Actividad En Linea-CRUD

app.js

```
const express = require('express');
const app = express();
const cors = require('cors');

// Middleware
app.use(cors());
app.use(express.json());

// Rutas
app.use('/api/articulos', require('./routes/articulos'));
app.use('/api/clientes', require('./routes/clientes'));
app.use('/api/empleados', require('./routes/empleados'));
app.use('/api/proveedores', require('./routes/proveedores'));

// Puerto
const PORT = process.env.PORT || 3000;
app.listen(PORT, () => {
    console.log(`Servidor corriendo en el puerto ${PORT}`);
});
```

- Este es el archivo principal que inicia el servidor.
- express() crea una nueva aplicación Express
- cors() permite peticiones desde otros dominios
- express.json() permite procesar datos JSON en las peticiones
- Las rutas se configuran con app.use() para cada entidad
- El servidor se inicia en el puerto 3000 por defecto

config/database.js

```
const { Sequelize } = require('sequelize');

const sequelize = new Sequelize('crud_db', 'root', '', {
    host: 'localhost',
    dialect: 'mysql',
    logging: false
});

module.exports = sequelize;
```

- Configura la conexión a la base de datos MySQL
- Usa Sequelize como ORM (Object-Relational Mapping)
- Parámetros: nombre de la base de datos ('crud_db'), usuario ('root'), contraseña ('')
- logging: false desactiva los logs de SQL en consola

modelos/articulos.js

```
const { DataTypes } = require('sequelize');
const sequelize = require('../config/database');

const Articulos = sequelize.define('Articulos', {
    id: {
        type: DataTypes.INTEGER,
        primaryKey: true,
        autoIncrement: true
    },
    nombre: {
        type: DataTypes.STRING,
        allowNull: false
    },
    descripcion: {
        type: DataTypes.TEXT
    },
    precio: {
        type: DataTypes.DECIMAL(10, 2),
        allowNull: false
    },
    stock: {
        type: DataTypes.INTEGER,
        defaultValue: 0
    }
}, {
    timestamps: true
});
```

```
module.exports = Articulos;
```

- Define la estructura de la tabla Artículos en la base de datos
- Cada campo tiene su tipo de dato y restricciones
- allowNull: false significa que el campo es obligatorio
- timestamps: true añade automáticamente campos de creación y actualización

routes/articulos.js

```
const express = require('express');
const router = express.Router();
const Articulos = require('../modelos/articulos');

// Obtener todos los artículos
router.get('/', async (req, res) => {
    try {
        const articulos = await Articulos.findAll();
        res.json(articulos);
    } catch (error) {
        res.status(500).json({ error: 'Error al obtener artículos' });
    }
});

// Crear un nuevo artículo
router.post('/', async (req, res) => {
    try {
        const nuevo = await Articulos.create(req.body);
        res.status(201).json(nuevo);
    } catch (error) {
        res.status(400).json({ error: 'Error al crear artículo' });
    }
});

// Actualizar un artículo
router.put('/:id', async (req, res) => {
    try {
        const articulo = await Articulos.findByPk(req.params.id);
        if (articulo) {
            await articulo.update(req.body);
            res.json(articulo);
        } else {
            res.status(404).json({ error: 'Artículo no encontrado' });
        }
    } catch (error) {
        res.status(400).json({ error: 'Error al actualizar artículo' });
    }
```

```
});

// Eliminar un artículo
router.delete('/:id', async (req, res) => {
    try {
        const eliminado = await Articulos.destroy({
            where: { id: req.params.id }
        });
        eliminado ?
            res.json({ mensaje: 'Artículo eliminado' }) :
            res.status(404).json({ error: 'No encontrado' });
    } catch (error) {
        res.status(500).json({ error: 'Error al eliminar artículo' });
    }
});
```

- Define las rutas para las operaciones CRUD
- GET / - Obtiene todos los artículos
- POST / - Crea un nuevo artículo
- PUT /:id - Actualiza un artículo existente
- DELETE /:id - Elimina un artículo
- Cada ruta maneja sus propios errores con try/catch
- Usa async/await para operaciones asíncronas con la base de datos

package.json

```
{
  "name": "crud-app",
  "version": "1.0.0",
  "description": "Sistema CRUD para gestión de artículos, clientes, empleados y
proveedores",
  "main": "app.js",
  "scripts": {
    "start": "node app.js",
    "dev": "nodemon app.js"
  },
  "dependencies": {
    "cors": "^2.8.5",
    "express": "^4.18.2",
    "mysql2": "^3.6.5",
    "sequelize": "^6.35.2"
  },
  "devDependencies": {
    "nodemon": "^3.0.2"
  }
}
```

- Define la configuración del proyecto
- Lista todas las dependencias necesarias
- scripts define los comandos para ejecutar el proyecto
- dependencies son las librerías necesarias para producción
- devDependencies son las librerías solo para desarrollo