

# Conceptos de bases de datos

1

## Definición de base de datos

Seguramente existen muchas definiciones del término base de datos, así como también hay diferentes sistemas de gestión de bases de datos. Por esta razón, trataremos de ser directos y concisos. Una base de datos es una colección de datos, que puede estar representada por información acerca de cualquier tema. Un número telefónico es un dato, un nombre es un dato, un libro contiene datos, etcétera. De qué tratan los datos en realidad no es importante en el contexto de la definición de datos.

Por ejemplo, una agenda es conceptualmente una base de datos normalmente organizada (ordenada) por días. Una base de datos para ser considerada como tal no tiene obligación de tener un soporte electrónico, ni tampoco tiene que estar organizada de una manera particular.

## Motivos que justifican el uso de una base de datos

Cuando pensamos en los motivos por los que se usan las bases de datos, algunos conceptos fundamentales nos vienen a la mente: almacenamiento, accesibilidad, organización y manipulación.

- **Almacenamiento:** Podemos almacenarla electrónicamente en un disco y llevársela con nosotros, si eso es lo que se quiere, o entregarla a alguna otra persona. Una base de datos también podría ser almacenada en una hoja de cálculo Excel o en un archivo Access de Microsoft.
- **Accesibilidad:** Si no pudiésemos acceder a nuestros datos existentes dentro de nuestra base de datos, la base de datos no tendría ninguna utilidad. La accesibilidad depende del formato en el cual se encuentra guardada la base de datos, y existen distintos formatos estándar conocidos, por ejemplo, dBase, Access, SQL Server, Oracle o MySQL.
- **Organización:** El modo más lógico para realizar una agenda es por días. Sin embargo, en la agenda escribiremos muchos datos duplicados, que pueden ocupar el doble de espacio o aún más. Los datos duplicados nos permiten usar más de un modo para buscar los datos. Podemos buscar en la agenda utilizando un apellido, y también podríamos realizar la búsqueda por una palabra determinada. La duplicación no es un problema cuando tratamos con bases de datos electrónica si éstas han sido implementadas.

das de la manera adecuada. Podemos utilizar índices y claves que brindan a nuestros clientes distintos modos para acceder a los datos sin necesidad de duplicarlos. Los índices y las claves sirven como punteros a los datos, lo que significa que son herramientas muy valiosas para evitar la duplicación de los datos.

- **Manipulación:** La edición de una agenda parece simple pero no lo es tanto. Podemos tachar una tarea que no realizaremos en el futuro o tachar una tarea ya realizada en el pasado. Pero si modificamos un dato en un sitio quizás no lo estemos haciendo en todas las partes en donde aparezca.

Cuando tratamos con nuestra base de datos, y si fue implementada correctamente, esta tarea simplemente se reduce a encontrar la información que queremos editar o eliminar, introducir los datos, y habremos terminado. Los datos se habrán cambiado o añadido en los sitios correctos.

Realicemos ahora un rápido resumen, éstos son los principales beneficios del uso de una base de datos:

- La localización y la manipulación de los datos es mucho más fácil cuando nuestros datos se encuentran en un único sitio.
- La organización de los datos es más simple cuando se encuentra en un sitio y se almacena electrónicamente.

Podemos acceder a los datos por medio de distintas aplicaciones y desde distintas ubicaciones cuando todos nuestros datos se encuentran en un único sitio; nuestra base de datos.

*En esta parte del curso aprenderemos los conceptos básicos que dan soporte a las bases de datos y aprenderemos a crear bases de datos y tablas.*

## Conceptos básicos sobre base de datos

Ya conocemos cómo es un archivo secuencial o plano, ya que todos hemos trabajado con documentos de texto. En su expresión más simple, un archivo plano es un registro largo de caracteres, con o sin caracteres de final de línea. En sus versiones más complejas, estos archivos pueden contener caracteres especiales para definir formateos y estilos.

Las ventajas de un archivo plano son varias: es fácil de escribir y ocupa un espacio mínimo, para ciertas aplicaciones son la mejor alternativa, por ejemplo, para generar copias de seguridad, históricos de datos, o registros de movimientos (log).

Pero, ¿qué sucede cuando se quiere recuperar o modificar un dato específico en un archivo plano? Debemos leer todo el archivo para encontrar el dato buscado y para modificarlo, normalmente, hay que reescribir todo el archivo. ¿Cuánto tiempo requiere esa operación cuando se trata de un archivo extenso, de, por ejemplo, 10.000 registros?

Aquí es donde comienzan las desventajas de los archivos planos de acceso secuencial y donde hacen su aparición otras técnicas de almacenamiento: archivos de acceso directo, archivos indexados y las bases de datos. Las bases de datos permiten recuperar y modificar la información más rápidamente.

¿En qué consiste una base de datos? Un archivo plano muy extenso, como podría ser el archivo de los datos de los clientes de un banco, se convierte en un conjunto de archivos compuesto por los propios datos y los índices que nos permiten acceder más rápidamente a la información.

Obviamente, el espacio ocupado por la información crece considerablemente para dar cabida a los diferentes índices: algún coste hay que pagar por la velocidad de acceso.

Pero hay más, los datos de las bases de datos no tienen la misma organización que tenían en el archivo plano. Los datos de las bases de datos se estructuran siguiendo técnicas de normalización de la información que hacen que el antiguo archivo plano de cuentas de los clientes de un banco dé lugar a una gran gama de nuevos archivos (denominados tablas) compuestos de campos.

Quizá al descomponer el antiguo archivo plano de cuentas de clientes veamos nacer diez o más tablas en los que repetiremos algunos campos. Esa redundancia de información es un mal necesario dentro del diseño de bases de datos. Es lo que nos permite vincular y estructurar la información.

Resumiendo, las ventajas del uso de las bases de datos son las siguientes:

- Consulta y actualización inmediata de la información, especialmente notable en grandes volúmenes de datos.
- Estandarización y automatización de las técnicas de acceso a los datos.

En la sección siguiente comenzaremos a entender el concepto de tabla.

## Tablas

Un modo lógico de organizar cualquier tipo de información, y no sólo dentro del ámbito de bases de datos, es agruparla por su tipo. Por ejemplo, el antiguo archivo plano de clientes podía tener los datos de la persona, los datos de las cuentas corrientes, los datos de los préstamos, etc. Un modo de clasificar esta información podría ser el siguiente:

- Datos de personas
- Datos de contratos
- Movimientos de contratos

Cabe señalar que en el diseño de bases de datos no hay una única solución. Muchas veces los datos se agrupan de una determinada manera para conseguir un determinado tiempo de acceso a la información.

Esta forma de clasificar nos permite definir las tablas de una base de datos, donde cada una almacena los datos sobre un objeto o entidad. Por ejemplo, la tabla Personas contendría la información personal sobre todos los clientes del banco (documento de identidad, nombres y apellidos, dirección, teléfono, fecha de nacimiento, etc.). Por su parte, la tabla de Contratos contendría la información sobre las cuentas de todos los clientes del banco (identificador de la cuenta, identificador del cliente, fecha de apertura de la cuenta, estado de la cuenta, etc.). La tabla Movimientos contendría toda la información sobre los movimientos de las cuentas de todos los clientes del banco (identificador de la cuenta, fecha del movimiento, tipo de movimiento, cantidad del movimiento, etc.). Por lo visto, cada tabla tiene su nombre que la identifica dentro de la base de datos. En este caso, la base de datos del banco tiene tres tablas: Personas, Contratos y Movimientos.

Los datos en la base de datos se almacenan utilizando un formato predefinido en la creación de la base datos. Cada una de las tablas de una base de datos tiene dos dimensiones: **filas y columnas**. Las filas están compuestas por todos los campos de un registro. Una columna está compuesta por todos los valores de un determinado campo. También se suele denominar **registro** a toda una fila y **campo** al valor de una columna. El conjunto de valores de una columna se denomina **dominio**. En estas tres figuras se visualiza un ejemplo del contenido de cada una de las tablas:

nombre de campo

fila →

columna ↑

valor de campo ↑

Tabla Personas				
NIF	Nombre	Dirección	Teléfono	
1234	José Pérez	avda. Libertad 4, Madrid	91 2230045	
12345	Juan Rodríguez	calle Montemar 5, Toledo	91 45660045	
45678	Luis García	calle Constitución 67, Valencia	977 45660045	
65778	Jaime Font	calle Italia 88, Gerona	975 45655042	

Tabla Contratos

ID. CTO.	NIF	Fecha apertura	Estado
44500	12345	12/02/2003	Activo
44501	12345	12/04/2003	Activo
44502	45678	22/04/2001	Cancelado
44503	65778	25/08/2001	Activo
44504	65778	30/03/2002	Activo

Tabla Movimientos

ID. CTO.	Fecha movimiento	Tipo	Cantidad
44500	1/2/2004	Debe	2000
44501	2/2/2004	Haber	1550
44503	13/3/2004	Debe	8500
44503	15/4/2004	Debe	6070
44503	16/4/2004	Haber	12090
44504	1/4/2004	Debe	9500

### Campos

Los campos o **columnas** son los atributos que describen el contenido de la tabla. En cada campo se almacena un tipo de dato definido, sea **texto**, **número**, **fecha**, **objectId**, etc. y a cada campo se le asigna un **nombre único** en cada tabla.

Cuando se realiza la definición de la tabla podemos declarar campos obligatorios y campos opcionales. Los campos obligatorios son aquellos que son imprescindibles para la existencia de la fila. Es decir, la fila no se puede insertar en la tabla si ese campo no tiene un valor. Los campos opcionales, cuando se omiten, pueden quedar vacíos o con un valor predeterminado. Por ejemplo, siguiendo el ejemplo de la sección anterior, en la tabla Personas podemos suponer que el NIF (número de identificación de la persona) puede ser un campo obligatorio y el campo Teléfono podría ser un campo opcional. Obviamente, esto depende totalmente del diseño de la base de datos, ya que en un sistema de pedidos telefónicos la situación podría ser exactamente al revés. El teléfono pasaría a ser un dato fundamental y el NIF puede perder importancia y ser un campo opcional.

## SGBD: Sistemas de gestión de base de datos

Un sistema de gestión de base de datos representa una infraestructura con la que se almacenan, recuperan y administran los datos de una base de datos. Esta infraestructura además debe ocuparse del mantenimiento de la integridad de los datos almacenados, es decir, controlar que los datos se guardan manteniendo unas condiciones establecidas de vinculación y cumpliendo las restricciones que se hayan definido. Por ejemplo, si al insertar una fila de la tabla Personas se debe informar un código de nacionalidad válido (que existe en una tabla de Nacionalidades), el sistema de gestión se preocupa por detectar un código de nacionalidad faltante (si es un campo obligatorio) o con un valor inválido (si se ha definido esa restricción) antes de insertar una fila en la Tabla de Personas.

---

*No se puede implementar ninguna aplicación con bases de datos utilizando un sistema de gestión de bases de datos que no sepa mantener la integridad de los datos.*

---

Los sistemas de gestión de base de datos se ocupan también de suministrar la conectividad con una amplia gama de lenguajes de programación y de garantizar la seguridad del acceso a los datos implementando el control de acceso en función a los permisos de cada usuario.

### Bases de datos relacionales

Hemos visto que una de las maneras más prácticas de representar datos es mediante el uso de tablas de dos dimensiones. Las tablas que componen una base de datos deben organizarse de manera que no se pierdan las relaciones existentes entre los datos, por ese motivo, este sistema de gestión de base de datos se denomina relacional. Como guía de diseño de una base de datos se deben seguir estas pautas:

- En cada fila de datos no deben aparecer grupos repetitivos. Si eso ocurriese, se debe generar una nueva entidad (tabla) para hacer desaparecer esa repetición.
- Todos los elementos de una determinada columna son todos del mismo tipo de dato y representan la misma clase de dato.
- Cada columna tiene su nombre, que es el nombre del campo.
- Todas las filas de una tabla son diferentes, no puede haber filas duplicadas.
- Cada fila debe tener un campo o conjunto de campos que sirva de clave de identificación.

### Tipos de relaciones

Entre las tablas de una base de datos se establecen tres tipos de relaciones que las vinculan:

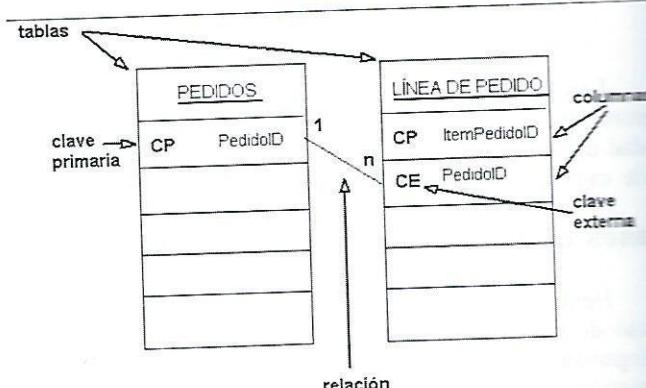
- **Uno a uno:** Un registro de una tabla sólo se vincula con un registro de otra.
- **Uno a muchos:** Un registro de una tabla se vincula con varios registros de otra tabla. Por ejemplo, una persona puede tener varios contratos (cuentas).
- **Muchos a muchos:** Varios registros de una tabla se vinculan con varios registros de otra tabla.

### Claves

En el diseño de las bases de datos relacionales las claves son importantes, porque se utilizan como identificadores de los registros y para clasificar las consultas. Existen dos tipos de claves: **claves primarias** y **claves externas**.

#### Clave primaria

La clave primaria de una tabla siempre es única, es decir, dentro de la tabla no pueden existir dos registros que contengan el mismo valor en el campo que está designado como clave primaria (CP). Esto significa que la clave primaria identifica de manera única a cada registro de la tabla. Esto también significa que un campo que forma parte de la clave primaria no puede contener Null. Una clave primaria que se genera con más de un campo se denomina clave primaria compuesta. Frecuentemente, la clave primaria es la clave de búsqueda, que ayuda a localizar un registro; es decir, se busca un valor del campo por la clave primaria. En términos de relaciones, la clave primaria siempre se utiliza en combinación con una clave externa (la analizaremos luego) en la tabla hija. En la figura siguiente se muestran las claves primarias y externas: la clave primaria de la tabla Pedidos está marcada como CP. Sólo puede haber una fila de la tabla Pedidos que se relaciona con cero o más filas de la tabla LineadePedido.



#### Clave externa

Una *clave externa* se utiliza en las relaciones como valor de búsqueda en la tabla hija. Esto significa que existe una correspondencia directa con el valor de la tabla primaria en la tabla padre, por lo que podemos utilizar el valor de la clave primaria para buscar un registro relacionado en la tabla hija. Se debe tener en cuenta que a diferencia de lo que sucede con la clave primaria, la clave externa puede contener valores nulos. No es algo recomendable, pero es posible. En una relación uno a uno, la clave externa debe ser única; en caso contrario puede contener duplicados.

En la figura de las claves se marca la clave externa de la tabla LíneadePedido con CE. En la tabla LíneadePedido puede haber muchas líneas que se relacionan con una fila de la tabla Pedidos.

## Índice

Los **índices** se utilizan para especificar campos de búsqueda que no son claves primarias ni externas. Al crearse un índice, será mucho más rápido buscar un determinado valor en un campo, cuando éste forma parte de un índice.

La creación de índices incrementa la carga de procesamiento y de accesos al disco en el momento de la actualización de la tabla. Pero si el campo elegido para ser indexado interviene en muchas consultas, las ventajas pueden superar a las desventajas.

La decisión de crear o no crear un índice es importante. En algunos casos será necesario eliminar un índice y hacer nuevas pruebas de rendimiento para comprobar si se gana o se pierde rendimiento por la implementación de ese índice.

### Los índices en la práctica

El uso de los índices exige un equilibrio entre velocidad de actualización y eficiencia de las consultas. Como regla general no es aconsejable una excesiva cantidad de índices ya que harán que el proceso de actualización sea más lento. Pero, si tenemos un caso de una base de datos que se consulta frecuentemente y se actualiza muy poco, podría tener sentido dotarla de varios índices para los campos más consultados. Evidentemente, los índices tendrán mayor importancia en las bases de datos más voluminosas donde los algoritmos de búsqueda deben gestionar cientos de miles de registros.

También se debe considerar la diferencia existente entre un índice de un campo único de un índice compuesto por varios campos. Los primeros son generalmente más eficientes, mientras que los últimos generalmente tienen mayores requerimientos de memoria. Sin embargo, éste podría no ser siempre el caso si se tiene en cuenta la cantidad de bytes por columna. Un índice compuesto por dos campos de cuatro bytes será más eficiente que otro de un única campo de cincuenta caracteres, además de minimizar la entrada/salida en disco. Dicho esto, se deduce fácilmente que es preferible indexar columnas que tengan pocos caracteres.

La elección de los campos sobre los que se construye un índice es una decisión importante dentro de la tarea de diseño de una base de datos. Como norma general deberíamos prestar atención a estos tipos de campos:

- Claves primarias
- Claves externas
- Campos que se usan en combinaciones de tablas (joins)
- Campos sobre los que se realizan búsquedas (Where)
- Campos sobre los que se realizan clasificaciones (Order By)

Esto no implica que todos los campos utilizados en una cláusula Where u Order By deben estar indexados. En general, los índices funcionan mejor sobre campos cuyos valores varían en las distintas filas. Por el contrario, un índice sobre un campo cuyos valores se repiten mucho a lo largo de la tabla pierde su utilidad. La clave primaria tiene un valor único para cada fila, por lo cual es un campo ideal, para ser indexado. En muchos sistemas de gestión de base de datos la clave primaria está indexada sin necesidad de especificarlo especialmente.

*La creación de nuevos índices no siempre implica mejores resultados. En tablas pequeñas el motor de gestión de bases de datos suele hacer caso omiso a la presencia del índice y lee toda la tabla. En esos casos, el mantenimiento del índice es una carga que no produce beneficios.*

## Integridad de datos

La **integridad de datos** significa que en la base de datos no hay datos obsoletos o incorrectamente relacionados debido a actualizaciones incompletas o fallidas. La integridad de datos se puede asegurar mediante la comprobación de uno o más de los siguientes mecanismos:

- Integridad de entidades
- Integridad referencial
- Integridad de dominio

### Integridad de entidad

La integridad de entidades declara que ningún valor de la clave primaria puede ser **nulo**. Esto es válido tanto para claves primarias compuestas (claves primarias que están compuestas por más de un campo) así como también para las claves primarias de un **único** campo. El valor especial **NULL** se refiere a un valor vacío o no-asignado. Como las claves primarias deben ser únicas, no se debe permitir que contengan valores nulos, y por ese motivo en el diseño de nuestra base de datos debemos aplicar la integridad de entidades.

### Integridad referencial

La integridad referencial nos asegura que ningún registro de una tabla hija tenga una clave externa que no exista como clave primaria en la tabla padre. Esto significa que cuando se fuerza la integridad referencial, no podemos añadir un registro a una tabla hija con una clave externa que no coincida con una clave primaria de la tabla padre.

El comportamiento aplicado en la integridad referencial es el que se presenta en la sección siguiente, dependiendo si se ha especificado restricciones o cascadas en la configuración de la integridad referencial. El concepto cascada se refiere al proceso por el cual las tablas relacionadas se actualizan cuando se actualiza o se elimina un registro de la tabla padre.

### Comportamiento de la integridad referencial

**Acción:** Eliminar un registro en la tabla padre para el cual existen registros relacionados en una tabla hija.

**Cascada:** Los registros relacionados en la tabla hija también se eliminan.

**Restricción:** No se permite la eliminación

**Acción:** Actualizar / cambiar la clave primaria de un registro en la tabla padre para el cual existen registros relacionados en una tabla hija.

**Cascada:** También se actualizan los registros relacionados en la tabla hija.

**Restricción:** No se permite la actualización / cambio.

La integridad referencial nos asegura que ningún registro queda huérfano en la tabla hija, es decir, que no tenga ningún registro relacionado en la tabla madre.

**Precaución:** cuando se emplean actualizaciones en cascada se debe ser precavido en el diseño. Una mala codificación puede destrozarnos la base de datos con una única sentencia SQL.

### **Integridad de dominio**

La integridad de dominio nos asegura que los valores de un campo determinado cumplen con la descripción de dominio correspondiente. La integridad de dominio se asegura por medio de las restricciones físicas y lógicas que se aplican a un campo específico. Cuando se pretenda añadir un valor inválido a un campo con limitaciones de dominio, el valor no se aceptará. Este tipo de limitaciones de dominio se podrían aplicar al campo de un número telefónico de la tabla Clientes, como se muestra a continuación:

- Limitación física: tipo de dato: cadena, longitud: 13
- Limitación lógica: formato de entrada: (999)9999999

### **Guía para la normalización de base de datos**

Ya conocemos los tipos de relaciones que podemos encontrar entre distintas tablas de una base de datos. Cuando debemos diseñar una base de datos suele plantearse el problema de no saber cómo agrupar los campos entre las distintas tablas y de no saber cuándo es el momento de dividir una tabla en dos o unir dos en una para optimizar el diseño.

El proceso de optimización de un diseño se denomina normalización de la base de datos. El objetivo de la normalización es lograr un diseño en el que no se produzcan inconsistencias ni errores de actualización. La normalización también nos ayuda a eliminar la redundancia, es decir, un dato que se almacena en dos o más sitios. La redundancia implica problemas de actualización (se modifica el dato en un sitio pero no en otro) que generan inconsistencias y problemas de mayores requerimientos de espacio de almacenamiento.

El proceso de normalización se realiza en varios niveles progresivos. Se considera que una base de datos tiene un diseño normalizado cuando se alcanza al menos el tercer nivel o tercera forma normal.

#### **Primera forma normal**

Una tabla está en esta forma si cada campo de la tabla sólo contiene un valor. Es decir, un campo no puede almacenar una matriz de valores.

#### **Segunda forma normal**

Una tabla está en la segunda forma normal si está en la primera forma normal y además todos los atributos no claves dependen de la clave principal. Esta anomalía se resuelve generando una tabla nueva con los campos que no dependen de la clave principal.

#### **Tercera forma normal**

Una tabla está en la tercera forma normal si y sólo si está en la segunda forma normal y además todos los atributos no claves dependen de manera no transitiva de la clave prin-

cipal. Esta anomalía se resuelve generando una tabla nueva con los campos que dependen transitivamente de la clave principal.

### Desnormalización

Hemos dicho que la normalización de un diseño nos lleva, entre otras cosas, a eliminar las redundancias. No obstante, no todo es blanco y negro y muchas veces optamos por cierto grado de redundancia controlada para lograr mejores prestaciones en los accesos a la información. Esto no es una crítica a la normalización, pero en la teoría llevada a la práctica suele presentar problemas de excesivo consumo de procesador o excesivas operaciones de entrada/salida para acceder a un determinado dato. En estos casos, una desnormalización controlada (redundancia) puede traer más beneficios que desventajas.

## PHP y los SGBD

PHP puede conectarse prácticamente con todos los principales sistemas de gestión de bases de datos y en diferentes entornos operativos. Entre los sistemas de bases de datos más empleados se pueden mencionar:

- **SQLite:** Es un motor de base de datos implementado en una biblioteca C que cumple con la mayor parte del estándar SQL92 y que se incluye de modo predeterminado con PHP 6. Es un motor ligero y veloz, en ciertas operaciones es dos veces más rápido que MySQL y PostgreSQL, pero no está diseñado para contener grandes base de datos.
- **MySQL:** Hasta antes de la versión 5 de PHP MySQL ha sido el sistema de base de datos de preferencia para su utilización con PHP, con el sistema operativo Linux y el servidor web Apache, conjunto que se denomina LAMP. Se destaca por su velocidad y fiabilidad. No obstante, a partir de PHP 5, MySQL modificó su política de licenciamiento y el motor de base de datos predeterminado es SQLite. MySQL mantiene una licencia dual: la licencia de código abierto (GNU/GPL) y la licencia comercial.
- **mSQL:** Es un sistema ligero de gestión base de datos relacionales que suministra las típicas funcioanlidades de bases de datos con escasos requerimientos de recursos (mSQL significa mSQL).
- **Microsoft SQL Server:** Es el sistema de gestión de base de datos profesional de Microsoft para las plataformas Windows.
- **Oracle:** Uno de los sistemas de gestión de bases de datos más populares y de mejor rendimiento.
- **PostgreSQL:** Antes denominada Postgres, es un sistema de gestión de base de datos de objetos relacionales de código abierto. Originado en la universidad de California de Berkeley especialmente diseñado para sistemas Unix, aunque ahora ya se puede instalar también en Windows.

---

*En esta parte del curso nos centraremos en MySQL, que será el tema de los capítulos siguientes. Se debe tener en cuenta que a nivel programación no existen grandes diferencias en el código de acceso y explotación de las bases de datos cuando se conoce bien un sistema determinado que cumple con el estándar SQL resulta muy sencillo adaptar el código para otro sistema que también cumpla con el estándar.*

---