Índice.

1. Herramientas proporcionadas por los SGBD.

Un **Sistema Gestor de Bases de Datos (SGBD)** es un sistema que permite la creación, gestión y administración de Bases de Datos, así como también la elección y el manejo de todas las estructuras necesarias para el almacenamiento y la búsqueda de información del modo más eficiente posible.

Actualmente hay un sinnúmero de SGBD y éstos se pueden clasificar de múltiples formas, pero una de ellas es a través de la forma en que se administran los datos:

- **Relacionales (SQL).**

  El modelo de bases de datos relaciones desde 1970 ha ido sufriendo múltiples transformaciones hasta convertirse en el más utilizado en la administración de bases de datos.

  Este modelo se basa en establecer **relaciones** (o vínculos) entre los datos, imaginando una tabla independiente para cada relación existente, con sus propios registros y atributos.

  Algunos de los principales Sistemas Gestores de Bases de Datos Relacionales son los siguientes:

  - **MySQL** → sistema gestor de bases de datos relacional por excelencia.
  - **MariaDB** → derivación de MySQL y que nace por la adquisición de MySQL por parte de Oracle.
  - **SQLite** → más que un SGBD es una biblioteca escrita en C que implementa un SGBD y permite transacciones sin necesidad de un servidor ni de más configuraciones.
  - **PostgreSQL** → orientado a objetos, publicado bajo licencia BSD.
  - **Microsoft SQL Server** → basado en el lenguaje Transact-SQL, con el que se pone a disposición de los usuarios de grandes cantidades de datos de forma simultánea.
  - **Oracle** → por excelencia en el mundo empresarial y considerado siempre como el más robusto y completo.

- **No relacionales (NoSQL).**

  Una base de datos es no relacional (NoSQL) es aquélla que cumple con las siguientes condiciones:

  - No requiere ni de estructuras de datos ni de tablas.
  - No garantiza completamente las características ACID.
  - Presenta datos muy bien escalados horizontalmente.

  Algunos de los principales Sistemas Gestores No Relacionales son los siguientes:

  - **MongoDB** → el más conocido y utilizado.
  - **Redis** → basado en el almacenamiento por clave-valor.
  - **Casandra** → utiliza el almacenamiento por clave-valor y es distribuido y masivamente escalable. Utilizado por Facebook, Twiter, Instagram, Spofity o Netflix.
  - **Azure Cosmos DB.**
  - **ObjectDB.**
  - **Apache CouchDB.**
  - **Google BigTable.**
  - **Amazon DynamoDB.**

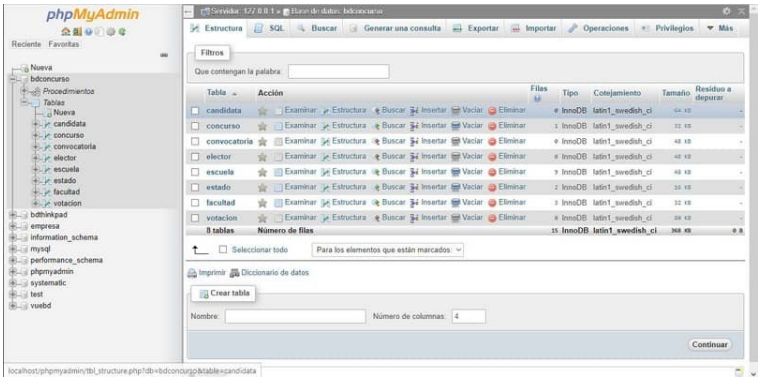
1.1. Herramientas gráficas.

Una herramienta gráfica permite la manipulación de una base de datos compleja de una forma muy sencilla, al disponer de una interfaz gráfica de usuario que ayude al Administrador de la Base de Datos (DBA) en el envío de comandos de administración de forma automática y sin necesidad de conocer toda su sintaxis.

Algunas de las herramientas gráficas de un SGBD son las siguientes:

- **PhpMyAdmin de MySQL.**

  PhpMyAdmin es la interfaz basada en páginas web de MySQL y trabaja a través de un servidor web que permite administrar datos de un servidor desde cualquier equipo de la red.



  PhpMyAdmin:

  - Dispone de opciones que permiten realizar cualquier operación realizable vía SQL
  - Gestiona bases de datos de un servidor, crea, borrar y modifica tables, lanza comandos SQL, exporta e importa información, recopila estadísticas, hace copias de seguridad, …
  - Dispone de un pequeño diseñador, tipo MySQL Workbench para gestionar relaciones de las tablas.

- **Oracle Enterprise Manager y Grid Control.**

  El SGBD Oracle dispone de dos herramientas gráficas con interfaz gráfica y montadas sobre un servidor web propietario de Oracle:

  - **Enterpise Manager** → manipula todas las funciones básicas de una base de datos. Incorporada directamente en el software de Oracle y configurada por el asistente de creación de bases de datos.
  - **Grid Control** → gestiona múltiples bases de datos en diversos servidores, permitiendo consultar el estado y rendimiento de cada de una de ellas. Se ha de instalar aparte del software de Oracle.

- **DB2 Data Studio.**

  Software que sustituirá a la herramienta llamada Control Center de DB2, y que permite manipular los objetos de las bases de datos DB2 e Informix.

  Soporta la administración avanzada de DB2 y simplifica la construcción de consultas SQL.

  La gran potencia radica es la creación de servicios Web que distribuyen los datos de las consultas SQL a las aplicaciones Cliente.

1.2. Herramientas textuales.

La principal utilidad de un SGBD radica en su intérprete de comandos, es decir, en su aplicación cliente cuya misión consiste en enviar comandos al SGBD y mostrar resultados por pantalla devueltos por el SGBD.

El cliente del servidor MySQL (mysql-server) es mysql, pero en Oracle se llama sqlplus y en DB2 se llama db2.

La forma de invocarlo desde un sistema operativo es desde un terminal, escribiendo su nombre y ciertas opciones.

La ejecución de comandos SQL se puede realizar de las siguientes formas:

- Escribir desde la consola el comando MySQL y aparecerá una línea precedida con mysql> desde la que invocar los comandos MySQL finalizados por ;

```
mysql> select now();
```

- Almacenar los comandos en un fichero de texto y mandarlos a ejecución mediante el comando source ubicación_fichero:

```
mysql> source  ejemplo.sql
```

- Ejecutar los comandos de un fichero de texto desde la shell:

```
mysql  -u root  -pPaswordUsuario  <ejemplo.sql
```

La ejecución desde MySQL consiste en localizar el fichero mysql.exe e invocarlo:

```
mysql  [options]  [database]
```

Options permite especificar una serie de parámetros de conexión:

--help	visualizar la ayuda
{-p | --pasword}[=frase]	password de conexión
{-P | --port}[=numero]	puerto TCPIP remoto al que se conecta
{-h | --host}[=numero]	nombre host o IP al que se conecta
{-u | --user}[=usuario]	usuario con el que se conecta
{-s | --socket}[nombre_fich]	fichero socket con el que se conecta

Database especifica sobre qué base de datos se ejecutarán los parámetros introducidos.

Algunos tipos de conexión son los siguientes:

- Conexión sin usuario ni password:

```
mysql
```

- Conexión con usuario y password:

```
mysql -u root -p
```

- Conexión con usuario y password en claro a una base de datos llamada Ejercicio:

```
mysql  -u root  -pPasswordUsuario ejercicio
```

- Conexión con usuario y password en claro a la base de datos Ejercicio del host 192.168.1.1:

```
mysql  -u root  -pPaswordUsuario  -h 192.168.1.1  ejercicio
```

- Conexión con usuario y password a la base de datos Ejercicio del host 192.168.1.1 con el puerto 14000:

```
mysql  -u root  -pPaswordUsuario  -h 192.168.1.1  ejercicio  -P 14000
```

2. Lenguaje de definición de datos.

El DDL es el sublenguaje de SQL que permite la definición de los datos a través de las siguientes funciones:

- Creación de bases de datos, tablas, índices y otros objetos (vistas, …).
- definición de estructuras físicas que contendrán los objetos de las bases de datos.

El DDL tiene tres instrucciones básicas:

- CREATE tipo_objeto Nombre Definición.
- DROP tipo_objeto Nombre;
- ALTER tipo_objeto Nombre Modificación.

```
Create Database Teatro;

Use Teatro;

Create table Actores(
        #Actor INTEGER PRIMARY KEY,
        Nombre VARCHAR(40),
        Fecha DATE,
        Nacionalidad VARCHAR(20)
);
```

3. Operaciones con Bases de Datos.

Las operaciones que se pueden realizar desde el DDL con las bases de datos son las siguientes:

- **Creación** → crea una tabla: CREATE DATABASE.

```
CREATE DATABASE    [IF NOT EXISTS]  nombre
            [especificación_create [, especificación_create] ...]
especificación_create:
            [DEFAULT] CHARACTER SET juego_caracteres
        |   [DEFAULT]  COLLATE nombre_colación
```

```
CREATE DATABASE ejemplo;
CREATE DATABASE ejemplo CHARACTER SET latin1 COLLATE latin1_spanish_ci;
```

- **Modificación** → cambia algún aspecto de una base de datos: ALTER DATABASE. En MySQL sólo se puede cambiar el juego de caracteres y su colación:

```
ALTER DATABASE      nombre
            [DEFAULT] CHARACTER SET juego_caracteres
        |   [DEFAULT]  COLLATE nombre_colación
```

```
ALTER DATABASE ejemplo CHARACTER SET latin1 COLLATE latin1_german1_ci;
```

- **Borrado** → elimina una base de datos: DROP DATABASE.

```
DROP DATABASE [IF EXISTS] nombre_base_datos;
```

```
DROP DATABASE ejemplo;
```

- **Consulta** → muestra las base de datos: SHOW DATABASES.

```
SHOW DATABASES;
```

- **Uso** → se carga la base de datos para trabajar: USE.

```
USE nombre_base_datos;
```

```
USE ejemplo;
```

El juego de caracteres de la base de datos puede ser utf8, latin1, latin2, …

La colación especifica cómo se va a tratar el alfabeto del juego de caracteres (ordenación y cómo se compararán los caracteres): si la ñ va después de la n.

4. Operaciones con Tablas.

Las operaciones que se pueden realizar desde el DDL con las tablas son las siguientes:

- **Creación** → crea una tabla: CREATE TABLE.

```
CREATE TABLE  nombre_tabla
            [definición_create [, definición_create] ...]
            [opciones_tabla]
definición_create:
            definición_columna
      |     [CONSTRAINT [símbolo]] PRIMARY KEY (nombre_columna, ...)
      |     [CONSTRAINT [símbolo]] FOREIGN KEY (nombre_columna, ...)
            [definición_referencia]
definición_columna:
            nombre_columna tipo_datos [NOT NULL | NULL]  [DEFAULT valor]  [UNIQUE [KEY] | [PRIMARY] KEY]
            [definición_referencia ]
definición_referencia:
            REFERENCES nombre_tabla [(nombre_columna, ...)]
                        [ON DELETE {CASCADE | SET NULL | NO ACTION} ]
                        [ON UPDATE {CASCADE | SET NULL | NO ACTION} ]
opciones_tabla:
            opción_tabla [opción_tabla] ...
opción_tabla:
            ENGINE = nombre_motor
      |     AUTO_INCREMENT = valor
      |     [DEFAULT] CHARACTER SET juego caracteres [collate colación]
      |     CHECKSUM = {0 | 1}
      |     COMMENT = 'string'
      |     MAX_ROWS = valor
      |     MIN_ROWS = valor
```

```
CREATE TABLE ejem(
      id  int  PRIMARY KEY,
      nombre varchar(20),
      direccion varchar(40)
);
```

- **Modificación** → cambia algún aspecto del contenido de una tabla: ALTER TABLE.

```
ALTER TABLE  nombre_tabla
            especificación_alter [, especificación_alter] ...
especificación_alter:
            ADD definición_columna [FIRST|AFTER nombre_columna]
      |     ADD (definición_columna, ...)
      |     ADD [CONSTRAINT [símbolo]]  PRIMARY KEY (nombre_columna, ...)
      |     ADD [CONSTRAINT [símbolo]]  UNIQUE (nombre_columna, ...)
      |     ADD [CONSTRAINT [símbolo]]  FOREIGN KEY (nombre_columna, ...)  [definición_referencia]
      |     CHANGE  [column]  nombre_anterior  definición_columna  [FIRST|AFTER nombre_columna]
      |     RENAME COLUMN  nombre_anterior  TO nuevo_nombre
      |     MODIFY definición_columna  [FIRST|AFTER nombre_columna]
      |     DROP COLUMN nombre_columna
      |     DROP PRIMARY KEY
      |     DROP FOREIGN KEY fk_símbolo
      |     opciones_tabla
```

```
ALTER TABLE nombre_tabla
            add fechaNacimiento DATE
;
```

- **Borrado** → elimina una tabla: DROP TABLE.

```
DROP TABLE nombre_tabla;
```

DROP TABLE ejem;

- **Renombrado** → reasigna un nombre a una tabla creada: RENAME TABLE.

```
RENAME TABLE nombre_tabla TO nuevo_nombre_tabla;
```

RENAME TABLE ejem TO ejemplo;

- **Consulta** → mostrar las tablas de una base de datos: SHOW TABLES.

```
SHOW TABLES;
```

- **Consultar la estructura** → mostrar la estructura de una tabla: DESCRIBE.

```
DESCRIBE nombre_tabla;
```

DESCRIBE ejem;

5. Tipos de datos.

Los tipos de datos que pueden usarse en MySQL son los siguientes:

- Numéricos.

Tipo	Naturaleza	Tamaño
tinyint [unsigned]	Entero	1 byte
smallint [unsigned]	Entero	2 bytes
mediumint [unsigned]	Entero	3 bytes
int [unsigned]	Entero	4 bytes
integer [unsigned]	Entero	4 bytes
bigint [unsigned]	Entero	8 bytes
float [unsigned]	Real aproximado	4 bytes
double [unsigned]	Real aproximado	8 bytes
decimal(longitud, decimales)	Real exacto	Variable
numeric(longitud, decimales)	Real exacto	Variable

- String.

Tipo	Naturaleza	Tamaño
char(longitud)	Caracteres	Longitud fija
varchar(longitud)	Caracteres	Longitud variable
tinyblob	Objetos binarios	Hasta 255 caracteres
blob	Objetos binarios	Hasta 65.535 caracteres
mediumblob	Objetos binarios	Hasta 16.777.215 caracteres
longblob	Objetos binarios	Hasta 4.294.967.298 caracteres
tinytext	Texto plano	Hasta 255 caracteres
text	Texto plano	Hasta 65.535 caracteres
mediumtext	Texto plano	Hasta 16.777.215 caracteres
longtext	Texto plano	Hasta 4.294.967.298 caracteres
set(valor1,valor2,...)	Conjuntos	Conjuntos de valores
enum(valor1, valor2, ...)	Enumeraciones	Lista de valores

- De fecha.

Tipo	Naturaleza	Tamaño
date	Fecha	‘aaaa-mm-dd’
time	Hora	‘hh:mm:ss’
timestamp	Fecha y hora	‘aaaa-mm-dd hh:mm:ss’
datetime	Fecha y hora	‘aaaa-mm-dd hh:mm:ss’
year	Año	‘aaaa’

6. Restricciones.

Las restricciones (o constraints) en SQL definen condiciones o reglas que han de cumplir los elementos de las tablas, evitando la inserción de datos incorrectos y garantizando que el tipo de datos sea el correcto.

Las restricciones se pueden aplicar a una columna o a toda una tabla.

Las restricciones son las siguientes:

- NOT NULL → para que la columna tenga valores nulos.
- UNIQUE → todos los valores de la columna son distintos.
- PRIMARY KEY → las restricciones NOT NULL y UNIQUE permiten identificar de forma unívoca cada fila de la tabla.
- FOREIGN KEY → permite detectar o identificar de forma unívoca una fila o registro de otra tabla.
- CHECK → permite asegurar que todos los valores cumplen una condición.
- DEFAULT → valor concreto que tendrá una columna si no se especifica otro.
- INDEX → útil para recuperar y crear datos de forma rápida en la base de datos.

6.1. Restricción NOT NULL

La restricción NOT NULL asegura que todos los valores de una columna NO pueden tomar valor nulo.

Si queremos reflejar que el nombre de los socios NO puede ser nulo:

```
Create Table Socio(
    ID int,
    Nombre varchar(20) NOT NULL,
    Apellido varchar(30)
);
```

6.2. Restricción UNIQUE

La restricción UNIQUE asegura que todos los valores de una columna son diferentes.

Si queremos reflejar que el nombre de los socios NO se puede repetir:

```
Create Table Socio(
    ID int,
    Nombre varchar(20) UNIQUE,
    Apellido varchar(30)
);
```

```
Create Table Socio(
    ID int,
    Nombre varchar(20),
    Apellido varchar(30),
    Constraint Unique( Nombre )
);
```

6.3. Restricción PRIMARY KEY

La restricción de clave primaria (**Primary Key Constraint**) identifica de forma unívoca cada tupla de una tabla. Esta restricción IMPLICA AUTOMÁTICAMENTE una restricción UNIQUE

Los valores de las claves primarias deben contener **valores únicos** y no pueden tener valor **NULL**.

Si queremos reflejar que el ID de los socios es la clave primaria:

```
Create Table Socio(
    ID int PRIMARY KEY,
    Nombre varchar(20),
    Apellido varchar(30)
);
```

```
Create Table Socio(
    ID int,
    Nombre varchar(20),
    Apellido varchar(30),
    Primary Key( ID )
);
```

6.4. Restricción FOREIGN KEY

La restricción de clave externa (Foreign Key) es para establecer un vínculo entre dos tablas, siendo un campo (o colección de campos) en la tabla que hace referencia a la clave primaria de la otra tabla:

- La tabla con la clave externa se denomina **tabla secundaria**.
- La tabla con la clave candidata se denomina **tabla principal o referenciada**.

Si queremos reflejar que el nombre de los Socios es la clave foránea (o de enlace) con otra tabla llamada Datos a través de su campo Nombre:

```
Create Table Datos(
    Nombre   int  PRIMARY KEY,
    Tipo     varchar(20)
);

Create Table Socio(
    ID int PRIMARY KEY,
    Nombre varchar(20) FOREIGN KEY
                    References Datos(Nombre),
    Apellido varchar(30)
);
```

```
Create Table Socio(
    ID int,
    Nombre varchar(20),
    Apellido varchar(30),
    Primary Key( ID ),
    Foreign Key( Nombre) references Datos( Nombre )
);
```

6.5. Restricción CHECK

La restricción CHECK se usa para limitar el rango de valor de una columna:

- Si se define en una única columna → restringe los valores de esa columna.
- Si se define en una tabla → restringe los valores de todas las columnas.

Si queremos reflejar que el nombre de los socios no puede quedar vacío:

```
Create Table Socio(
    ID int,
    Nombre varchar(20),
    Apellido varchar(30),
    check ( Nombre != '')
);
```

6.6. Restricción DEFAULT

La restricción Default se usa para proporcionar un valor por defecto.

Si queremos reflejar que el nombre de los socios va a tomar AA ese valor por defecto:

```
Create Table Socio(
    ID int DEFAULT 'AA',
    Nombre varchar(20),
    Apellido varchar(30)
);
```

6.7. Restricción INDEX

La restricción INDEX sirve para crear índices en las tablas.

Si queremos crear un nuevo índice en la tabla Socio sobre el campo Nombre:

```
Create Table Socio(
    ID int,
    Nombre varchar(20),
    Apellido varchar(30)
);

Create Index indNombreSocio ON Socio(Nombre);
```

7. Diccionario de datos.

El Diccionario de Datos es un listado organizado de todos los datos pertenecientes al sistema (en nuestro caso, la base de datos), organizado con todos los datos y elementos que aparecen en los diagramas (Diagrama Entidad/Relación y Diagrama Relacional), con definiciones precisas y rigurosas para que tanto el usuario como el analista tengan un entendimiento común de entradas, salidas, componentes de los almacenes y cálculos intermedios.

El Diccionario de Datos contiene las características lógicas de los sitios en los que se almacenan los datos del sistema, incluyendo:

- Nombre.
- Descripción.
- Alias.
- Contenido.
- Organización
- Procesos que emplean dichos datos.
- Sitios que necesitan un acceso inmediato a la información.

7.1. Objetivo

El objetivo de un Diccionario de Datos consiste en dar precisión sobre los datos que se manejan en el sistema, evitando malas interpretaciones o ambigüedades.

7.2. Utilidad

La utilidad del Diccionario de Datos es porque permite:

- Validar la integridad y la exactitud de los diagramas entidad/relación y relacional.
- Proporcionar un punto de partida para el desarrollo de pantallas e informes.
- Determinar el contenido de los datos almacenados en las relaciones o en los archivos.
- Ayudar en todo el proceso de desarrollo de la base de datos o de aplicaciones.
- Coordinar la actividad de la base de datos.
- Desarrollar la lógica de procedimientos y funciones en la programación de bases de datos.

7.3. Ventajas

Las ventajas de la utilización del Diccionario de Datos son las siguientes:

- Contiene un listado de todos los objetos que forman parte del sistema.
- Facilita el manejo de los detalles en los grandes sistemas, permitiendo mayor visibilidad de todos los objetos.
- Facilita la ubicación de errores y también las omisiones durante el proceso de diseño.
- Todos los encargados de la base de datos tendrán un conocimiento universal estandarizado, facilitando la comunicación en el grupo de trabajo.
- Ayudan a encontrar respuesta a preguntas de los analistas tales como cuántos caracteres tiene un determinado dato, qué otros nombres recibe este dato, dónde se utiliza un determinado dato, etc.

7.4. Elementos

Los elementos del Diccionario de Datos son los siguientes:

- **Datos elementales** → parte más pequeña de los datos con significado en el sistema de información que, combinados con otros elementos de datos, proporciona una estructura de datos más compleja que provea la información completa que se desea guardar y consultar, y cuyos componente son:

- Nombre de los Datos → distingue un dato de otro dato y debe ser significativo con respecto al objeto (o entidad o relación) de la base de datos con el fin de tener un mayor control de la información.
- Descripción de los Datos → describe brevemente lo que representa el dato en el sistema. Las descripciones se deben realizar teniendo en cuenta que serán leídas por gente que no conozcan nada del sistema.
- Alias → distintos nombres que puede recibir un dato en función de quién y cuál sea su uso.
- Longitud de campo → cantidad de espacio que ocupa cada dato, normalmente en bytes.
- Valores específicos → valores de un campo restringidos a un intervalo específico.

- **Estructura de datos** → agrupación de datos (elementales o no) dotada de significado para describir un componente del sistema y cuya construcción se puede realizar a través de las siguientes combinaciones:
  - Relación secuencial → define los componentes que se incluyen en la estructura de datos.
  - Relación de selección → define la alternativas para datos (o estructuras de datos).
  - Relación de iteración → define la repetición de un componente.
  - Relación opcional → establece si el dato puede o no estar incluido (una o ninguna iteración).

7.5. Notación

Hay varias propuestas para la notación que se utiliza en el Diccionario de Datos, pero la más común y que emplea un conjunto reducido y simple de símbolos es la siguiente:

Símbolo	Significado
=	Significa «**está compuesto de**», o «**es definido como**», o «**esta hecho de**»
+	Significa «**y**»
( )	Significa que el ítem entre paréntesis es **opcional** (puede estar presente o ausente)
{ }	Significa cero o más de cualquier cosa que este dentro de las llaves, i.e. **repetición, iteración**
[|]	Significa que **uno** de los atributos entre las barras esta presente. Selecciona una de varias alternativas, separa opciones alternativas en la construcción.
* *	Incluye **comentario** – define el significado de datos, informalmente
@	identificador (campo clave) para un almacén
‘ ‘	Incluye **literales** (valor a utilizar)

Un ejemplo en la utilización es el siguiente:

CLIENTE := { cliente } * *el archivo de Clientes* *
cliente := @nro_cliente + nombre_cliente + dirección_para_remito + crédito
nro_cliente := * *identificador interno de un cliente, campo clave del depósito CLIENTES* * * [1 ... 999]* * *un número entre 1 y 999* *
crédito := [Positivo | Negativo]
nombre_cliente := título_de_cortesía + primer_nombre + (nombre-intermedio) + apellido
título_de_cortesía := [Sr. | Srta. | Sra. | Dr. | Prof. | Don | Doña]
primer_nombre := 1{ caracter_válido }30
nombre_intermedio := 1{ caracter_válido }30
apellido := 1{ caracter_válido }30
carácter_válido := [letra | dígito | ' | - |]
dígito := [0 | 1 | 2 | 3 | 4| 5| 6 | 7 | 8 | 9]
letra := [letra_en_mayúscula | letra_en_minúscula] * *[A ... Z | a ... z]**
dirección_para_remito := calle + número_dir + (departamento) + (localidad) **si la localidad no se detalla, la dirección es de Tandil**
calle := {carácter_valido}
número_dir := {dígito}
localidad := [Tandil | Villa Cacique | Barker | Juárez | Lobería | Posadas] * *localidades en las que se entregan pedidos* *
pedido := nro_cliente + nombre_cliente + dirección_para_remito + *1* {item_pedido} *10 * un pedido puede contener hasta 10 items* *
item_pedido := código_artículo + nombre_artículo + cantidad
código_artículo := 1 {dígito} 3 * identificador interno de un artículo, un número de hasta tres dígitos *