DICCIONARIO DB

1) ---------- Sentencias SQL utilizadas

2) ---------- NO SQL

3) ---------- Plan de contingencia

4) ---------- Tablas

5) ---------- Procedimiento (equivalente a PL/SLQ)

Consultas utilizadas:

AlertasStockDao {

```
    SELECT * FROM AlertasStockDao;
}
```

ALMACEN {

```
    SELECT * FROM Almacen;

    INSERT INTO Almacen (Nombre, Capacidad,
    Ubicacion) VALUES ('" + a.getNombre() + "', " +
    a.getCapacidad() + ", '" + a.getUbicacion() + "');

    "DELETE FROM Almacen WHERE ID_Almacen = "
    + a.getId_almacen();

    "UPDATE Almacen SET " +
        "Nombre = '" + a.getNombre() + "', " +
        "Capacidad = " + a.getCapacidad() + ", " +
        "Ubicacion = '" + a.getUbicacion() + "' " +
        "WHERE ID_Almacen = " + a.getId_almacen();
}
```

CATEGORIAS {

SELECT * FROM Categorias;

INSERT INTO Categorias (Nombre) VALUES ('" + categoria.getNombre() + "')

"DELETE FROM Categorias WHERE ID_Categoria = " + a.getId_categoria();

UPDATE Categorias SET Nombre = '" + categoria.getNombre() + "' WHERE ID_Categoria = " + categoria.getId_categoria()

}

Detalles_Facturas {

SELECT * FROM Detalles_Facturas;

INSERT INTO Detalles_Facturas (ID_Detalle_Pedido, ID_Factura, PrecioUnitario, TasaLocal, Descuento) VALUES (" + df.getId_detalle_pedido() + ", " +df.getId_factura() + ", " +df.getPrecioUnitario() + ", " +df.getTasaLocal() + ", " +df.getDescuento() + ")"

DELETE FROM Detalles_Facturas WHERE ID_DetalleFactura = " + df.getId_detalleFactura();

```
        "UPDATE Detalles_Facturas SET " +
        "ID_Detalle_Pedido = " + df.getId_detalle_pedido()
        + ", " + "ID_Factura = " + df.getId_factura() + ", " +
        "PrecioUnitario = " + df.getPrecioUnitario() + ", " +
        "TasaLocal = " + df.getTasaLocal() + ", " +
        "Descuento = " + df.getDescuento() + " WHERE
        ID_DetalleFactura = " + df.getId_detalleFactura();
}
```

Detalles_Pedidos {

```
        SELECT * FROM Detalles_Pedidos;

        INSERT INTO Detalles_Pedidos (ID_Pedido,
        ID_Producto, Cantidad, Observaciones) VALUES
        (?, ?, ?, ?) [Prepared Statement]

        DELETE FROM Detalles_Pedidos WHERE
        ID_DetallePedido = " + idDetalle;

        UPDATE Detalles_Pedidos SET " + "ID_Pedido = "
        + bean.getId_pedido() + ", " + "ID_Producto = " +
        bean.getId_producto() + ", " + "Cantidad = " +
        bean.getCantidad() + ", " + "Observaciones = '" +
        bean.getObservaciones() + "' " + "WHERE
        ID_DetallePedido = " + bean.getId_detallePedido();
}
```

Empleados {

SELECT * FROM Empleados;

INSERT INTO Empleados (ID_Restaurante, Nombre, Apellidos, DNI, Telefono, Sueldo, FechaContratacion) VALUES (" + empleado.getId_restaurante() + ", '" + empleado.getNombre() + "', '" + empleado.getApellidos() + "', '" + empleado.getDni() + "', '" + empleado.getTelefono() + "', " + empleado.getSueldo() + ", '" + sqlDate + "')"

DELETE FROM Empleados WHERE ID_Empleado = " + empleado.getId_empleado();

UPDATE Empleados SET " + "ID_Restaurante = " + empleado.getId_restaurante() + ", " + "Nombre = '" + empleado.getNombre() + "', " + "Apellidos = '" + empleado.getApellidos() + "', " + "DNI = '" + empleado.getDni() + "', " + "Telefono = '" + empleado.getTelefono() + "', " + "Sueldo = " + empleado.getSueldo() + ", " + "FechaContratacion = '" + sqlDate + "' " + "WHERE ID_Empleado = " + empleado.getId_empleado();

}

Historial_Puntos {

```
SELECT * FROM Historial_Puntos;

INSERT INTO Historial_Puntos (ID_Factura, Fecha,
Puntos, TipoMovimiento, Descripcion) VALUES (" +
h.getId_factura() + ", '" + new
Date(h.getFecha().getTime()) + "', " + h.getPuntos()
+ ", '" + h.getTipoMovimiento() + "', '" +
h.getDescripcion() + "')"

DELETE FROM Historial_Puntos WHERE
ID_HistorialPuntos = " + h.getId_historialPuntos();

UPDATE Historial_Puntos SET " + "ID_Factura = "
+ h.getId_factura() + ", " + "Fecha = '" + new
Date(h.getFecha().getTime()) + "', " + "Puntos = " +
h.getPuntos() + ", " + "TipoMovimiento = '" +
h.getTipoMovimiento() + "', " + "Descripcion = '" +
h.getDescripcion() + "' " + "WHERE
ID_HistorialPuntos = " + h.getId_historialPuntos();
}
```

Ingredientes {

```
SELECT * FROM Ingredientes;

INSERT INTO Ingredientes (Nombre,
UnidadMedida, StockDisponible,
TipoAlmacenamiento, EstaActivo) VALUES ('" +
ingrediente.getNombre() + "', '" +
ingrediente.getUnidadMedida() + "', " +
ingrediente.getStockDisponible() + ", '" +
ingrediente.getTipoAlmacenamiento() + "', " +
(ingrediente.isEstaActivo() ? 1 : 0) + ")"

DELETE FROM Ingredientes WHERE
ID_Ingrediente = " + ingrediente.getId_ingrediente()

UPDATE Ingredientes SET " + "Nombre = '" +
ingrediente.getNombre() + "', " + "UnidadMedida =
'" + ingrediente.getUnidadMedida() + "', " +
"StockDisponible = " +
ingrediente.getStockDisponible() + ", " +
"TipoAlmacenamiento = '" +
ingrediente.getTipoAlmacenamiento() + "', " +
"EstaActivo = " + (ingrediente.isEstaActivo() ? 1 :
0) + " WHERE ID_Ingrediente = " +
ingrediente.getId_ingrediente()
}
```

Ofertas {

SELECT * FROM Ofertas;

INSERT INTO Ofertas (Nombre, Precio, Descripcion, columnaImagen, fechaExpiracion) VALUES ('" + oferta.getNombre() + "', " + oferta.getPrecio() + ", '" + oferta.getDescripcion() + "', '" + oferta.getColumnaImagen() + "', '" + sqlDate + "')";

"DELETE FROM Categorias WHERE ID_Categoria = " + a.getId_categoria();

UPDATE Categorias SET Nombre = '" + categoria.getNombre() + "' WHERE ID_Categoria = " + categoria.getId_categoria()

}

Pagos{

SELECT * FROM Pagos;

INSERT INTO Pagos (ID_Factura, metodoPago, fechaPago, estadoPago) VALUES (" + p.getId_factura() + ", '" + p.getMetodoPago() + "', '" + new Date(p.getFechaPago().getTime()) + "', '" + p.getEstadoPago() + "')

```
        DELETE FROM Pagos WHERE ID_Pago = " +
        p.getId_pago();

        UPDATE Pagos SET " + "ID_Factura = " +
        p.getId_factura() + ", " + "metodoPago = '" +
        p.getMetodoPago() + "', " + "fechaPago = '" + new
        Date(p.getFechaPago().getTime()) + "', " +
        "estadoPago = '" + p.getEstadoPago() + "' " +
        "WHERE id_pago = " + p.getId_pago();
}
```

Pedidos {

```
        SELECT * FROM Pedidos;

        INSERT INTO Pedidos (ID_Factura, ID_Restaurante,
        ID_Usuario, Numero) VALUES (?, ?, ?, ?);
        [Prepared Statement]

        DELETE FROM Pedidos WHERE ID_Pedido = " +
        pedido.getId_pedido();

        UPDATE Pedidos SET ID_Factura = " +
        pedido.getId_factura() + ", " + "ID_Restaurante = "
        + pedido.getId_restaurante() + ", " + "ID_Usuario =
        " + pedido.getId_usuario() + ", " + "Numero = " +
        pedido.getNumero() + " WHERE ID_Pedido = " +
        pedido.getId_pedido();
}
```

Productos_Ingredientes {

```
SELECT * FROM Productos_Ingredientes;

INSERT INTO Productos_Ingredientes
(ID_Ingrediente, ID_Producto, Cantidad) VALUES ("
+ pi.getId_ingrediente() + ", " + pi.getId_producto()
+ ", " + pi.getCantidad() + ")";

DELETE FROM Productos_Ingredientes WHERE
ID_ProductoIngrediente = " +
pi.getId_productoIngrediente();

UPDATE Productos_Ingredientes SET " +
"ID_Ingrediente = " + pi.getId_ingrediente() + ", " +
"ID_Producto = " + pi.getId_producto() + ", " +
"Cantidad = " + pi.getCantidad() + " WHERE
ID_ProductoIngrediente = " +
pi.getId_productoIngrediente();
}
```

Productos {

SELECT * FROM Productos;

INSERT INTO Productos (ID_Oferta, ID_Categoria, Nombre, Descripcion, Precio) " + "VALUES (" + producto.getId_oferta() + ", " + producto.getId_categoria() + ", '" + producto.getNombre() + "', '" + producto.getDescripcion() + "', " + producto.getPrecio() + ")";

DELETE FROM Productos WHERE ID_Producto = " + producto.getId_producto();

UPDATE Productos SET " + "ID_Oferta = " + producto.getId_oferta() + ", " + "ID_Categoria = " + producto.getId_categoria() + ", " + "Nombre = '" + producto.getNombre() + "', " + "Descripcion = '" + producto.getDescripcion() + "', " + "Precio = " + producto.getPrecio() + " WHERE ID_Producto = " + producto.getId_producto();

}

Proveedores_Ingredientes {

SELECT * FROM Proveedores_Ingredientes;

INSERT INTO Proveedores_Ingredientes
(ID_Proveedor, ID_Ingrediente, precioUnitario,
tiempoEntregaDias) VALUES (" +
pi.getId_proveedor() + ", " + pi.getId_ingrediente()
+ ", " + pi.getPrecioUnitario() + ", " +
pi.getTiempoEntregaDias() + ")"

DELETE FROM Proveedores_Ingredientes WHERE
ID_ProveedorIngrediente = " +
pi.getId_proveedorIngrediente();

UPDATE Proveedores_Ingredientes SET " +
"ID_Proveedor = " + pi.getId_proveedor() + ", " +
"ID_Ingrediente = " + pi.getId_ingrediente() + ", " +
"precioUnitario = " + pi.getPrecioUnitario() + ", " +
"tiempoEntregaDias = " +
pi.getTiempoEntregaDias() + " " + "WHERE
ID_ProveedorIngrediente = " +
pi.getId_proveedorIngrediente();
}

Proveedores {

SELECT * FROM Proveedores;

INSERT INTO Proveedores (nombreEmpresa, Telefono, Email) VALUES ('" + p.getNombreEmpresa() + "', '" + p.getTelefono() + "', '" + p.getEmail() + "')";

DELETE FROM Proveedores WHERE ID_Proveedor = " + p.getId_proveedor();

UPDATE Proveedores SET " + "nombreEmpresa = '" + p.getNombreEmpresa() + "', " + "Telefono = '" + p.getTelefono() + "', " + "Email = '" + p.getEmail() + "' " + "WHERE ID_Proveedor = " + p.getId_proveedor();

}

Puntos {

SELECT * FROM Puntos;

INSERT INTO Puntos (ID_Usuario, PuntosActuales) VALUES (" + p.getId_usuario() + ", " + p.getPuntosActuales() + ");

“DELETE FROM Puntos WHERE ID_Puntos = " + p.getId_puntos();

```
        UPDATE Puntos SET " + "ID_Usuario = " +
        p.getId_usuario() + ", " + "PuntosActuales = " +
        p.getPuntosActuales() + " WHERE ID_Puntos = " +
        p.getId_puntos();

}

Resenas {

        SELECT * FROM Resenas;

        INSERT INTO Resenas (ID_Usuario,
        ID_Restaurante, Valoracion, Fecha) VALUES (" +
        resena.getId_usuario() + ", " +
        resena.getId_restaurante() + ", " +
        resena.getValoracion() + ", " + (sqlDate != null ? "'"
        + sqlDate + "'" : "NULL") + ")

        "DELETE FROM Resenas WHERE ID_Resena = " +
        resena.getId_resena();

        "UPDATE Resenas SET " + "ID_Usuario = " +
        resena.getId_usuario() + ", " + "ID_Restaurante = "
        + resena.getId_restaurante() + ", " + "Valoracion = "
        + resena.getValoracion() + ", " + "Fecha = " +
        (sqlDate != null ? "'" + sqlDate + "'" : "NULL") + " "
        + "WHERE ID_Resena = " + resena.getId_resena();
}
```

Restaurante {

```
SELECT * FROM Restaurante;

INSERT INTO Restaurante (Nombre, Direccion,
Telefono, Email, Aforo, imagenRestaurante)
VALUES ('" + restaurante.getNombre() + "', '" +
restaurante.getDireccion() + "', '" +
restaurante.getTelefono() + "', '" +
restaurante.getEmail() + "', " +
restaurante.getAforo() + ", '" +
restaurante.getImagenRestaurante() + "')

"DELETE FROM Restaurante WHERE
ID_Restaurante = " +
restaurante.getId_restaurante();

"UPDATE Restaurante SET " + "Nombre = '" +
restaurante.getNombre() + "', " + "Direccion = '" +
restaurante.getDireccion() + "', " + "Telefono = '" +
restaurante.getTelefono() + "', " + "Email = '" +
restaurante.getEmail() + "', " + "Aforo = " +
restaurante.getAforo() + ", " + "imagenRestaurante
= '" + restaurante.getImagenRestaurante() + "' " +
"WHERE ID_Restaurante = " +
restaurante.getId_restaurante();
```

}

Usuarios {

SELECT * FROM Usuarios;

INSERT INTO Usuarios (Nombre, Email, Contrasena, DNI, Telefono, Direccion) VALUES ('" + usuario.getNombre() + "', '" + usuario.getEmail() + "', '" + usuario.getContrasena() + "', '" + usuario.getDni() + "', '" + usuario.getTelefono() + "', '" + usuario.getDireccion() + "')

"DELETE FROM Usuarios WHERE ID_Usuario = " + usuario.getId_usuario();

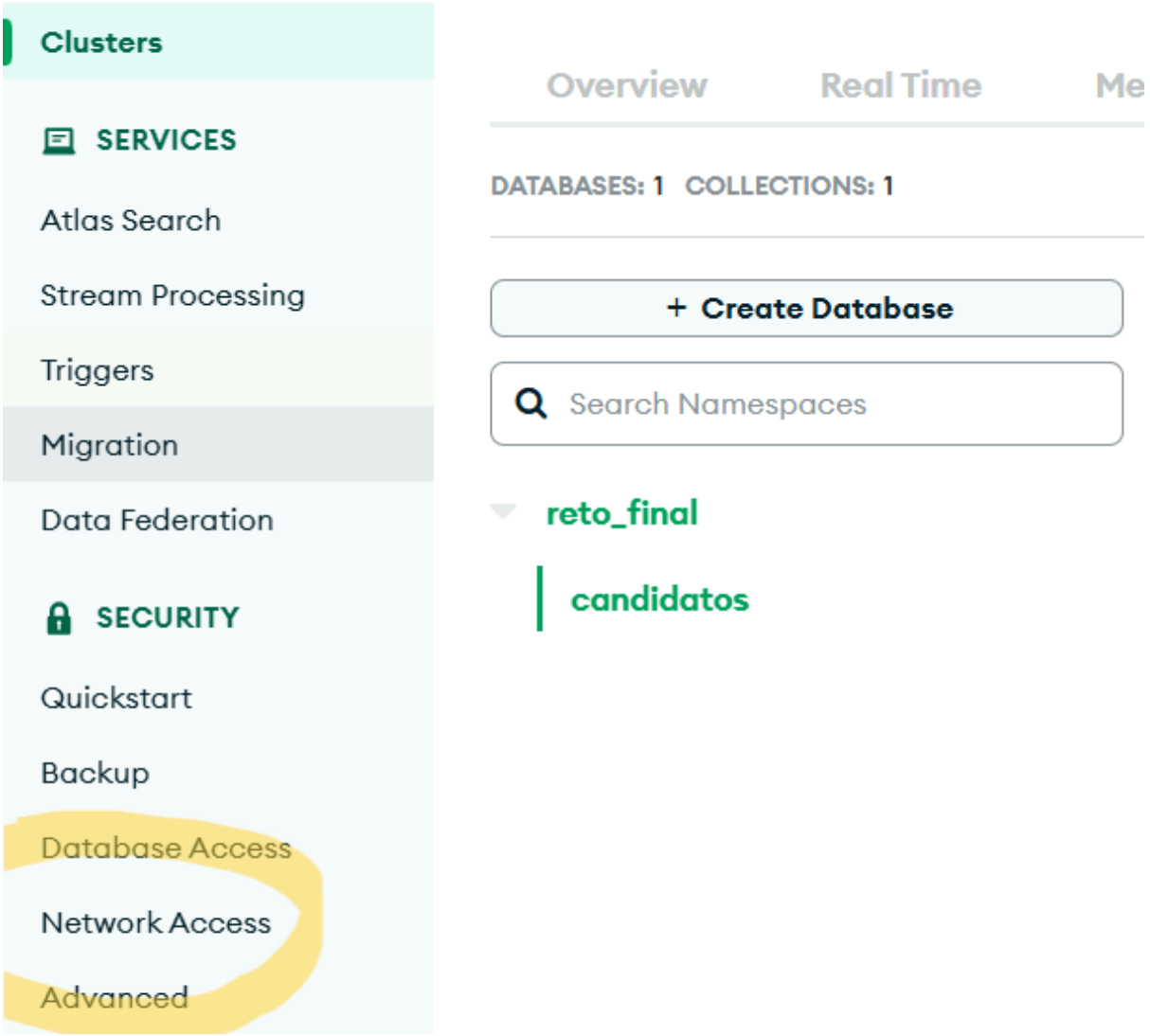
"UPDATE Usuarios SET " + "Nombre = '" + usuario.getNombre() + "', " + "Email = '" + usuario.getEmail() + "', " + "Contrasena = '" + usuario.getContrasena() + "', " + "DNI = '" + usuario.getDni() + "', " + "Telefono = '" + usuario.getTelefono() + "', " + "Direccion = '" + usuario.getDireccion() + "' " + "WHERE ID_Usuario = " + usuario.getId_usuario();

}

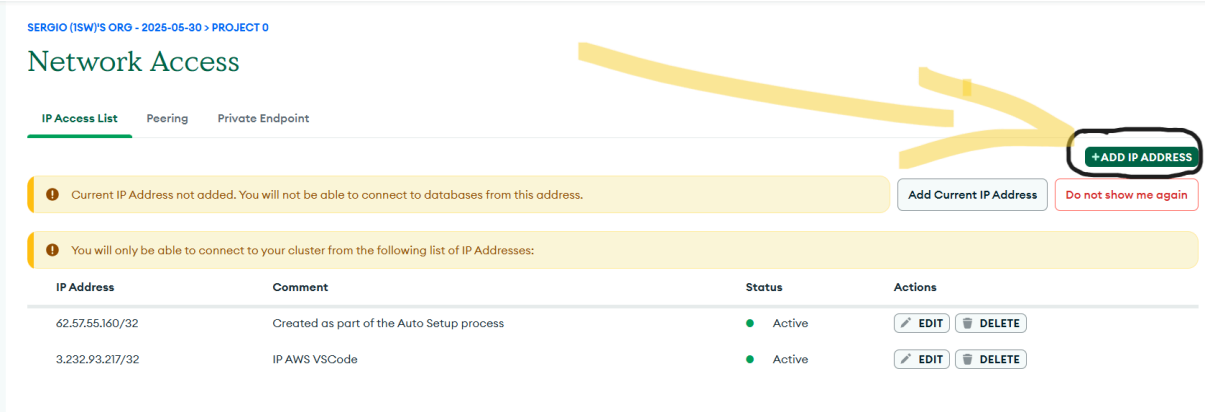
NO SQL

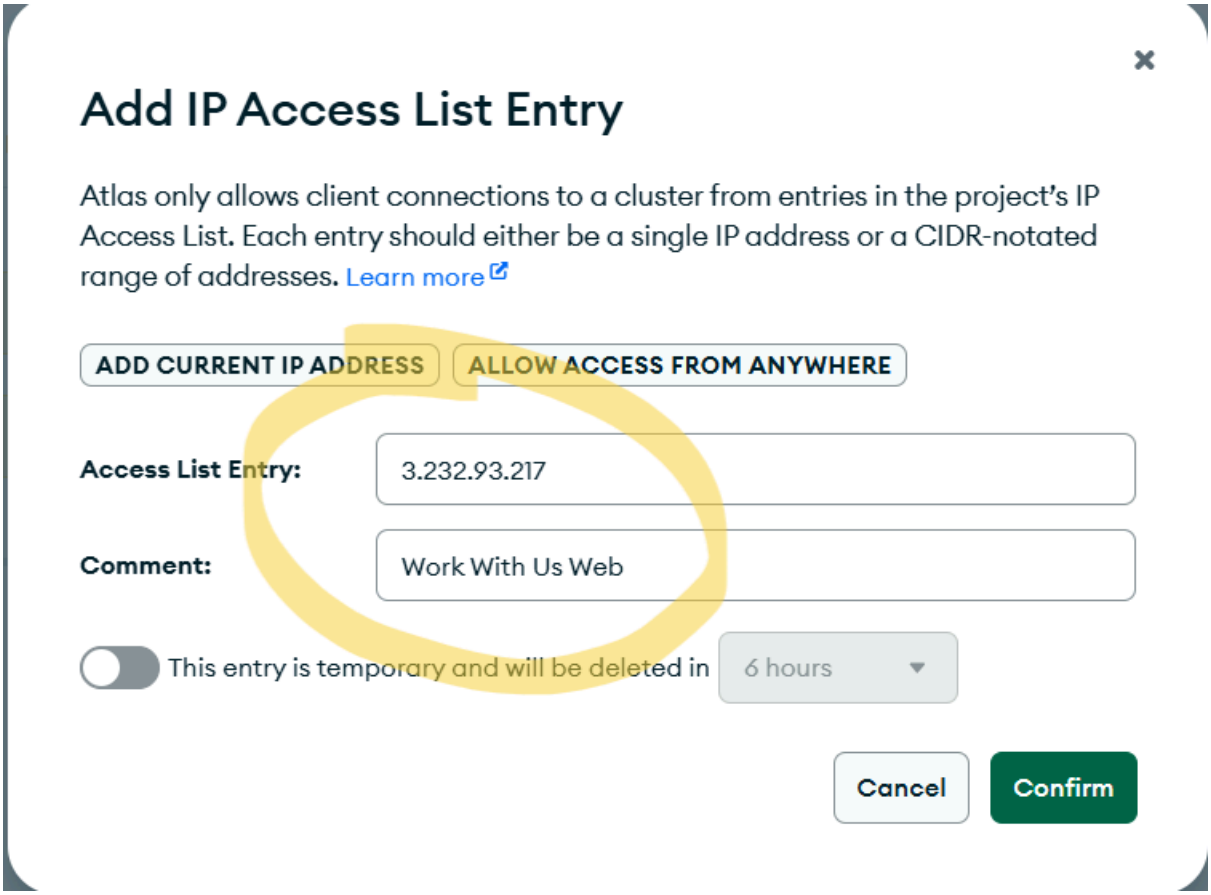
Nos hemos creado una cuenta en la web de mongodb Atlas. Hemos creado una BD nueva y una colección (el equivalente a una tabla en sql) llamada candidatos (creada con el back).

Tras ello, hemos configurado la DB para que se pueda conectar desde la IP conectada a nuestro dominio.
Le damos a Network Access



Después, le damos a Add IP Address y metemos nuestra IP para que la clase del backend se pueda conectar con la DB y pueda insertar los datos en ella.





Todo esto junto con un fetch que llama al servlet del backend que mostraremos a continuación y le envía los datos que inserta el usuario en la web con un POST, y este, a través de su conexión con la DB de mongo se conecta a ella y le envía los datos metiéndolos en la colección candidatos.

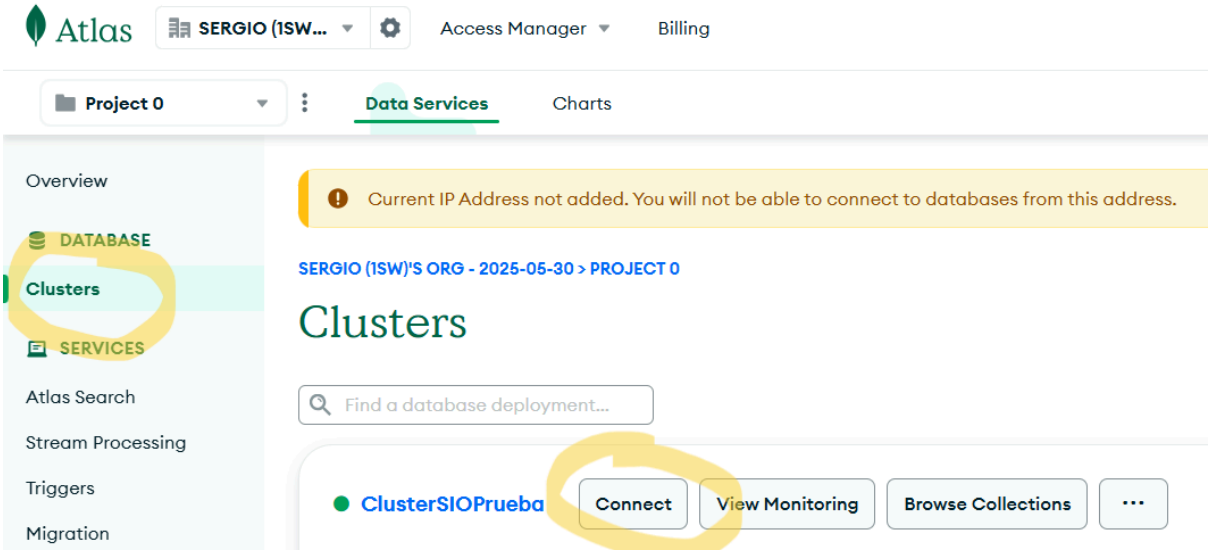
Fetch que gestiona el envío de currículums así como todos los datos cumplimentados con relación a estos.

```
//GESTIONAR CV
window.addEventListener('DOMContentLoaded', () => {
  console.log("JS cargado correctamente");
  document.getElementById('cv').addEventListener('submit', async function (e) {
    e.preventDefault();
    console.log("Interceptado el submit");
    const form = e.target;
    const formData = new FormData(form);

    for (let [key, value] of formData.entries()) {
      console.log(key, value);
    }

    try {
      const res = await fetch('http://3.232.93.217:8080/api/uploadCV', {
        method: 'POST',
        body: formData,
      });

      const data = await res.json();
      alert(data.message);
    } catch (err) {
      console.error(err);
      alert('Error al subir el archivo');
    }
  });
});
```

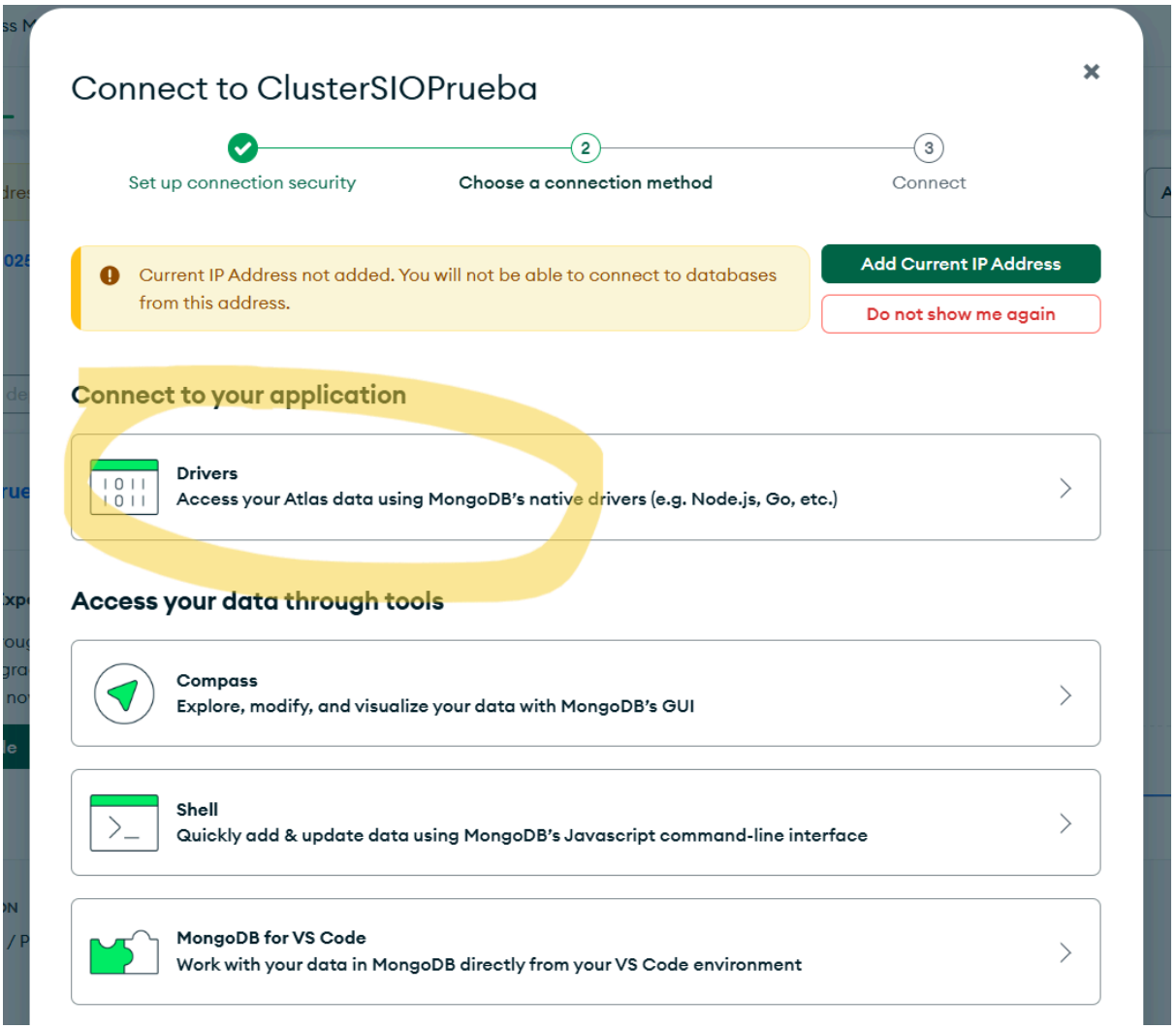
Servlet que gestiona la conexión con la MongoDB así como la publicación de los datos recogidos del frontend con la api a la colección “candidatos”.

```java
@WebServlet("/api/uploadCV")  new *
@MultipartConfig
public class CVServlet extends HttpServlet {
    private static final String UPLOAD_DIR = "/home/ec2-user/retoBueno/CV-candidatos";  2 usages
    private static final String MONGO_URI = "mongodb+srv://root:Aa12021888.@clustersioprueba.ykpwkjn.mor

    @Override  no usages  new *
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws IOException {
        try {
            // Crear carpeta si no existe
            File uploadDir = new File(UPLOAD_DIR);
            if (!uploadDir.exists()) uploadDir.mkdirs();

            Part filePart = req.getPart( s: "cv");
            String nombre = req.getParameter( s: "nombre");
            String apellido = req.getParameter( s: "last-name");
            String email = req.getParameter( s: "email");
            String restaurant = req.getParameter( s: "candidacy");
            String job = req.getParameter( s: "job");
            String policy = req.getParameter( s: "policy");

            String fileName = Paths.get(filePart.getSubmittedFileName()).getFileName().toString();
            String fullPath = UPLOAD_DIR + File.separator + fileName;
            filePart.write(fullPath);
```

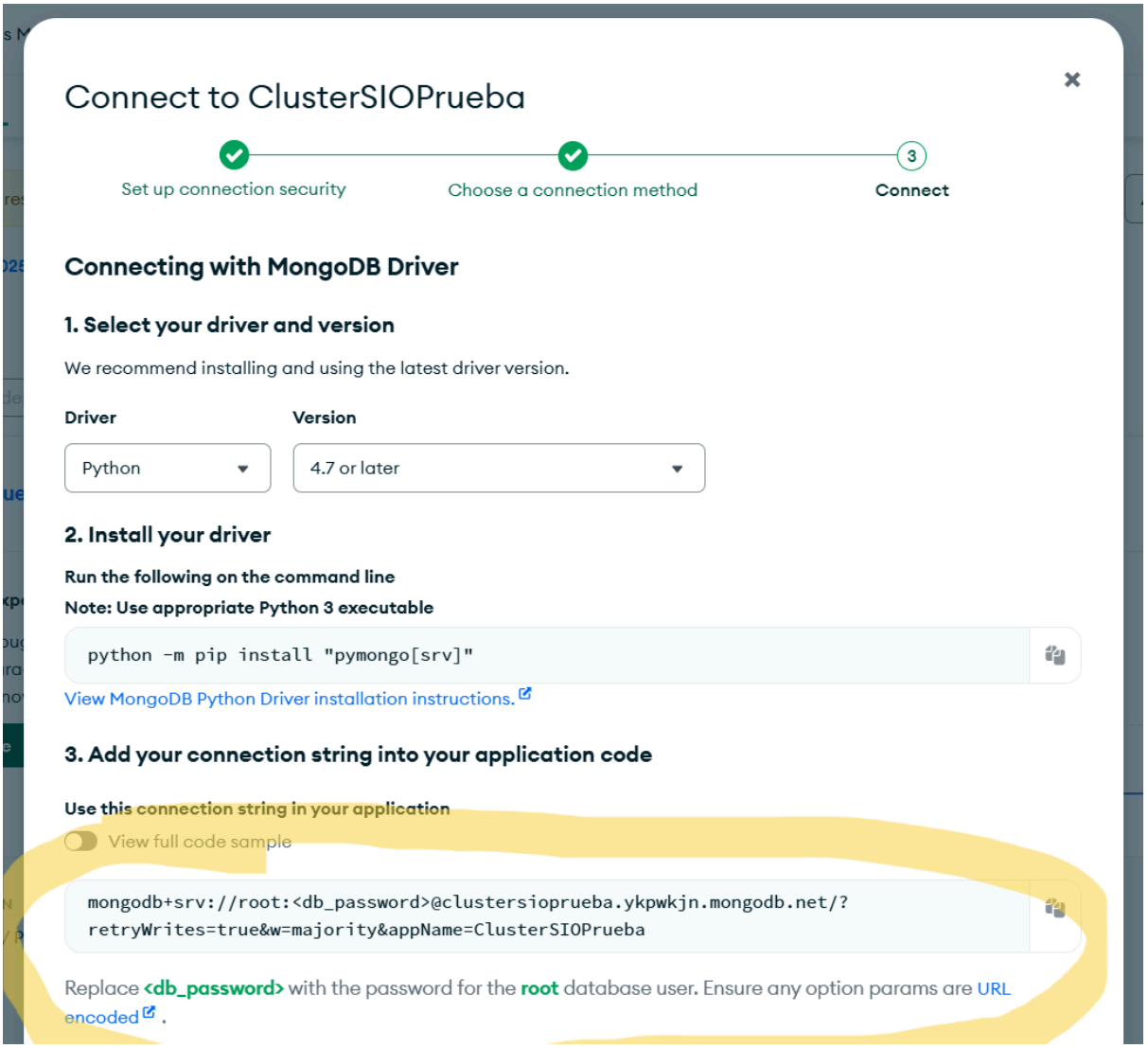
En la constante MONGO_URI definimos la url de conexión por la que se van a comunicar backend de Java y MongoDB. Que en la web de MongoDB lo obtenemos así:



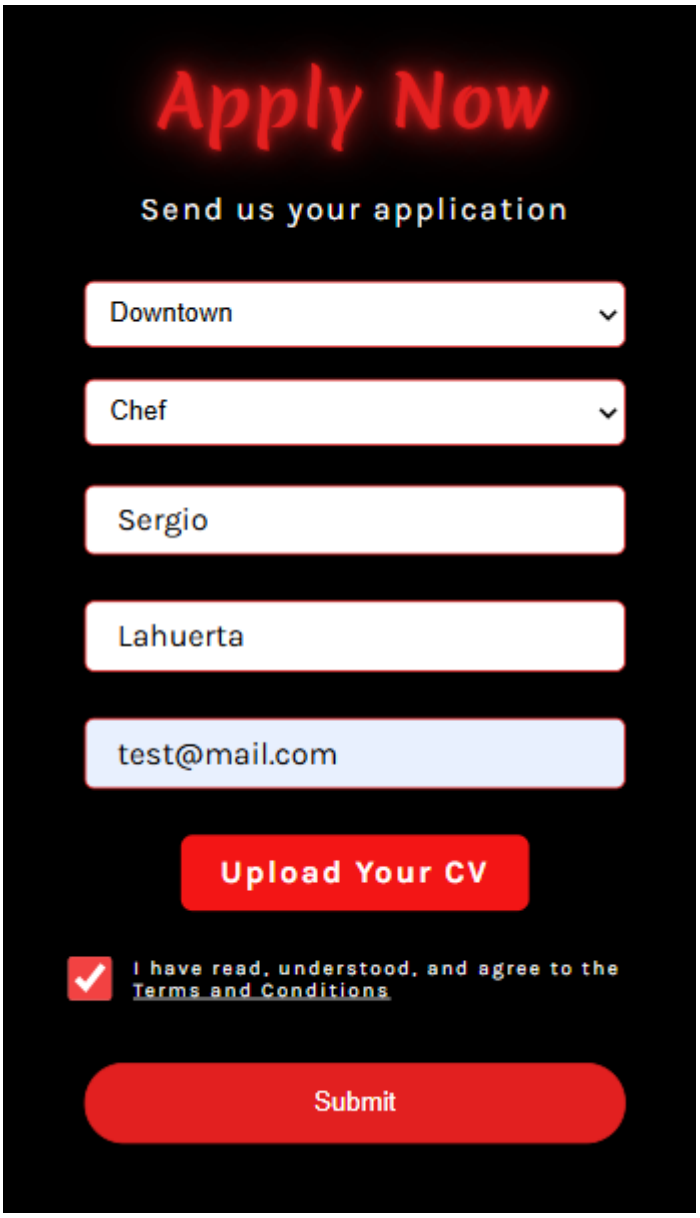
Tras darle a Clusters, nos saldrá el nuestro y cuando lo visualicemos, le daremos a Connect para elegir el método por el que nos vamos a comunicar con el back.
Elegimos por Drivers:



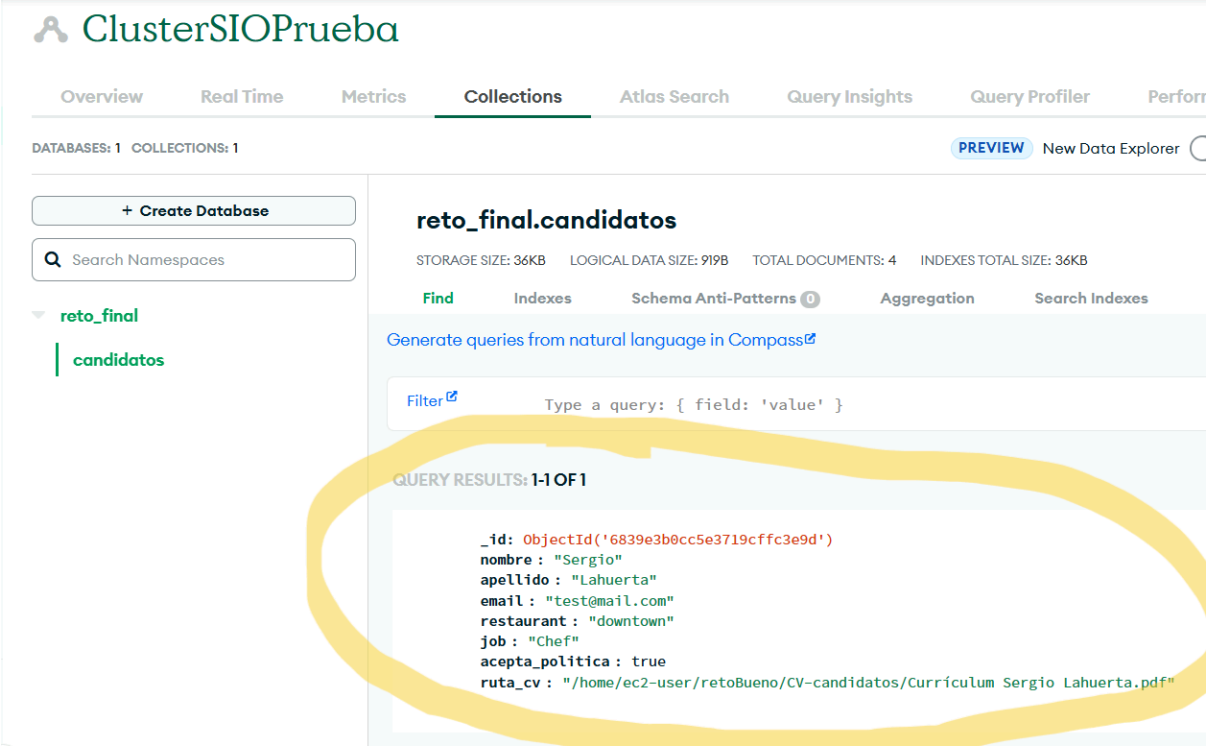
A continuación, debemos bajar en el display que nos aparece hasta ver la cadena de conexión que nos sale en "3. Add your connection string into your application code".
Con ella, cambiando el `<db_password>` por nuestra contraseña y el `root` por nuestro nombre de usuario de la DB para que la conexión sea exitosa. Como se muestra en la siguiente imagen.


Connect to ClusterSIOPrueba
Set up connection security
Choose a connection method
3
Connect
Connecting with MongoDB Driver
1. Select your driver and version
We recommend installing and using the latest driver version.
Driver
Python
Version
4.7 or later
2. Install your driver
Run the following on the command line
Note: Use appropriate Python 3 executable
python -m pip install "pymongo[srv]"
View MongoDB Python Driver installation instructions.
3. Add your connection string into your application code
Use this connection string in your application
View full code sample
mongodb+srv://root:<db_password>@clustersioprueba.ykpwkjn.mongodb.net/?retryWrites=true&w=majority&appName=ClusterSIOPrueba
Replace <db_password> with the password for the root database user. Ensure any option params are URL encoded .

Por último, el usuario al ingresar los datos en la web y apretar el botón Submit, envía los datos al backend con el fetch y este los publica en la MongoDB en la colección candidatos. De tal manera:



Así luego en la web de MongoDB en nuestra colección vemos los datos introducidos;



Plan de contingencia

Drop schema if exists esquema_reto ➜ Elimina el esquema con ese nombre si existe, si no, se ejecuta la línea siguiente.

Create schema esquema_reto ➜ Crea un esquema con ese nombre, por el cual no estará llamado otro esquema gracias a la consulta anterior.

Use esquema_reto ➜ Comunica al gestor de la base de datos que las consultas posteriores las aplique al esquema especificado.

Drop table if exists en el siguiente orden de las tablas ➜ Elimina las tablas en el orden en el que se encuentren con esos nombres si existen, en caso contrario, se ejecuta la siguiente consulta.

Historial_Puntos,
Puntos,
Detalles_Facturas,
Pagos,
Facturas,
Detalles_Pedidos,
Pedidos,
Resenas,
Empleados,
Proveedores_Ingredientes,
Productos_Ingredientes,
Productos,
Ofertas,
Usuarios,
Ingredientes,
Restaurante,
Proveedores,
Almacen,
Categorias,
Alertas_Stock;

Tablas:

Tabla	Campo	Tipo de dato	Restricciones	Descripción
Categorías	ID_Categoria	INT	PK, Auto_increment	Identificador único de la categoría
	Nombre	Varchar (100)	NOT NULL	Nombre de la Categoría

Tabla	Campo	Tipo de dato	Restricciones	Descripción
Ofertas	ID_Oferta	INT	PK, Auto_increment	Identificador de la oferta
	Nombre	Varchar (100)	NOT NULL	Nombre de la oferta
	Precio	Decimal (10,2)	NOT NULL	Precio de la oferta
	Descripción	TEXT	NULL permitido	Descripción de oferta
	columnaImagen	Varchar (255)	NULL permitido	Imagen de la oferta
	fechaExpiracion	DATE	NOT NULL	fecha de caducidad de la oferta

Tabla	Campo	Tipo de dato	Restricciones	Descripción
Productos	ID_Producto	INT	PK, Auto_increment	Identificador del producto
	Nombre	Varchar (100)	NOT NULL	Nombre del producto
	Precio	Decimal (10,2)	NOT NULL	Precio del producto
	Descripción	TEXT	NULL permitido	Descripción del producto
	ImagenProducto	Varchar (250)	NOT NULL	Imagen del producto
	ID_Oferta	INT	FK → Ofertas	Oferta asociada
	ID_Categoria	INT	FK → Categorías	Categoría del producto

Tabla	Campo	Tipo de dato	Restricciones	Descripción
Ingredientes	ID_Ingrediente	INT	PK, Auto_increment	Identificador del ingrediente
	Nombre	Varchar (100)	NOT NULL	Nombre del ingrediente
	UnidadMedida	Varchar(50)	NULL permitido	Unidad de Medida
	StockDisponible	Decimal (10,2)	NOT NULL	Cantidad disponible en stock
	TipoAlmacenamiento	Varchar(50)	NULL permitido	Condiciones almacenamiento
	EstaActivo	Boolean	NULL permitido	Si el ingrediente está disponible

Tabla	Campo	Tipo de dato	Restricciones	Descripción
Productos_Ingredientes	ID_ProductoIngrediente	INT	PK, Auto_increment	Identificador del vínculo producto-ingrediente
	ID_Ingrediente	INT	FK → Ingredientes	Ingrediente relacionado
	ID_Producto	INT	FK → Productos	Producto relacionado
	Cantidad	Decimal (10,2)	NOT NULL	Cantidad de ingrediente usado

Tabla	Campo	Tipo de dato	Restricciones	Descripción
Proveedores	ID_Proveedor	INT	PK, Auto_increment	Identificador del proveedor
	NombreEmpresa	Varchar (100)	NOT NULL	Nombre de la empresa proveedora
	Telefono	Varchar (100)	NOT NULL	Telefono del proveedor
	Email	Varchar (100)	NULL permitido	Email del proveedor

Tabla	Campo	Tipo de dato	Restricciones	Descripción
Almacen	ID_Almacen	INT	PK, Auto_increment	Identificador del almacén
	Nombre	Varchar (100)	NOT NULL	Nombre del almacén
	Capacidad	INT	NOT NULL	Capacidad total del almacén
	Ubicación	Varchar (100)	NULL permitido	Dirección o Ubicación

Tabla	Campo	Tipo de dato	Restricciones	Descripción
Proveedores_Ingredientes	ID_ProveedorIngrediente	INT	PK, Auto_increment	ID de relación proveedor-ingrediente
	ID_Proveedor	INT	FK → Proveedores	Proveedor del ingrediente
	ID_Ingrediente	DECIMAL(10,2)	FK → Ingredientes	Ingrediente suministrado
	precioUnitario	Varchar (100)	NOT NULL	Precio por unidad
	tiempoEntregaDias	INT	NULL permitido	Tiempo estimado de entrega

Tabla	Campo	Tipo de dato	Restricciones	Descripción
Restaurante	ID_Restaurante	INT	PK, Auto_increment	Identificador del restaurante
	Nombre	Varchar (100)	NOT NULL	Nombre restaurante
	Direccion	Varchar (100)	NULL permitido	Dirección
	Telefono	Varchar (100)	NULL permitido	Teléfono
	Email	Varchar (100)	NULL permitido	Correo Electrónico
	Aforo	INT	NULL permitido	Capacidad máxima del restaurante
	imagenRestaurante	Varchar (250)	NULL permitido	Imagen del restaurante

Tabla	Campo	Tipo de dato	Restricciones	Descripción
Empleados	ID_Empleado	INT	PK, Auto_increment	Identificador del empleado
	ID_Restaurante	INT	FK → Restaurante	Restaurante asignado
	Nombre	Varchar (100)	NOT NULL	Nombre empleados
	Apellidos	Varchar (100)	NOT NULL	Apellidos empleados
	DNI	Varchar (20)	NOT NULL	Documento de Identidad
	Telefono	Varchar (20)	NULL permitido	Teléfono
	Sueldo	DECIMAL(10,2)	NOT NULL	Salario
	FechaContratacion	DATE	NOT NULL	Fecha de contratación

Tabla	Campo	Tipo de dato	Restricciones	Descripción
Usuarios	ID_Usuarios	INT	PK, Auto_increment	Identificador del usuario
	Nombre	Varchar (100)	NULL permitido	Nombre usuarios
	DNI	Varchar (20)	NULL permitido	Documento de Identidad
	Telefono	Varchar (20)	NULL permitido	Teléfono
	Email	Varchar (100)	NOT NULL UNIQUE	Correo Electrónico único
	Contrasena	Varchar (100)	NOT NULL	Contraseña
	DNI	Varchar (20)	NULL permitido	Documento de identidad
	Direccion	Varchar (200)	NULL permitido	Dirección

Tabla	Campo	Tipo de dato	Restricciones	Descripción
Reseñas	ID_Reseñas	INT	PK, Auto_increment	ID de reseña
	ID_Usuario	INT	FK → Usuarios	Usuario que opina
	ID_Restaurante	INT	FK → Restaurante	Restaurante reseñado
	Valoracion	INT	NULL permitido	Puntuación
	Fecha	DATE	NULL permitido	Fecha de la reseña

Tabla	Campo	Tipo de dato	Restricciones	Descripción
Pedidos	ID_Pedido	INT	PK, Auto_increment	ID del pedido
	ID_Factura	INT	FK → Facturas	Factura asociada
	ID_Restaurante	INT	FK → Restaurante	Restaurante donde se realiza
	ID_Usuario	INT	FK → usuarios	Usuario que realiza el pedido
	Numero	INT	NOT NULL	Número o folio del pedido

Tabla	Campo	Tipo de dato	Restricciones	Descripción
Detalles_Pedidos	ID_DetallePedido	INT	PK, Auto_increment	ID de detalle del pedido
	ID_Pedido	INT	FK → Pedidos	Pedido al que pertenece
	ID_Producto	INT	FK → Productos	Producto solicitado
	Cantidad	INT	NOT NULL	Cantidad solicitada
	Observaciones	TEXT	NULL permitido	Comentarios o notas

Tabla	Campo	Tipo de dato	Restricciones	Descripción
Facturas	ID_Factura	INT	PK, Auto_increment	ID de la factura
	FechaFactura	DATE	NOT NULL	Fecha de emisión
	ImporteTotal	DECIMAL(10,2)	NOT NULL	Total de la factura

Tabla	Campo	Tipo de dato	Restricciones	Descripción
Detalles_Facturas	ID_DetalleFactura	INT	PK, Auto_increment	ID de detalle de la factura
	ID_Factura	INT	FK → Facturas	Factura correspondiente
	ID_DetallePedidos	INT	FK → Detalles_Pedidos	Línea de pedido asociada
	PrecioUnitario	DECIMAL(10,2)	NOT NULL	Precio por unidad
	TasaLocal	DECIMAL(10,2)	NULL permitido	Tasa local
	Descuento	DECIMAL(10,2)	NULL permitido	Descuento aplicado

Tabla	Campo	Tipo de dato	Restricciones	Descripción
Puntos	ID_puntos	INT	PK, Auto_increment	ID de puntos de usuario
	ID_Usuario	INT	FK → Usuarios	Usuario
	PuntosActuales	INT	NOT NULL	Puntos actuales acumulados

Tabla	Campo	Tipo de dato	Restricciones	Descripción
Historial_Puntos	ID_HistorialPuntos	INT	PK, Auto_increment	ID el movimiento de puntos
	ID_Factura	INT	FK → Facturas	Factura relacionada
	Fecha	DATE	NOT NULL	Fecha del movimiento
	Puntos	INT	NOT NULL	Cantidad de puntos ganados/canjeados
	TipoMovimiento	ENUM('GANADO', 'CANJEADO')	NOT NULL	Tipo de movimiento
	Descripcion	TEXT	NULL permitido	Descripción de movimiento

Tabla	Campo	Tipo de dato	Restricciones	Descripción
Pagos	ID_Pago	INT	PK, Auto_increment	ID del pago
	ID_Factura	INT	FK → Facturas	Factura pagada
	metodoPago	Varchar(50)	NOT NULL	Método de pago (efectivo, tarjeta…)
	fechaPago	DATE	NOT NULL	Fecha de pago
	estadoPago	Varchar(20)	NOT NULL	Estado del pago (pendiente, completado…)

Tabla	Campo	Tipo de dato	Restricciones	Descripción
Alertas_Stock	ID_Alerta	INT	PK, Auto_increment	ID del pago
	ID_Ingrediente	INT	FK → Facturas NOT NULL	ID del ingrediente
	fechaAlerta	DateTime	NOT NULL	fecha en la que se envía la alerta
	stockDisponible	decimal	NULL PERMITIDO	Cantidad de stock disponible
	mensaje	text	NULL PERMITIDO	Mensaje indicando el bajo stock de dicho producto

Procedimiento necesario:

Usamos el lenguaje procedural nativo de MySQL (procedimientos almacenados) que es el equivalente al PL/SQL de Oracle.

DELIMITER $$ ➜ Cambia el delimitador de SQL por defecto que es ; a $$.

Drop Procedure if exists RevisarStockBajo ➜ Elimina un procedimiento con ese nombre en caso de que exista.

Create Procedure RevisarStockBajo() ➜ Crea el procedimiento RevisarStockBajo.

Begin y End$$ ➜ Todo lo que ocurre dentro se ejecuta cuando se llama al procedimiento RevisarStockBajo().

INSERT INTO Alertas_Stock (ID_Ingrediente, StockDisponible, Mensaje)
SELECT
ID_Ingrediente,
StockDisponible,
CONCAT('Stock bajo: ', StockDisponible, ' unidades disponibles para ingrediente ID ', Nombre, '''')
FROM Ingredientes

WHERE StockDisponible < 10;

➜ Consulta que comunica al procedimiento lo que debe hacer. En este caso se insertan los valores de ID_Ingrediente, StockDisponible, Mensaje en la tabla de Alertas_Stock cuando un ingrediente de la tabla Ingredientes tiene un StockDisponible menor a 10.

Delimeter ; ➜ Final del procedimiento.

CALL RevisarStockBajo(); ➜ Llama a ese procedimiento para luego poder revisar en la tabla Alertas_Stock que ingredientes hay que reponer.