Manual Técnico - API Mechanical Workshop

1. Información General del Proyecto

1.1 Descripción

Sistema integral para la gestión de talleres mecánicos que incluye administración de repuestos, usuarios, clientes y seguimiento de trabajos de mantenimiento vehicular.

1.2 Tecnologías Utilizadas

Backend

- **Framework:** Spring Boot 3.5.4
- **Java:** Versión 21
- **Base de datos:** MySQL
- **ORM:** JPA/Hibernate
- **Seguridad:** Spring Security 3.5.4
- **Validación:** Spring Boot Validation
- **Herramientas:** Lombok, Spring DevTools
- **Empaquetado:** WAR

Frontend

- **Framework:** Angular 20.1.0
- **UI Components:** Angular Material 20.2.0
- **Styling:** Tailwind CSS 4.1.12
- **Animaciones:** Angular Animations 20.2.1
- **Herramientas:** TypeScript 5.8.2, RxJS 7.8.0

Endpoints

Modulo administrador

URL_PARTS:

http://localhost:8081/parts

Obtener todos los repuestos

- **URL**: /all
- **Método**: GET
- **Descripción**: Retorna la lista completa de repuestos registrados.
- **Parámetros**: Ninguno.


```
[
  {
    "partId": 1,
    "supplierId": 10,
    "namePart": "Filtro de aceite",
    "brandPart": "Bosch",
    "descriptionPart": "Filtro para motor gasolina",
    "costPrice": 50.0,
    "salePrice": 75.0,
    "stockPart": 20,
    "active": true
  },
  ...
]
```


Obtener repuestos por estado (activo/inactivo)

- **URL:** /active/{active}
- **Método:** GET
- **Descripción:** Retorna todos los repuestos que coincidan con el estado de actividad.
- Parámetros de ruta:
  - active *(boolean)* → true para activos, false para inactivos.
- Respuesta (200 OK):


```
[
  {
```

```
    "partId": 2,
    "supplierId": 11,
    "namePart": "Pastillas de freno",
    "brandPart": "Brembo",
    "descriptionPart": "Juego de pastillas delanteras",
    "costPrice": 200.0,
    "salePrice": 280.0,
    "stockPart": 15,
    "active": true
  }
]
```


Crear un nuevo repuesto

- URL: /createPart
- Método: POST
- Descripción: Crea un nuevo repuesto en el sistema.
- Body (JSON):


```
{
  "supplierId": 10,
  "namePart": "Filtro de aire",
  "brandPart": "Mann",
  "descriptionPart": "Filtro de aire para motor diésel",
  "costPrice": 120.0,
  "salePrice": 180.0,
  "stockPart": 30,
  "active": true
}
```


Actualizar un repuesto (con gestión de stock)

- URL: /updatePart/{addStock}
- Método: PUT
- Descripción: Actualiza los datos de un repuesto. El parámetro addStock indica si se debe sumar stock (true) o restar stock (false) al valor existente.

- Parámetros de ruta:
    - addStock (boolean) → true = sumar stock, false = restar stock.
- Body (JSON):

```
{
 "partId": 5,
 "supplierId": 10,
 "namePart": "Filtro de aire",
 "brandPart": "Mann",
 "descriptionPart": "Filtro de aire para motor diésel",
 "costPrice": 125.0,
 "salePrice": 190.0,
 "stockPart": 10,
 "active": true
}
```

URL_USERS

http://localhost:8081/users

Crear un usuario (proveedor / empleado)

- **URL:** /createUser/{typeSupplier}
- **Método:** POST
- **Descripción:** Crea un nuevo usuario en el sistema. El parámetro typeSupplier indica el tipo de proveedor (puede usarse para distinguir categorías de usuario).
- **Parámetros de ruta:**
    - typeSupplier *(String)* → Tipo de proveedor: “company” , “person”
- **Body (JSON):**

```
{
 "rolId": 2,
 "username": "juanperez",
 "email": "juan.perez@example.com",
```

```
  "phone": "50255551234",
  "name": "Juan",
  "lastName": "Pérez",
  "twoFactorAuth": false,
  "typeTwoFactorId": <ID>
}
```

Actualizar usuario

- **URL:** /updateUser
- **Método:** PUT
- **Descripción:** Actualiza los datos de un usuario existente.
- **Body (JSON):**

```
{
  "userId": 1,
  "rolId": 2,
  "username": "juanperez",
  "active": true,
  "email": "juan.perez@example.com",
  "phone": "50255556789",
  "name": "Juan Carlos",
  "lastName": "Pérez",
  "twoFactorAuth": true,
  "typeTwoFactorId": 1
}
```

Crear un cliente

- **URL:** /createClient
- **Método:** POST
- **Descripción:** Registra un nuevo cliente en el sistema.
- **Body (JSON):**

```
{
```

```
  "username": "mariagarcia",
  "email": "maria.garcia@example.com",
  "phone": "50255559876",
  "name": "María",
  "lastName": "García"
}
```

- **Respuesta (201 Created):**

```
{
  "userId": 2,
  "rolId": 3,
  "username": "mariagarcia",
  "active": true,
  "email": "maria.garcia@example.com",
  "phone": "50255559876",
  "name": "María",
  "lastName": "García",
  "twoFactorAuth": false,
  "typeTwoFactorId": <ID>
}
```

Obtener todos los usuarios

- **URL:** /all
- **Método:** GET
- **Descripción:** Retorna la lista de todos los usuarios registrados.
- **Respuesta (200 OK):**

```
[
  {
    "userId": 1,
    "rolId": 2,
    "username": "juanperez",
    "active": true,
    "email": "juan.perez@example.com",
```

```
    "phone": "50255556789",
    "name": "Juan Carlos",
    "lastName": "Pérez",
    "twoFactorAuth": true,
    "typeTwoFactorId": 1
  },
  {
    "userId": 2,
    "rolId": 3,
    "username": "mariagarcia",
    "active": true,
    "email": "maria.garcia@example.com",
    "phone": "50255559876",
    "name": "María",
    "lastName": "García",
    "twoFactorAuth": false,
    "typeTwoFactorId": null
  }, …
]
```


Obtener usuarios por estado

- **URL:** /isActive/{active}
- **Método:** GET
- **Descripción:** Retorna los usuarios que coincidan con el estado de actividad.
- **Parámetros de ruta:**
    - active *(boolean)* → true para activos, false para inactivos.
- **Respuesta (200 OK):**


```
[
  {
    "userId": 1,
    "rolId": 2,
    "username": "juanperez",
    "active": true,
    "email": "juan.perez@example.com",
    "phone": "50255556789",
```

```
    "name": "Juan Carlos",
    "lastName": "Pérez",
    "twoFactorAuth": true,
    "typeTwoFactorId": 1
  }
]
```


Login de usuario

- **URL:** /login
- **Método:** POST
- **Descripción:** Permite que un usuario inicie sesión en el sistema.
- **Body (JSON):**


```
{
  "username": "juanperez",
  "password": "123456"
}
```


- **Respuesta (200 OK):**


```
{
  "userId": 1,
  "rolId": 2,
  "username": "juanperez",
  "active": true,
  "email": "juan.perez@example.com",
  "phone": "50255556789",
  "name": "Juan Carlos",
  "lastName": "Pérez",
  "twoFactorAuth": true,
  "typeTwoFactorId": 1
}
```

Modulo empleado

1. Dashboard del Empleado

   URL: /employee/dashboard

   Método: GET

   Descripción: Endpoint de bienvenida para el dashboard del empleado

   Respuesta: (200 OK)


```
{
 "message": "Welcome to employe dashboard",
 "status": "success",
 "rol": "EMPLEADO"
}
```

2. Obtener Trabajos asignados

URL: /employee/jobs/{empleadoId}

Método: GET

Descripción: Retorna todos los trabajos asignados a un empleado específico

Respuesta: (200 OK)

Parametros: empleadoId: ID del empleado


```
{
 "mensaje": "Trabajos asignados obtenidos correctamente",
 "empleadoId": 2,
 "trabajos": [
  {
   "trabajoId": 1,
   "vehiculoId": 1,
   "placaVehiculo": "P123ABC",
   "marcaVehiculo": "Toyota",
   "modeloVehiculo": "Corolla",
   "tipoMantenimiento": "Preventivo",
   "estadoTrabajo": "En progreso",
   "descripcion": "Cambio de aceite y filtros",
   "horasEstimadas": 2.5,
   "fechaAsignacion": "2025-08-20T09:00:00",
   "clienteNombre": "María López"
  }
 ],
 "total": 1,
 "status": "success"
}
```

3. Iniciar un Trabajo

URL: /employee/job/{trabajoId}/iniciar

Método: PUT

Descripción: Cambia el estado de un trabajo de "Pendiente" a "En progreso"

Parametros: trabajoId: ID del trabajo

Body:

```
{
 "empleadoId": 2
}
```

Respuesta:

```
{
 "mensaje": "Trabajo iniciado correctamente",
 "trabajoId": 1,
 "empleadoId": 2,
 "status": "success"
}
```

4. Finalizar un Trabajo

URL: /employee/job/{trabajoId}/finalizar

Método: PUT

Descripción: Marca un trabajo como completado

Parametros: trabajoId: ID del trabajo

Body:

```
{
 "empleadoId": 2,
 "observaciones": "Trabajo completado satisfactoriamente"
}
```

Respuesta: 200 OK

5. Perfil del empleado

   URL: /employee/profile

   Método: GET

   Descripción: Informacion del Perfil del Empleado

   Respuesta: 200 OK


```
{
 "message": "Welcome to employe profile",
 "name": "Test employee",
 "role": "EMPLEADO",
 "status": "success"
}
```


6. Verificacion de sesion

   URL: /employee/verification

   Método: GET

   Descripción: Verifica el estado de autenticación del empleado

   Respuesta: 200 OK


```
{
 "message": "Session employe verification",
 "role": "EMPLEADO",
 "status": "Authenticated"
}
```

7. Detalles del trabajo

URL: /employee/job/{trabajoId}/details/{empleadoId}

Método: GET

Descripción: Obtiene información detallada de un trabajo específico

Parametros:

- trabajoId
- empleadoId

Respuesta: 200 OK

8. Registrar avance

URL: /employee/job/{trabajoId}/avance

Método: POST

Descripción: Registra el progreso/avance en un trabajo

Parametros:

- trabajoId

Body:

```
{
 "empleadoId": 2,
 "descripcion": "Se completó el cambio de aceite",
 "horasTrabajadas": 1.5,
 "tipo": "AVANCE"
}
```

9. Historial de trabajo

URL: /employee/job/{trabajoId}/historial

Método: GET

Descripción: Obtiene el historial de actualizaciones de un trabajo

Parametros:

- trabajoId

10. Solicitar apoyo del especialista

URL: /employee/job/{trabajoId}/solicitar-apoyo

Método: POST

Descripción: Solicita ayuda de un especialista para un trabajo

Parametros:

- trabajoId

```
{
 "empleadoId": 2,
 "motivo": "Problema con la transmisión",
 "tipoEspecialista": "Transmisiones",
 "especialistaId": 3
}
```

11. Reportar da;os

URL: /employee/job/{trabajoId}/reportar-danos

Método: POST

Descripción: Reporta daños adicionales encontrados durante el trabajo

Parametros:

- trabajoId

```
{
 "empleadoId": 2,
 "descripcionDanos": "Se encontró daño en el radiador",
 "severidad": "ALTO"
}
```

12. Usar repuesto

URL: /employee/job/{trabajoId}/usar-repuesto

Método: POST

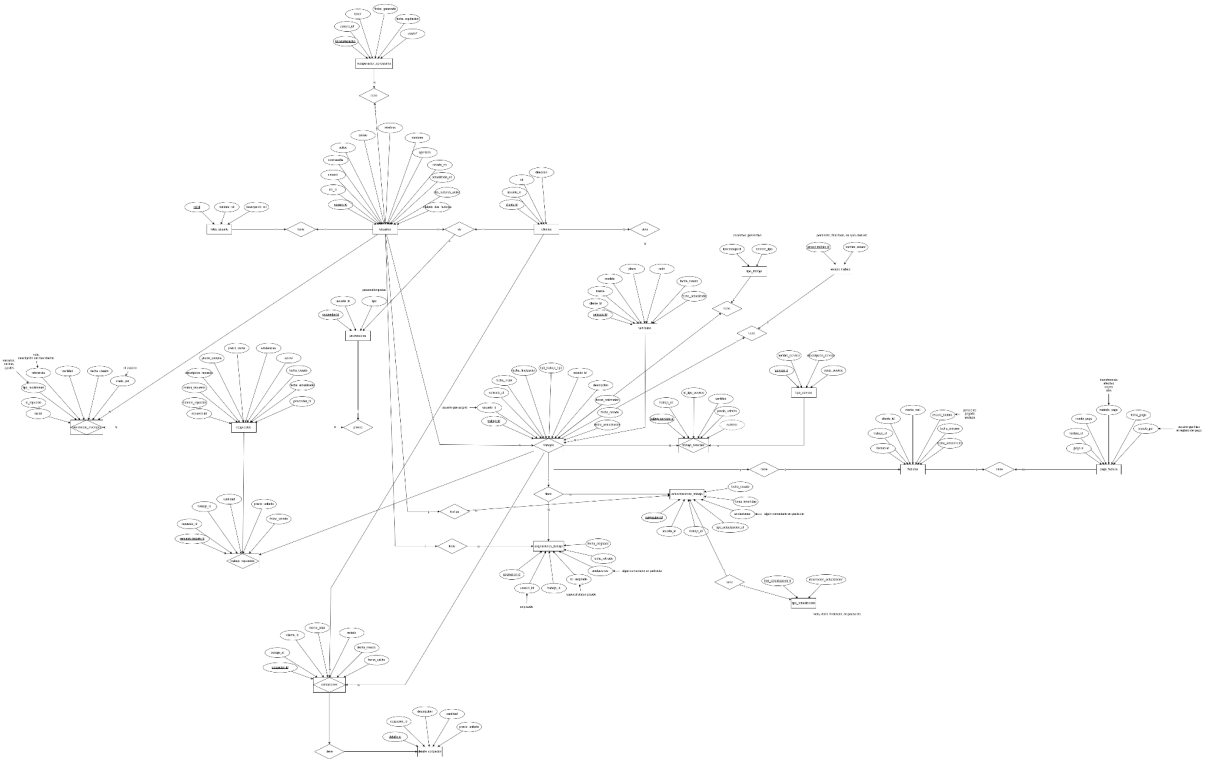
Descripción: Registra el uso de un repuesto en un trabajo

Parametros:

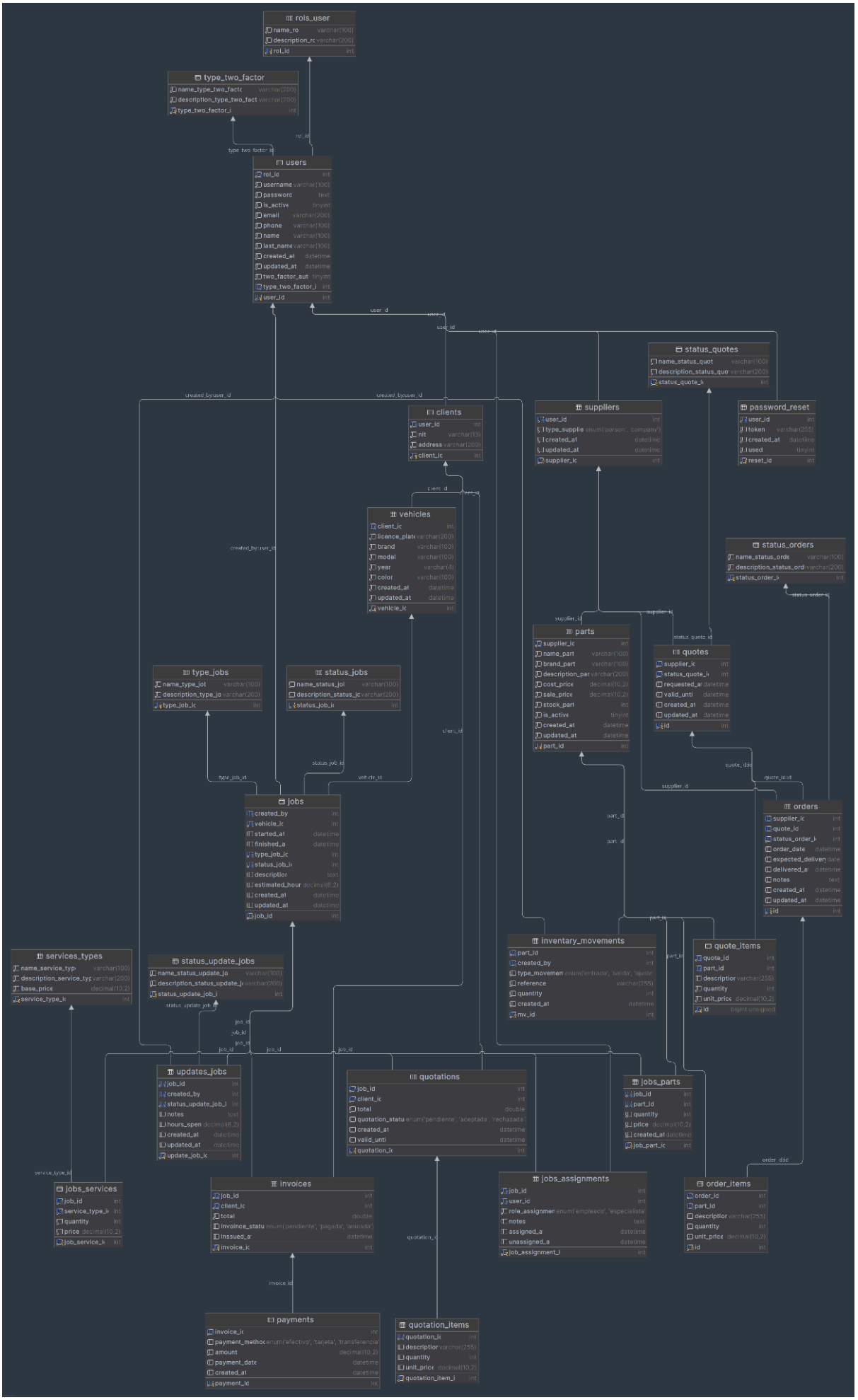
- trabajoId

```
{
 "empleadoId": 2,
 "repuestoId": 1,
 "cantidad": 2,
 "observaciones": "Filtros de aceite utilizados"
}
```

Diagrama entidad relación



Diagramas de tablas


rols_user
name_ro varchar(100)
description_rc varchar(200)
rol_id int
type_two_factor
name_type_two_factc varchar(200)
description_type_two_fact varchar(200)
type_two_factor_i int
rol_id
type_two_factor_id
users
rol_id int
username varchar(100)
password text
is_active tinyint
email varchar(200)
phone varchar(100)
name varchar(100)
last_name varchar(100)
created_at datetime
updated_at datetime
two_factor_aut tinyint
type_two_factor_i int
user_id int
user_id
user_id
user_id
status_quotes
name_status_quot varchar(100)
description_status_quo varchar(200)
status_quote_i int
created_byuser_id
created_byuser_id
clients
user_id int
nit varchar(13)
address varchar(200)
client_ic int
suppliers
user_id int
type_supplie enum('person', 'company')
created_at datetime
updated_at datetime
supplier_ic int
password_reset
user_id int
token varchar(255)
created_at datetime
used tinyint
reset_id int
client_id
user_id
vehicles
client_ic int
licence_plate varchar(200)
brand varchar(100)
model varchar(100)
year varchar(4)
color varchar(100)
created_at datetime
updated_at datetime
vehicle_ic int
created_byuser_id
status_orders
name_status_orde varchar(100)
description_status_ord varchar(200)
status_order_ic int
status_order_id
supplier_id
supplier_id
parts
supplier_ic int
name_part varchar(100)
brand_part varchar(100)
description_par varchar(200)
cost_price decimal(10,2)
sale_price decimal(10,2)
stock_part int
is_active tinyint
created_at datetime
updated_at datetime
part_id int
status_quote_id
quotes
supplier_ic int
status_quote_ic int
requested_at datetime
valid_unti datetime
created_at datetime
updated_at datetime
id int
type_jobs
name_type_job varchar(100)
description_type_jo varchar(200)
type_job_ic int
status_jobs
name_status_job varchar(100)
description_status_jc varchar(200)
status_job_ic int
client_id
status_job_id
type_job_id
vehicle_id
quote_dat
quote_id id
supplier_id
jobs
created_by int
vehicle_ic int
started_at datetime
finished_a datetime
type_job_ic int
status_job_ic int
description text
estimated_hour decimal(6,2)
created_at datetime
updated_at datetime
job_id int
part_id
part_id
orders
supplier_ic int
quote_id int
status_order_ic int
order_date datetime
expected_deliver date
delivered_at datetime
notes text
created_at datetime
updated_at datetime
id int
part_id
services_types
name_service_type varchar(100)
description_service_ty varchar(200)
base_price decimal(10,2)
service_type_ic int
status_update_jobs
name_status_update_jo varchar(100)
description_status_update_j varchar(200)
status_update_job_i int
status_update_job_id
job_id
job_id
job_id
job_id
job_id
inventary_movements
part_id int
created_by int
type_movemen enum('entrada', 'salida', 'ajuste'
reference varchar(255)
quantity int
created_at datetime
mv_id int
part_id
quote_items
quote_id int
part_id int
descriptior varchar(255)
quantity int
unit_price decimal(10,2)
id bigint unsigned
updates_jobs
job_id int
created_by int
status_update_job_i int
notes text
hours_spen decimal(6,2)
created_at datetime
updated_at datetime
update_job_ic int
quotations
job_id int
client_ic int
total double
quotation_statu enum('pendiente', 'aceptada', 'rechazada')
created_at datetime
valid_unti datetime
quotation_ic int
jobs_parts
job_id int
part_id int
quantity int
price decimal(10,2)
created_at datetime
job_part_ic int
service_type_id
order_dat
jobs_services
job_id int
service_type_i int
quantity int
price decimal(10,2)
job_service_i int
invoices
job_id int
client_ic int
total double
invoince_statu enum('pendiente', 'pagada', 'anulada')
issued_at datetime
invoice_ic int
jobs_assignments
job_id int
user_ic int
role_assignmen enum('empleado', 'especialista'
notes text
assigned_a datetime
unassigned_a datetime
job_assignment_i int
order_items
order_ic int
part_id int
descriptior varchar(255)
quantity int
unit_price decimal(10,2)
id int
quotation_ic
invoice_id
payments
invoice_ic int
payment_metho enum('efectivo', 'tarjeta', 'transferencia'
amount decimal(10,2)
payment_date datetime
created_at datetime
payment_id int
quotation_items
quotation_ic int
descriptior varchar(255)
quantity int
unit_price decimal(10,2)
quotation_item_i int