***Grupo ATU***

***Administración de Servicios de Internet***

*Trabajo realizado por:*

*Francisco Javier Otero Herrero*

*María del Carmen Pérez Rivas*

***Métodos de Acceso más usados para Conectar con los Sistemas Gestores de Bases de Datos***

***Contenido***

[***Conexión Directa a través de Drivers Específicos 3***](#_Toc195902822)

[***Uso de APIs RESTful o GraphQL 4***](#_Toc195902823)

[***ORM (Object-Relational Mapping) 5***](#_Toc195902824)

[***Middleware de Base de Datos 6***](#_Toc195902825)

[***Conexión a través de Protocolos de Red 7***](#_Toc195902826)

[***Conexión a través de Herramientas de Integración 8***](#_Toc195902827)

Cuando se desarrolla un sistema que interactúa con una base de datos, es fundamental establecer una conexión eficiente y segura entre la aplicación y el gestor de bases de datos ***(DBMS).*** Existen varios métodos de acceso para conectar sistemas con gestores de bases de datos. A continuación, se van a describir detalladamente los métodos más utilizados:

# Conexión Directa a través de Drivers Específicos

Este método utiliza drivers específicos proporcionados por el gestor de bases de datos para conectarse directamente al servidor de la base de datos.

* ***Características Principales:***
* Drivers Comunes: JDBC (Java), ODBC (Open Database Connectivity), PDO (PHP Data Objects), etc.

***Funcionamiento:***

El sistema utiliza un driver específico del lenguaje de programación o plataforma para conectarse a la base de datos. El driver traduce las consultas ***SQL*** en instrucciones que el ***DBMS*** puede entender.

***Ejemplo Práctico:***

* ***En PHP, se usa PDO o MySQLi*** para conectarse a ***MySQL/MariaDB.***
* En ***Java,*** se usa ***JDBC*** para conectarse a bases de datos como ***MySQL, PostgreSQL o Oracle.***

***Ventajas:***

* *Rendimiento optimizado debido a la comunicación directa.*
* *Soporte nativo para características específicas del DBMS.*
* *Mayor seguridad si se configuran correctamente las conexiones (uso de SSL, cifrado, etc.).*

***Desventajas:***

* Dependencia del driver específico del ***DBMS.***
* Puede ser complejo migrar a otro ***DBMS*** si el código está fuertemente acoplado al driver.

# Uso de APIs RESTful o GraphQL

Este método implica que el sistema interactúe con la base de datos a través de una API intermedia, como una ***API RESTful o GraphQL,*** en lugar de conectarse directamente al ***DBMS.***

* ***Características Principales:***
* Arquitectura: Cliente-Servidor, donde el cliente envía solicitudes HTTP/HTTPS al servidor API.

***Funcionamiento:***

El servidor API actúa como intermediario entre el sistema y la base de datos. Las consultas ***SQL*** se encapsulan en endpoints ***RESTful*** o consultas ***GraphQL***.

Ejemplos comunes incluyen frameworks como ***Express.js (Node.js) o Django REST Framework (Python).***

***Ventajas:***

* Abstracción de la base de datos: El sistema no necesita conocer detalles del DBMS.
* Facilita la escalabilidad y el desacoplamiento entre el sistema y la base de datos.
* Mejora la seguridad al exponer solo los endpoints necesarios en lugar de permitir acceso directo a la base de datos.

***Desventajas:***

* Introduce una capa adicional (el servidor API), lo que puede aumentar la latencia.
* Requiere más recursos para mantener y desarrollar la API.

# ORM (Object-Relational Mapping)

El uso de un ***ORM*** permite trabajar con la base de datos utilizando objetos en lugar de escribir consultas ***SQL*** directamente.

* ***Características Principales:***
* Herramientas Comunes: ***Hibernate (Java), Entity Framework (.NET), Sequelize (Node.js), SQLAlchemy (Python), Eloquent (Laravel).***

***Funcionamiento:***

El ***ORM*** traduce objetos y relaciones en el código del sistema a tablas y relaciones en la base de datos. Permite realizar operaciones ***CRUD (Create, Read, Update, Delete)*** sin escribir ***SQL*** manualmente.

***Ventajas:***

* Abstracción completa de la base de datos: No es necesario escribir consultas SQL.
* Portabilidad: Cambiar de DBMS puede ser más fácil si el ORM soporta múltiples bases de datos.
* Productividad: Reduce el tiempo de desarrollo al automatizar tareas repetitivas.

***Desventajas:***

* Menor control sobre las consultas SQL generadas, lo que puede afectar el rendimiento.
* Curva de aprendizaje para dominar el ORM.
* Puede generar consultas ineficientes si no se configura correctamente.

# Middleware de Base de Datos

Este método utiliza un software intermedio que actúa como puente entre el sistema y la base de datos.

* ***Características Principales:***
* Ejemplos: ***Servidores de aplicaciones como Apache Tomcat, JBoss,*** o soluciones específicas como ***Redis*** como caché intermedia.

***Funcionamiento:***

El ***middleware*** gestiona las conexiones a la base de datos, realiza operaciones de ***caché*** y optimiza el rendimiento. Actúa como un ***proxy*** que maneja múltiples solicitudes simultáneas.

***Ventajas:***

* Mejora el rendimiento mediante el uso de cachés y la gestión de conexiones persistentes.
* Centraliza la lógica de acceso a la base de datos, facilitando su mantenimiento.
* Proporciona una capa de seguridad adicional al filtrar y validar solicitudes.

***Desventajas:***

* Aumenta la complejidad del sistema.
* Requiere configuración y mantenimiento adicional.

# Conexión a través de Protocolos de Red

Este método utiliza protocolos de red estándar para comunicarse con la base de datos.

* ***Características Principales:***
* Protocolos Comunes: ***TCP/IP, SSH, HTTP/HTTPS.***

***Funcionamiento:***

El sistema se conecta al ***DBMS*** a través de un protocolo de red, generalmente utilizando sockets. Por ejemplo: Conexión a ***MySQL*** a través de ***TCP/IP*** usando localhost o una dirección IP remota.

***Ventajas:***

* Flexibilidad para conectarse a bases de datos remotas.
* Compatible con la mayoría de los DBMS modernos.

***Desventajas:***

* Mayor exposición a ataques si no se configuran correctamente las conexiones (por ejemplo, falta de cifrado).
* Latencia adicional si la base de datos está en un servidor remoto.

# Conexión a través de Herramientas de Integración

Este método utiliza herramientas de integración de datos para conectar sistemas con bases de datos.

* ***Características Principales:***

Herramientas Comunes: ***Apache Kafka, Apache NiFi, Talend, Pentaho.***

***Funcionamiento:***

Estas herramientas permiten integrar datos de múltiples fuentes y enviarlos a la base de datos. Se utilizan comúnmente en proyectos de ***Big Data o ETL (Extract, Transform, Load).***

***Ventajas:***

* Ideal para sistemas que requieren integración de datos desde múltiples fuentes.
* Automatización de procesos ETL.

***Desventajas:***

* Complejidad en la configuración y mantenimiento.
* Requiere infraestructura adicional.

# Comparativa de Métodos de Acceso

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *MÉTODO* | *RENDIMIENTO* | *SEGURIDAD* | *FACILIDAD DE USO* | *ESCALABILIDAD* | *PORTABILIDAD* |
| *Conexión directa* | ***Alto*** | ***Media-Alta*** | ***Media*** | ***Media*** | ***Baja*** |
| *API RESTful/GraphQL* | ***Medio*** | ***Alta*** | ***Alta*** | ***Alta*** | ***Alta*** |
| *ORM* | ***Medio*** | ***Media*** | ***Alta*** | ***Media*** | ***Alta*** |
| *Middleware* | ***Alto*** | ***Alta*** | ***Media*** | ***Alta*** | ***Media*** |
| *Protocolos de red* | ***Medio*** | ***Media*** | ***Media*** | ***Media*** | ***Media*** |
| *Herramientas de integración* | ***Bajo-Medio*** | ***Alta*** | ***Baja*** | ***Alta*** | ***Media*** |

*El método de acceso más adecuado depende de las necesidades específicas del proyecto:*

* *Para proyectos simples o de alta velocidad,* ***conexión directa a través de drivers es ideal.***
* *Para sistemas distribuidos o APIs,* **API RESTful/GraphQL** *ofrece mayor flexibilidad y seguridad.*
* *Para desarrollos ágiles y portabilidad,* **ORM** *es una excelente opción.*
* *Para mejorar el rendimiento y gestionar múltiples conexiones,* **middleware** *es una buena elección.*
* *Para proyectos de Big Data o integración compleja,* **herramientas de integración** *son indispensables.*