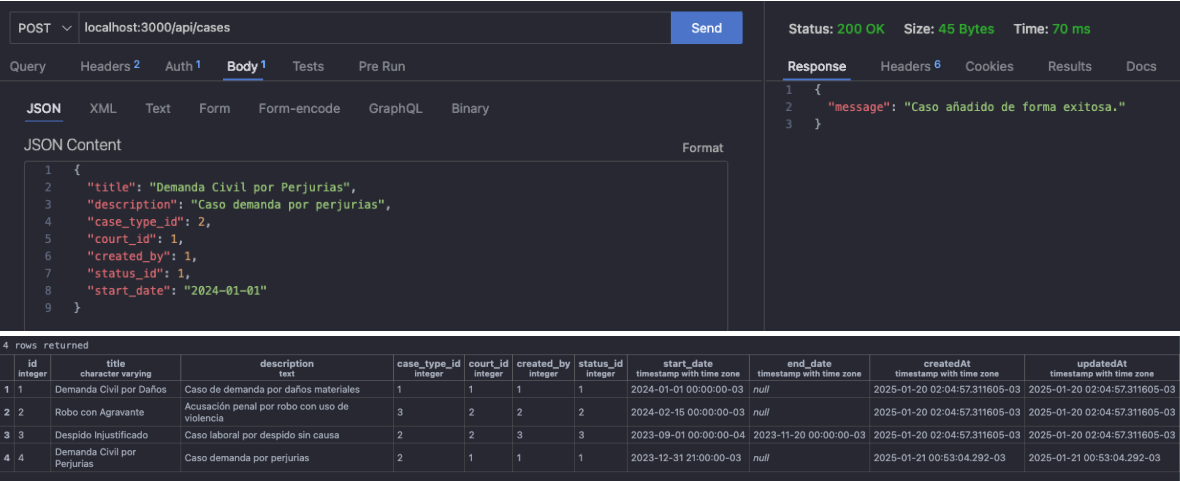
- d. **GET “/api/cases/:id”**: Con el token recibido anteriormente, también podemos buscar un caso específico mediante el id entregado como parámetro.



También se maneja el caso cuando el id no existe.



e. **POST “/api/cases”**: Se pueden crear nuevos casos entregando la información del mismo.



f. PUT “/api/cases/:id”: Se puede actualizar el estado de un caso.

PUTlocalhost:3000/api/cases/4Send

Status: 200 OKSize: 45 BytesTime: 42 ms

QueryHeaders 2Auth 1Body 1TestsPre Run

ResponseHeaders 6CookiesResultsDocs

JSONXMLTextFormForm-encodeGraphQLBinary

JSON ContentFormat

```
1 {
2   "status_id": 2
3 }
```

1 {
2 "message": "Caso 4: status actualizado a 2."
3 }

4 rows returned

	id integer	title character varying	description text	case_type_id integer	court_id integer	created_by integer	status_id integer	start_date timestamp with time zone	end_date timestamp with time zone	createdAt timestamp with time zone	updatedAt timestamp with time zone
1	1	Demanda Civil por Daños	Caso de demanda por daños materiales	1	1	1	1	2024-01-01 00:00:00-03	null	2025-01-20 02:04:57.311605-03	2025-01-20 02:04:57.311605-03
2	2	Robo con Agravante	Acusación penal por robo con uso de violencia	3	2	2	2	2024-02-15 00:00:00-03	null	2025-01-20 02:04:57.311605-03	2025-01-20 02:04:57.311605-03
3	3	Despido Injustificado	Caso laboral por despido sin causa	2	2	3	3	2023-09-01 00:00:00-04	2023-11-20 00:00:00-03	2025-01-20 02:04:57.311605-03	2025-01-20 02:04:57.311605-03
4	4	Demanda Civil por Perjurias	Caso demanda por perjurias	2	1	1	2	2023-12-31 21:00:00-03	null	2025-01-21 00:53:04.292-03	2025-01-21 01:02:29.492-03

g. DELETE “/api/cases/:id”: Se puede eliminar un caso por su id.

DELETElocalhost:3000/api/cases/4Send

Status: 200 OKSize: 45 BytesTime: 51 ms

QueryHeaders 2Auth 1Body 1TestsPre Run

ResponseHeaders 6CookiesResultsDocs

JSONXMLTextFormForm-encodeGraphQLBinary

JSON ContentFormat

```
1
```

1 {
2 "message": "Caso 4: eliminado exitosamente."
3 }

3 rows returned

	id integer	title character varying	description text	case_type_id integer	court_id integer	created_by integer	status_id integer	start_date timestamp with time zone	end_date timestamp with time zone	createdAt timestamp with time zone	updatedAt timestamp with time zone
1	1	Demanda Civil por Daños	Caso de demanda por daños materiales	1	1	1	1	2024-01-01 00:00:00-03	null	2025-01-20 02:04:57.311605-03	2025-01-20 02:04:57.311605-03
2	2	Robo con Agravante	Acusación penal por robo con uso de violencia	3	2	2	2	2024-02-15 00:00:00-03	null	2025-01-20 02:04:57.311605-03	2025-01-20 02:04:57.311605-03
3	3	Despido Injustificado	Caso laboral por despido sin causa	2	2	3	3	2023-09-01 00:00:00-04	2023-11-20 00:00:00-03	2025-01-20 02:04:57.311605-03	2025-01-20 02:04:57.311605-03