



UNIVERSIDAD GALILEO  
Técnico en desarrollo de software  
Base de datos III  
Sección A

### Proyecto I

Carnet	Nombre
24001350	Dimas Andres Quintana Ramirez
17005131	Jose Carlos Perez Pamech
13005169	Byron Nefthalí amírez Rodríguez
24001219	Maria Esther Pineda Izquierdo

## **Sistema de Gestión de Inventario**

### **Descripción**

Este sistema gestiona múltiples almacenes, productos, órdenes de compra y venta, proveedores y usuarios. La base de datos está diseñada para garantizar escalabilidad, seguridad, integridad de datos, eficiencia y flexibilidad.

### **Entidades**

1. Warehouse (bodega)
2. Product (producto)
3. Inventory (inventario)
4. Suppliers (Proveedores)
5. Purchase\_order (Ordenes de compra)
6. Purchase\_order\_item (Artículos de compra)
7. Sales\_order (Ordenes de venta)
8. Sales\_order\_item (Artículos de venta)
9. User (Usuarios)

### **Características**

#### **1. Escalabilidad**

- Permite la administración de múltiples almacenes y productos.
- Uso de índices en claves primarias y foráneas para optimizar búsquedas.

#### **2. Seguridad**

- Autenticación basada en usuarios con roles para restringir accesos.

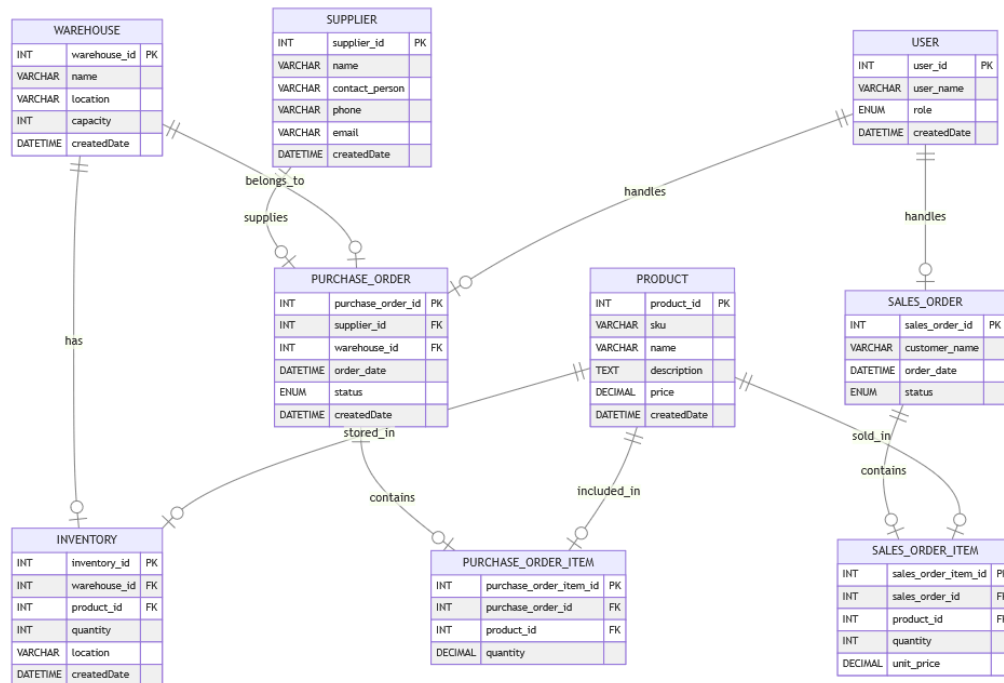
#### **3. Integridad de Datos**

- Uso de claves foráneas para garantizar consistencia en relaciones.
- Restricciones de unicidad y valores no nulos donde corresponda.

#### **4. Eficiencia**

- Uso de tipos de datos eficientes para almacenamiento y rendimiento.

# Diagrama Entidad-Relación



## CREACIÓN DE GESTIÓN DE INVENTARIO

```
CREATE DATABASE gestion_inventario;
USE gestion_inventario;

-- Tabla Bodega
CREATE TABLE warehouse (
    warehouse_id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    location VARCHAR(100) NOT NULL,
    capacity INT NOT NULL,
    createdAt DATETIME DEFAULT CURRENT_TIMESTAMP
);

-- Tabla Productos
CREATE TABLE product (
    product_id INT AUTO_INCREMENT PRIMARY KEY,
    sku VARCHAR(50) UNIQUE NOT NULL,
    name VARCHAR(100) NOT NULL,
    description TEXT NOT NULL,
    price DECIMAL(10,2) NOT NULL,
    createdAt DATETIME DEFAULT CURRENT_TIMESTAMP
);

-- Tabla Inventario
CREATE TABLE inventory (
    inventory_id INT AUTO_INCREMENT PRIMARY KEY,
    warehouse_id INT NOT NULL,
    product_id INT NOT NULL,
    quantity INT NOT NULL DEFAULT 0,
    location VARCHAR(100) NOT NULL,
    createdAt DATETIME DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (warehouse_id) REFERENCES warehouse(warehouse_id),
    FOREIGN KEY (product_id) REFERENCES product(product_id)
```

```
);
```

```
-- Tabla Proveedores
```

```
CREATE TABLE supplier (  
    supplier_id INT AUTO_INCREMENT PRIMARY KEY,  
    name VARCHAR(150),  
    contact_person VARCHAR(150),  
    phone VARCHAR(25),  
    email VARCHAR(100),  
    createdAt DATETIME DEFAULT CURRENT_TIMESTAMP  
);
```

```
-- Tabla Ordenes de Compra
```

```
CREATE TABLE purchase_order (  
    purchase_order_id INT AUTO_INCREMENT PRIMARY KEY,  
    supplier_id INT NOT NULL,  
    warehouse_id INT NOT NULL,  
    order_date DATETIME DEFAULT CURRENT_TIMESTAMP,  
    status ENUM('Pendiente', 'Recibida', 'Cancelada') DEFAULT 'Pendiente',  
    createdAt DATETIME DEFAULT CURRENT_TIMESTAMP,  
    FOREIGN KEY (supplier_id) REFERENCES supplier(supplier_id),  
    FOREIGN KEY (warehouse_id) REFERENCES warehouse(warehouse_id)  
);
```

```
-- Tabla artículos de compra
```

```
CREATE TABLE purchase_order_item (  
    purchase_order_item_id INT AUTO_INCREMENT PRIMARY KEY,  
    purchase_order_id INT NOT NULL,  
    product_id INT NOT NULL,  
    quantity DECIMAL(10,2) NOT NULL,  
    FOREIGN KEY (purchase_order_id) REFERENCES  
purchase_order(purchase_order_id),  
    FOREIGN KEY (product_id) REFERENCES product(product_id)  
);
```

```
-- Tabla Ordenes de Venta
```

```
CREATE TABLE sales_order (  
    sales_order_id INT AUTO_INCREMENT PRIMARY KEY,  
    customer_name VARCHAR(200) NOT NULL,  
    order_date DATETIME DEFAULT CURRENT_TIMESTAMP,  
    status ENUM('Pendiente', 'Enviada', 'Cancelada') DEFAULT 'Pendiente',  
    createdAt DATETIME DEFAULT CURRENT_TIMESTAMP  
);  
  
-- Tabla de Artículos de la Orden de Venta  
CREATE TABLE sales_order_item (  
    sales_order_item_id INT AUTO_INCREMENT PRIMARY KEY,  
    sales_order_id INT NOT NULL,  
    product_id INT NOT NULL,  
    quantity INT NOT NULL,  
    unit_price DECIMAL(10,2) NOT NULL,  
    FOREIGN KEY (sales_order_id) REFERENCES sales_order(sales_order_id),  
    FOREIGN KEY (product_id) REFERENCES product(product_id)  
);  
  
-- Tabla Usuarios  
CREATE TABLE user (  
    user_id INT AUTO_INCREMENT PRIMARY KEY,  
    user_name VARCHAR(50),  
    role VARCHAR(250) NOT NULL,  
    createdAt DATETIME DEFAULT CURRENT_TIMESTAMP  
);
```

## Funciones y procedimientos en MySQL

### Función para obtener el valor total de un producto (stock \* precio)

```
DELIMITER $$
```

```
CREATE FUNCTION calculate_product_value(product_id INT)
```

```
RETURNS DECIMAL(10,2)
```

```
DETERMINISTIC
```

```
BEGIN
```

```
    DECLARE total_value DECIMAL(10,2);
```

```
    DECLARE price DECIMAL(10,2);
```

```
    DECLARE stock INT;
```

```
    -- Obtener el precio del producto
```

```
    SELECT price INTO price
```

```
    FROM product
```

```
    WHERE product_id = product_id;
```

```
    -- Obtener el stock disponible del producto
```

```
    SELECT quantity INTO stock
```

```
    FROM inventory
```

```
    WHERE product_id = product_id;
```

```
    -- Calcular el valor total
```

```
    SET total_value = price * stock;
```

```
    RETURN total_value;
```

```
END $$
```

```
DELIMITER ;
```

Ejemplo:

```
SELECT calculate_product_value(1);
```



### **Procedimiento para eliminar un producto del inventario**

DELIMITER \$\$

```
CREATE PROCEDURE delete_product_from_inventory(  
    IN p_product_id INT,  
    IN p_warehouse_id INT  
)  
BEGIN  
    DELETE FROM inventory  
    WHERE product_id = p_product_id  
    AND warehouse_id = p_warehouse_id;  
END $$
```

DELIMITER ;

### **Procedimiento para agregar o actualizar el inventario**

DELIMITER \$\$

```
CREATE PROCEDURE add_or_update_inventory(  
    IN p_warehouse_id INT,  
    IN p_product_id INT,  
    IN p_quantity INT,  
    OUT p_result VARCHAR(255)  
)  
BEGIN  
    DECLARE v_existing_id INT;  
  
    -- Verifico si ya existe el producto en el almacén  
    SELECT inventory_id INTO v_existing_id  
    FROM inventory  
    WHERE product_id = p_product_id AND warehouse_id =  
p_warehouse_id;  
  
    IF v_existing_id IS NOT NULL THEN  
        -- Si ya existe, actualizo la cantidad  
        UPDATE inventory  
        SET quantity = quantity + p_quantity  
        WHERE inventory_id = v_existing_id;  
        SET p_result = 'Inventory updated successfully';
```

```

ELSE
-- Si no existe, inserto un nuevo registro
INSERT INTO inventory (product_id, warehouse_id, quantity)
VALUES (p_product_id, p_warehouse_id, p_quantity);
SET p_result = 'Item added successfully';
END IF;
END $$

DELIMITER ;

```

### Procedimiento para procesar una venta

```

DELIMITER $$

CREATE PROCEDURE process_sale(
    IN p_user_id INT,
    IN p_product_id INT,
    IN p_warehouse_id INT,
    IN p_quantity INT,
    IN p_unitary_price DECIMAL(10,2),
    OUT p_result VARCHAR(255)
)
BEGIN
    DECLARE v_available_stock INT;
    DECLARE v_total DECIMAL(10,2);

    -- Verificar stock disponible
    SELECT quantity INTO v_available_stock
    FROM inventory
    WHERE product_id = p_product_id AND warehouse_id =
p_warehouse_id;

    IF v_available_stock IS NULL THEN
        SET p_result = 'Producto no encontrado en inventario.';
    ELSEIF v_available_stock < p_quantity THEN
        SET p_result = 'Stock insuficiente.';
    ELSE
        -- Registrar la venta en sales
        INSERT INTO sales (user_id, date, total)
        VALUES (p_user_id, NOW(), 0);
    END IF;
END

```

```

SET @sale_id = LAST_INSERT_ID();

-- Calcular el total de la venta
SET v_total = p_quantity * p_unitary_price;

-- Insertar en sales_detail
INSERT INTO sales_detail (sale_id, product_id, quantity, unitary_price,
warehouse_id)
VALUES (@sale_id, p_product_id, p_quantity, p_unitary_price,
p_warehouse_id);

-- Descontar del inventario
UPDATE inventory
SET quantity = quantity - p_quantity
WHERE product_id = p_product_id AND warehouse_id =
p_warehouse_id;

-- Actualizar total en sales
UPDATE sales
SET total = v_total
WHERE sale_id = @sale_id;

SET p_result = 'Venta procesada correctamente.';
END IF;
END $$

DELIMITER ;

```

### Procedimiento para obtener inventario

DELIMITER \$\$

DROP PROCEDURE IF EXISTS get\_inventory\_report; \$\$

CREATE PROCEDURE get\_inventory\_report()

BEGIN

```
    SELECT
        p.product_id,
        p.sku,
        p.name,
        COALESCE(SUM(i.quantity), 0) AS stock,
        p.price,
        (COALESCE(SUM(i.quantity), 0) * p.price) AS total_value
    FROM product p
    LEFT JOIN inventory i ON p.product_id = i.product_id
    GROUP BY p.product_id, p.sku, p.name, p.price;
```

END \$\$

DELIMITER ;

### Procedimiento para procesar una compra

DELIMITER \$\$

DROP PROCEDURE IF EXISTS process\_purchase; \$\$

CREATE PROCEDURE process\_purchase(

```
    IN p_supplier_id INT,
    IN p_warehouse_id INT,
    IN p_product_id INT,
    IN p_quantity INT,
    IN p_unitary_price DECIMAL(10,2),
    IN p_order_date DATETIME,
    OUT p_result VARCHAR(255)
```

)

BEGIN

```
    DECLARE v_purchase_order_id INT;
    DECLARE v_existing_inventory INT;
```

```

-- Crear la orden de compra
INSERT INTO purchase_order (supplier_id, warehouse_id, order_date,
total)
VALUES (p_supplier_id, p_warehouse_id, p_order_date, 0);

SET v_purchase_order_id = LAST_INSERT_ID(); -- Obtener el ID de la
orden de compra creada

-- Insertar el detalle de compra
INSERT INTO purchase_order_item (purchase_order_id, product_id,
quantity, unitary_price)
VALUES (v_purchase_order_id, p_product_id, p_quantity,
p_unitary_price);

-- Verificar si el producto ya existe en el inventario del almacén
SELECT inventory_id INTO v_existing_inventory
FROM inventory
WHERE product_id = p_product_id AND warehouse_id =
p_warehouse_id;

IF v_existing_inventory IS NOT NULL THEN
-- Si ya existe, actualizar la cantidad sumándola
UPDATE inventory
SET quantity = quantity + p_quantity
WHERE inventory_id = v_existing_inventory;
ELSE
-- Si no existe, agregarlo al inventario
INSERT INTO inventory (product_id, warehouse_id, quantity)
VALUES (p_product_id, p_warehouse_id, p_quantity);
END IF;

-- Actualizar el total en purchase_order
UPDATE purchase_order
SET total = (p_quantity * p_unitary_price)
WHERE purchase_order_id = v_purchase_order_id;

SET p_result = 'Compra procesada correctamente.';
END $$

DELIMITER ;

```

**Trigger para actualizar el valor total en la tabla `sales_order_item` después de realizar una venta:**

DELIMITER \$\$

```
CREATE TRIGGER after_sales_order_item_insert
AFTER INSERT ON sales_order_item
FOR EACH ROW
BEGIN
    DECLARE total_value DECIMAL(10,2);

    -- Calcular el valor total (precio unitario * cantidad)
    SET total_value = NEW.unit_price * NEW.quantity;

    -- Actualizar el valor total en la orden de venta
    UPDATE sales_order
    SET total_value = total_value + total_value
    WHERE sales_order_id = NEW.sales_order_id;
END $$
```

DELIMITER ;

# Configuración de bases de datos en MongoDB

Creación de colecciones
product_changes
product_comments
transactions

Atlas

Maria's Org ...

Access Manager

Billing

All ClustersGet HelpMaria

Project 0

Data ServicesCharts

DATABASE

Clusters

SERVICES

Atlas Search

Stream Processing

Triggers

Migration

Data Federation

SECURITY

Quickstart

Backup

Database Access

Network Access

Advanced

Goto

OverviewReal TimeMetricsCollectionsAtlas SearchPerformance AdvisorOnline ArchiveCmd Line ToolsInfrastructure As Code

DATABASES: 3COLLECTIONS: 11

+ Create Database

Search Namespaces

HabitsApp

gestion\_inventario

gestion\_inventario

product\_changes

product\_comments

transactions

sample\_mflix

gestion\_inventario

LOGICAL DATA SIZE: 467BSTORAGE SIZE: 48KBINDEX SIZE: 48KBTOTAL COLLECTIONS: 4

CREATE COLLECTION

Collection Name	Documents	Logical Data Size	Avg Document Size	Storage Size	Indexes	Index Size	Avg Index Size
gestion_inventario	0	0B	0B	4KB	1	4KB	4KB
product_changes	1	168B	168B	20KB	1	20KB	20KB
product_comments	1	156B	156B	4KB	1	4KB	4KB
transactions	1	143B	143B	20KB	1	20KB	20KB

# Ejemplo de Documentos

Atlas

Maria's Org ...

Access Manager

Billing

All ClustersGet HelpMaria

Project 0

Data ServicesCharts

DATABASE

Clusters

SERVICES

Atlas Search

Stream Processing

Triggers

Migration

Data Federation

SECURITY

Quickstart

Backup

Database Access

Network Access

Advanced

Goto

OverviewReal TimeMetricsCollectionsAtlas SearchPerformance AdvisorOnline ArchiveCmd Line ToolsInfrastructure As Code

DATABASES: 3COLLECTIONS: 11

+ Create Database

Search Namespaces

HabitsApp

gestion\_inventario

gestion\_inventario

product\_changes

product\_comments

transactions

sample\_mflix

gestion\_inventario.product\_changes

STORAGE SIZE: 20KBLOGICAL DATA SIZE: 168BTOTAL DOCUMENTS: 1INDEXES TOTAL SIZE: 20KB

FindIndexesSchema Anti-PatternsAggregationSearch Indexes

Generate queries from natural language in Compass

INSERT DOCUMENT

FilterType a query: { field: 'value' }

ResetApplyOptions

QUERY RESULTS: 1-1 OF 1

```
_id: ObjectId('67c8f6dec9686c6f15fdac62')
change_id: 1
product_id: 101
change_type: "price_update"
old_value: 25
new_value: 27.5
changed_by: 3
date: "2025-01-01T12:00:00Z"
```

Atlas

Maria's Org ...

Access Manager

Billing

All ClustersGet HelpMaria

Project 0

Data ServicesCharts

DATABASE

Clusters

SERVICES

Atlas Search

Stream Processing

Triggers

Migration

Data Federation

SECURITY

Quickstart

Backup

Database Access

Network Access

Advanced

Goto

OverviewReal TimeMetricsCollectionsAtlas SearchPerformance AdvisorOnline ArchiveCmd Line ToolsInfrastructure As Code

DATABASES: 3COLLECTIONS: 11

+ Create Database

Search Namespaces

HabitsApp

gestion\_inventario

gestion\_inventario

product\_changes

product\_comments

transactions

sample\_mflix

gestion\_inventario.product\_comments

STORAGE SIZE: 20KBLOGICAL DATA SIZE: 168BTOTAL DOCUMENTS: 1INDEXES TOTAL SIZE: 20KB

FindIndexesSchema Anti-PatternsAggregationSearch Indexes

Generate queries from natural language in Compass

INSERT DOCUMENT

FilterType a query: { field: 'value' }

ResetApplyOptions

QUERY RESULTS: 1-1 OF 1

```
_id: ObjectId('67c8f701c9686c6f15fdac63')
comment_id: 1
product_id: 101
comment: "El producto se encuentra en buen estado."
operator_id: 4
date: "2025-01-01T12:00:00Z"
```

Atlas

Maria's Org ...

Access Manager

Billing

All ClustersGet HelpMaria

Project 0

Data ServicesCharts

DATABASE

Clusters

SERVICES

Atlas Search

Stream Processing

Triggers

Migration

Data Federation

SECURITY

Quickstart

Backup

Database Access

Network Access

Advanced

Goto

OverviewReal TimeMetricsCollectionsAtlas SearchPerformance AdvisorOnline ArchiveCmd Line ToolsInfrastructure As Code

DATABASES: 3COLLECTIONS: 11

+ Create Database

Search Namespaces

HabitsApp

gestion\_inventario

gestion\_inventario

product\_changes

product\_comments

transactions

sample\_mflix

gestion\_inventario.transactions

STORAGE SIZE: 20KBLOGICAL DATA SIZE: 143BTOTAL DOCUMENTS: 1INDEXES TOTAL SIZE: 20KB

FindIndexesSchema Anti-PatternsAggregationSearch Indexes

Generate queries from natural language in Compass

INSERT DOCUMENT

FilterType a query: { field: 'value' }

ResetApplyOptions

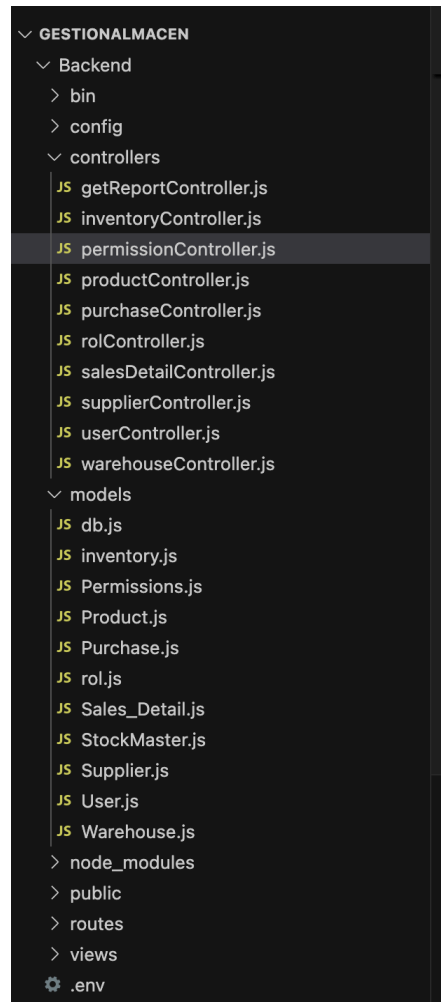
QUERY RESULTS: 1-1 OF 1

```
_id: ObjectId('67c8f69ec9686c6f15fdac5b')
transaction_id: 1
product_id: 101
quantity: 10
transaction_type: "sale"
date: "2025-01-01T12:00:00Z"
user_id: 2
```



## Backend

El Backend se realizó con Node.js y Express, utilizando Modelos y Controladores, para la creación de los endpoints.



Pruebas de conexión:

```
54 | | |
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS POSTMAN CO
● → gestionAlmacen cd Backend
○ → Backend npm start

> stockmaster@0.0.0 start
> node ./bin/www

Conexión a MySQL establecida correctamente.
Conectado a MongoDB
GET /getProducts 400 3.142 ms - 33
GET /getProducts 404 33.920 ms - 28
GET /getProducts 200 25.259 ms - 1183
POST /createProduct 400 0.778 ms - 33
POST /createProduct 201 31.275 ms - 179
GET /getProducts 200 8.614 ms - 1310
GET /getRoles 200 18.245 ms - 51
```

## Pruebas de endpoints

Las pruebas se realizaron con Postman, incluyendo perfiles de seguridad y permisos en cada petición.

```
//NOTE - Endpoints de productos
router.post("/createProduct", checkPermission("GESTIONAR_PRODUCTOS"), createProduct);
router.get("/getProducts", checkPermission("GESTIONAR_PRODUCTOS"), getProducts);
router.delete("/deleteProduct/:product_id", checkPermission("GESTIONAR_PRODUCTOS"), deleteProducts);

//NOTE - Endpoints de almacen
router.post("/createWarehouse", checkPermission("GESTIONAR_ALMACENES"), createWarehouse);
router.get("/getWarehouse", checkPermission("GESTIONAR_ALMACENES"), getWarehouse);
router.delete("/deleteWarehouse/:warehouse_id", checkPermission("GESTIONAR_ALMACENES"), deleteWarehouse);

//NOTE - Endpoints de inventario
router.post("/addInventoryItem", checkPermission("GESTIONAR_INVENTARIO"), addInventoryItem);
router.get("/getInventory", checkPermission("GESTIONAR_INVENTARIO"), getInventory);
router.delete("/deleteInventoryItem/:product_id/:warehouse_id", checkPermission("GESTIONAR_INVENTARIO"), deleteInventoryItem);
```

```
const checkPermission = (requiredAction) => {
  return async (req, res, next) => {
    try {
      const { user_id } = req.body;

      if (!user_id) {
        return res.status(400).json({ message: "User ID is required" });
      }

      const [user] = await sequelize.query(
        "SELECT rol_id FROM user WHERE user_id = ?",
        { replacements: [user_id], type: sequelize.QueryTypes.SELECT }
      );

      if (!user) {
        return res.status(404).json({ message: "User not found" });
      }

      const [permission] = await sequelize.query(
        "SELECT accion FROM permissions WHERE rol_id = ? AND accion = ?",
        { replacements: [user.rol_id, requiredAction], type: sequelize.QueryTypes.SELECT }
      );

      if (!permission) {
        return res.status(403).json({ message: "No tienes permisos para esta acción" });
      }

      next();
    } catch (error) {
      console.error("Error en la validación de permisos:", error);
      return res.status(500).json({ message: "Error en el servidor" });
    }
  };
};
```

## Permisos, Roles y Usuarios

Se crearon 4 tipos de Roles:

```
"name": "UA" //Administrador
"name": "UO" //Operador de inventarios
"name": "UV" //Ventas
"name": "UAD" //Auditor
```

A cada uno de los Roles se les asignan distintos permisos:

```
"rol_id": 1,
"accion": [
  "GESTIONAR_USUARIOS",
```

```
"GESTIONAR_PRODUCTOS",  
"GESTIONAR_VENTAS",  
"GESTIONAR_ALMACENES",  
"GESTIONAR_INVENTARIO",  
"VER_REPORTES"  
]
```

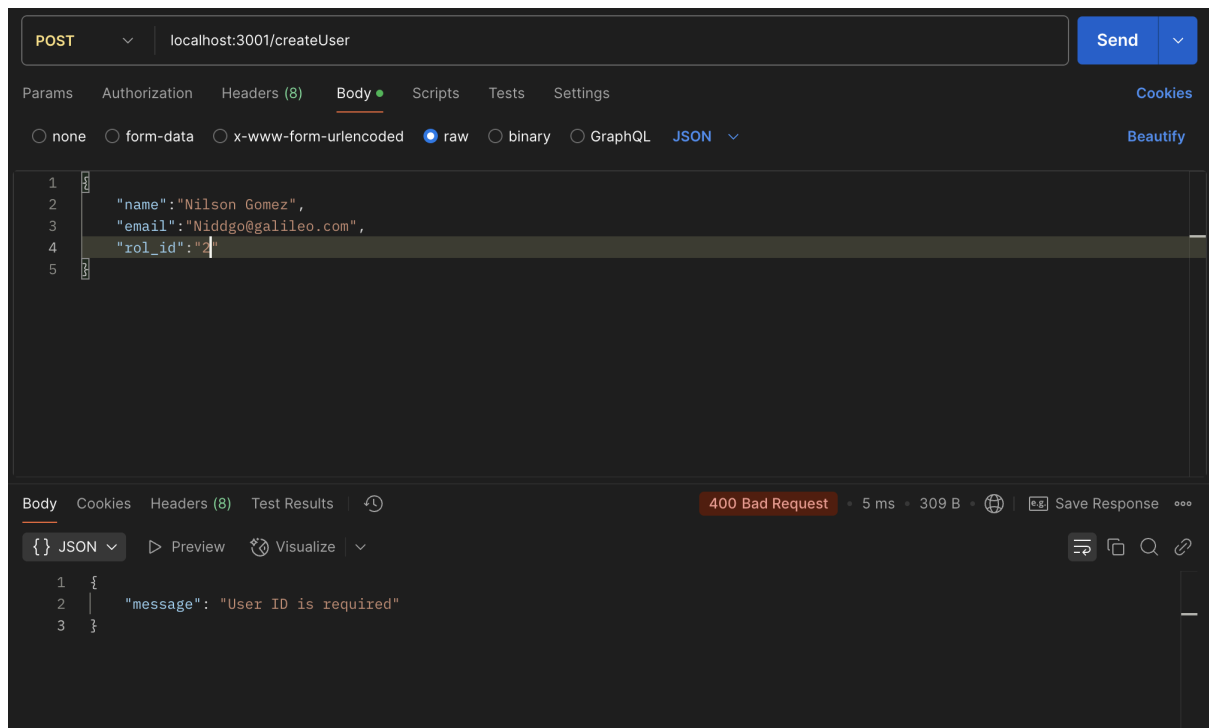
```
"rol_id": 2,  
"accion": [  
  "GESTIONAR_PRODUCTOS",  
  "GESTIONAR_ALMACENES",  
  "GESTIONAR_INVENTARIO"  
]
```

```
"rol_id": 3,  
"accion": [  
  "GESTIONAR_VENTAS"  
]
```

```
"rol_id": 4,  
"accion": [  
  "VER_REPORTES"  
]
```

Limitando de esta manera los distintos permisos que puede realizar cada uno de los usuarios, por ejemplo, si un usuario tiene Rol 4, este no podrá gestionar usuarios,, productos, ventas, etc.

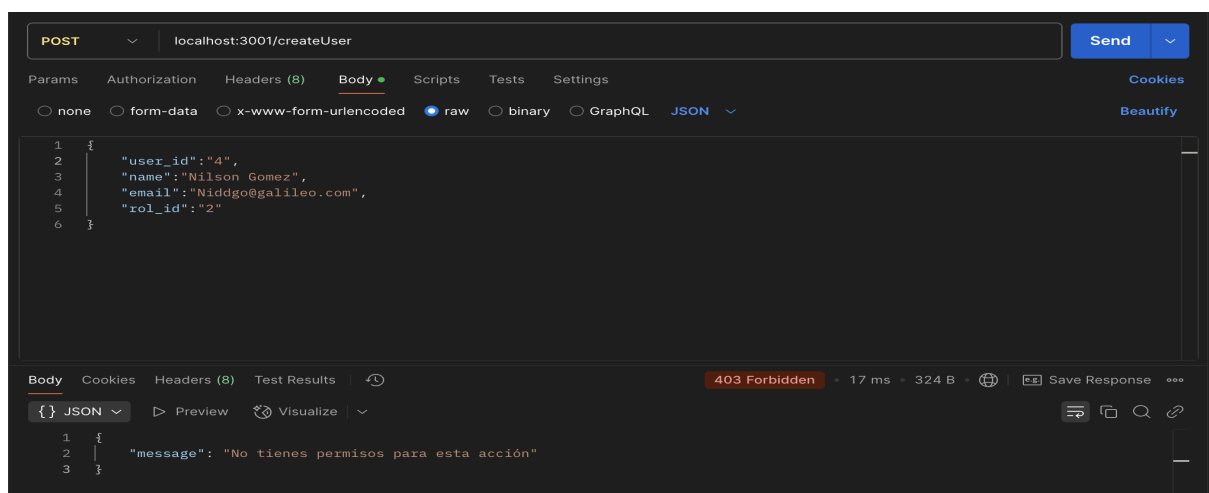
Ejemplo:



En este caso, se intenta crear un usuario con Rol 2 (UV), sin embargo no se está proporcionando el user\_id de quien está creando la solicitud, por lo que procedemos a incluirlo en la solicitud.

Se utiliza el user\_id:4, que tiene la siguiente info:

```
{
  "user_id": 4,
  "name": "Nery Reyes",
  "email": "Guares@galileo.com",
  "rol_id": 2
}
```



al ser un usuario con rol\_id 4 (UAD/Usuario administrador), no tiene permisos suficientes, por lo que procedemos a hacer la solicitud con el siguiente usuario, con rol:1 (UA):

```
{
```

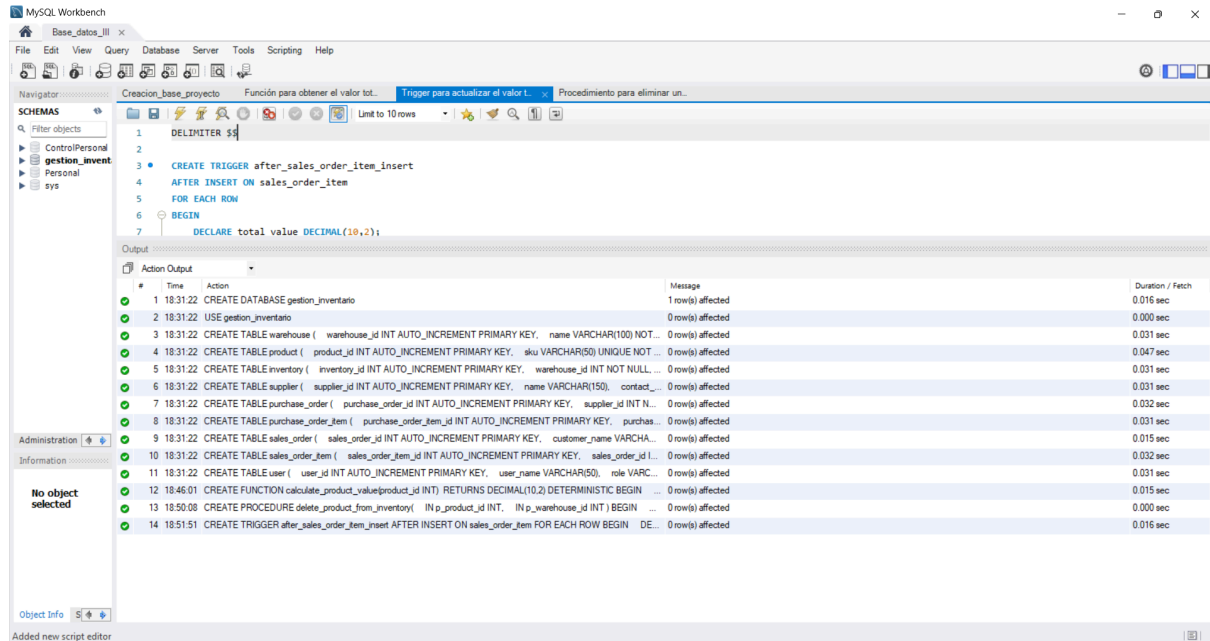
```
"user_id": 2,  
"name": "Byron Ramírez",  
"email": "bramirez@galileo.com",  
"rol_id": 1  
},
```

The screenshot displays a REST client interface with a POST request to `localhost:3001/createUser`. The request body is a JSON object with the following fields: `user_id` (2), `name` (Nilson Gomez), `email` (Niddgo@galileo.com), and `rol_id` (2). The response is a 201 status code, indicating successful creation, with a response time of 41 ms and a body size of 397 B. The response body is a JSON object containing a success message and the created user details.

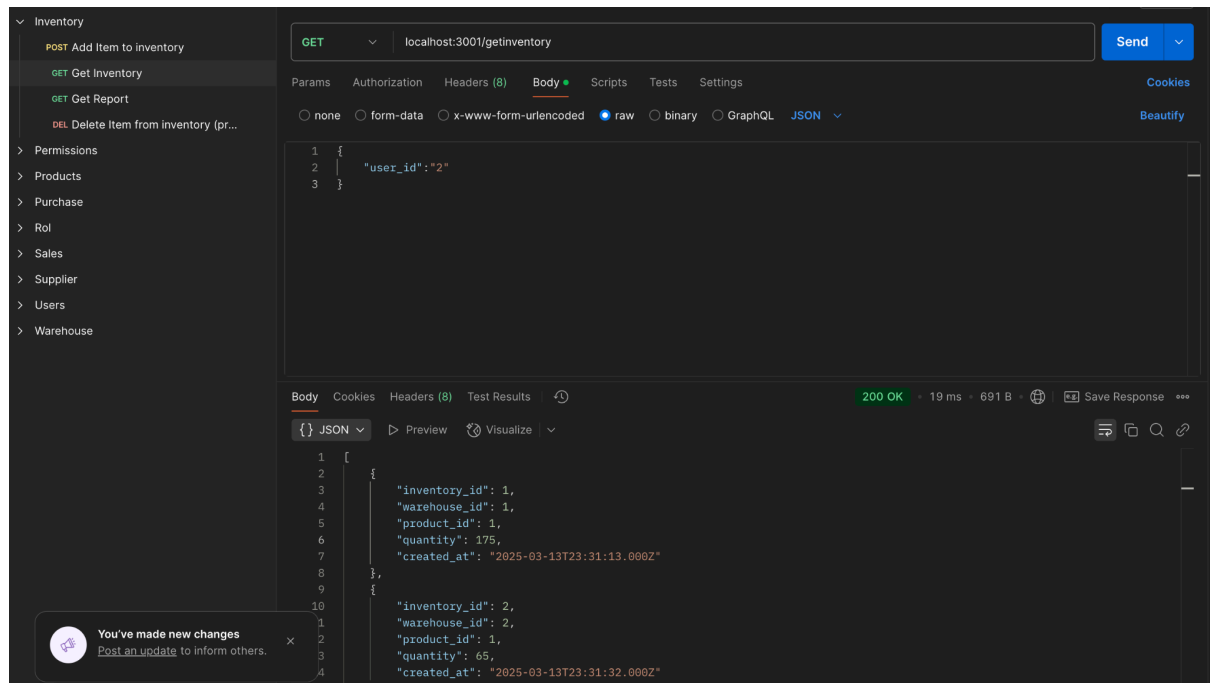
```
POST localhost:3001/createUser  
Send  
Params Authorization Headers (8) Body Scripts Tests Settings Cookies  
none form-data x-www-form-urlencoded raw binary GraphQL JSON  
Beautify  
1 {  
2   "user_id": "2",  
3   "name": "Nilson Gomez",  
4   "email": "Niddgo@galileo.com",  
5   "rol_id": "2"  
6 }  
Body Cookies Headers (8) Test Results  
201 Created 41 ms 397 B Save Response  
JSON Preview Visualize  
1 {  
2   "message": "User created successfully",  
3   "user": {  
4     "user_id": 5,  
5     "name": "Nilson Gomez",  
6     "email": "Niddgo@galileo.com",  
7     "rol_id": "2"  
8   }  
9 }
```

Acá vemos el funcionamiento puntual, de los permisos y roles.

# Apéndice



Ejemplos con postman:



+

☰

...

> Inventory

> Permissions

> POST Create Permissions

> Products

> Purchase

> Rol

> Sales

> Supplier

> Users

> Warehouse

Permissions / Create Permissions

POSTlocalhost:3001/createPermission

ParamsAuthorizationHeaders (8)BodyScriptsTestsSettings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

// {

// "rol\_id": 2,

// "accion": [

// "GESTIONAR\_PRODUCTOS",

// "GESTIONAR\_ALMACENES",

// "GESTIONAR\_INVENTARIO"

// ]

// }

{

"rol\_id": 3,

"accion": [

"GESTIONAR\_VENTAS"

]

}

// {

// "rol\_id": 4,

// "accion": [

// "VER\_REPORTES"

// ]

// }

BodyCookiesHeaders (8)Test Results

201 Created18 ms

{}

JSON

Preview

Visualize

1

2

3

{

"message": "Permisos agregados exitosamente"

}

You've made new changes

+

☰

...

> Inventory

> Permissions

> Products

> GET Get Products

> POST Create Product

> DEL Delete Product

> Purchase

> Rol

> Sales

> Supplier

> Users

> Warehouse

Products / Get Products

GETlocalhost:3001/getProducts

ParamsAuthorizationHeaders (8)BodyScriptsTestsSettings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

112

113

114

115

116

117

118

119

120

121

122

123

124

125

126

127

128

129

130

131

132

133

134

135

136

137

138

139

140

141

142

143

144

145

146

147

148

149

150

151

152

153

154

155

156

157

158

159

160

161

162

163

164

165

166

167

168

169

170

171

172

173

174

175

176

177

178

179

180

181

182

183

184

185

186

187

188

189

190

191

192

193

194

195

196

197

198

199

200

201

202

203

204

205

206

207

208

209

210

211

212

213

214

215

216

217

218

219

220

221

222

223

224

225

226

227

228

229

230

231

232

233

234

235

236

237

238

239

240

241

242

243

244

245

246

247

248

249

250

251

252

253

254

255

256

257

258

259

260

261

262

263

264

265

266

267

268

269

270

271

272

273

274

275

276

277

278

279

280

281

282

283

284

285

286

287

288

289

290

291

292

293

294

295

296

297

298

299

300

301

302

303

304

305

306

307

308

309

310

311

312

313

314

315

316

317

318

319

320

321

322

323

324

325

326

327

328

329

330

331

332

333

334

335

336

337

338

339

340

341

342

343

344

345

346

347

348

349

350

351

352

353

354

355

356

357

358

359

360

361

362

363

364

365

366

367

368

369

370

371

372

373

374

375

376

377

378

379

380

381

382

383

384

385

386

387

388

389

390

391

392

393

394

395

396

397

398

399

400

401

402

403

404

405

406

407

408

409

410

411

412

413

414

415

416

417

418

419

420

421

422

423

424

425

426

427

428

429

430

431

432

433

434

435

436

437

438

439

440

441

442

443

444

445

446

447

448

449

450

451

452

453

454

455

456

457

458

459

460

461

462

463

464

465

466

467

468

469

470

471

472

473

474

475

476

477

478

479

480

481

482

483

484

485

486

487

488

489

490

491

492

493

494

495

496

497

498

499

500

501

502

503

504

505

506

507

508

509

510

511

512

513

514

515

516

517

518

519

520

521

522

523

524

525

526

527

528

529

530

531

532

533

534

535

536

537

538

539

540

541

542

543

544

545

546

547

548

549

550

551

552

553

554

555

556

557

558

559

560

561

562

563

564

565

566

567

568

569

570

571

572

573

574

575

576

577

578

579

580

581

582

583

584

585

586

587

588

589

590

591

592

593

594

595

596

597

598

599

600

601

602

603

604

605

606

607

608

609

610

611

612

613

614

615

616

617

618

619

620

621

622

623

624

625

626

627

628

629

630

631

632

633

634

635

636

637

638

639

640

641

642

643

644

645

646

647

648

649

650

651

652

653

654

655

656

657

658

659

660

661

662

663

664

665

666

667

668

669

670

671

672

673

674

675

676

677

678

679

680

681

682

683

684

685

686

687

688

689

690

691

692

693

694

695

696

697

698

699

700

701

702

703

704

705

706

707

708

709

710

711

712

713

714

715

716

717

718

719

720

721

722

723

724

725

726

727

728

729

730

731

732

733

734

735

736

737

738

739

740

741

742

743

744

745

746

747

748

749

750

751

752

753

754

755

756

757

758

759

760

761

762

763

764

765

766

767

768

769

770

771

772

773

774

775

776

777

778

779

780

781

782

783

784

785

786

787

788

789

790

791

792

793

794

795

796

797

798

799

800

801

802

803

804

805

806

807

808

809

810

811

812

813

814

815

816

817

818

819

820

821

822

823

824

825

826

827

828

829

830

831

832

833

834

835

836

837

838

839

840

841

842

843

844

845

846

847

848

849

850

851

852

853

854

855

856

857

858

859

860

861

862

863

864

865

866

867

868

869

870

871

872

873

874

875

876

877

878

879

880

881

882

883

884

885

886

887

888

889

890

891

892

893

894

895

896

897

898

899

900

901

902

903

904

905

906

907

908

909

910

911

912

913

914

915

916

917

918

919

920

921

922

923

924

925

926

927

928

929

930

931

932

933

934

935

936

937

938

939

940

941

942

943

944

945

946

947

948

949

950

951

952

953

954

955

956

957

958

959

960

961

962

963

964

965

966

967

968

969

970

971

972

973

974

975

976

977

978

979

980

981

982

983

984

985

986

987

988

989

990

991

992

993

994

995

996

997

998

999

1000

BodyCookiesHeaders (8)Test Results

200 OK12 ms1.54 KB

{}

JSON

Preview

Visualize

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

112

113

114

115

116

117

118

119

120

121

122

123

124

125

126

127

128

129

130

131

132

133

134

135

136

137

138

139

140

141

142

143

144

145

146

147

148

149

150

151

152

153

154

155

156

157

158

159

160

161

162

163

164

165

166

167

168

169

170

171

172

173

174

175

176

177

178

179

180

181

182

183

184

185

186

187

188

189

190

191

192

193

194

195

196

197

198

199

200

201

202

203

204

205

206

207

208

209

210

211

212

213

214

215

216

217

218

219

220

221

222

223

224

225

226

227

228

229

230

231

232

233

234

235

236

237

238

239

240

241

242

243

244

245

246

247

248

249

250

251

252

253

254

255

256

257

258

259

260

261

262

263

264

265

266

267

268

269

270

271

272

273

274

275

276

277

278

279

280

281

282

283

284

285

286

287

288

289

290

291

292

293

294

295

296

297

298

299

300

301

302

303

304

305

306

307

308

309

310

311

312

313

314

315

316

317

318

319

320

321

322

323

324

325

326

327

328

329

330

331

332

333

334

335

336

337

338

339

340

341

342

343

344

345

346

347

348

349

350

351

352

353

354

355

356

357

358

359

360

361

362

363

364

365

366

367

368

369

370

371

372

373

374

375

376

377

378

379

380

381

382

383

384

385

386

387

388

389

390

391

392

393

394

395

396

397

398

399

400

401

402

403

404

405

406

407

408

409

410

411

412

413

414

415

416

417

418

419

420

421

422

423

424

425

426

427

428

429

430

431

432

433

434

435

436

437

438

439

440

441

442

443

444

445

446

447

448

449

