

práctica 2 sistemas de información

Diseño, desarrollo e implementación de un sistema de información web



2 de diciembre de 2020

roberto jiménez y álvaro fraidias

Contenido

[1. Objetivos 2](#_Toc63183320)

[2. Servidor web 2](#_Toc63183321)

[2.1 Configuración del entorno de trabajo. 2](#_Toc63183322)

[3. Base de datos 4](#_Toc63183323)

[3.1 Sistema Gestor de Bases de Datos 4](#_Toc63183324)

[3.1.1 PostgreSQL 4](#_Toc63183325)

[3.2 Diseño conceptual 5](#_Toc63183326)

[3.3 Diseño lógico 6](#_Toc63183327)

[3.4 Tablas 7](#_Toc63183328)

[4. Conclusión 8](#_Toc63183329)

[5. Bibliografía 8](#_Toc63183330)



# 1. Objetivos

El objetivo de esta práctica es crear una base de datos que almacene de forma persistente los datos que el usuario nos da a la hora de navegar por la página web desarrollada en la práctica anterior. Para realizar la práctica la hemos dividido en 3 partes:

1º Instalar y configurar el sistema gestor de bases de datos que utilizaremos como entorno de trabajo. En nuestro caso hemos usado el SGBD PostgreSQL con la interfaz gráfica Pgadmin.

2º Diseñar el modelo Entidad-Relación de la base de datos.

Para ello primero hemos construido el diseño conceptual, después el diseño lógico y por último la implementación.

3º Aunque en esta práctica no hacía falta, hemos visto necesario cambiar el servidor web. Anteriormente habíamos usado Apache Tomcat y en esta práctica hemos usado el framework Django escrito en Pyhton.

# 2. Servidor web

Todas las páginas que vemos en nuestro navegador se deben alojar en un servidor. En nuestro caso hemos utilizado el framework Djando escrito en Python para alojar la página y tratar con los datos que el usuario inserta al navegar por la página web.



Ilustración : Logo Django

## 2.1 Configuración del entorno de trabajo.

Para instalar django primero debemos tener en nuestro equipo Python. La instalación de Django es muy sencilla:

Simplemente con poner el siguiente comando en una terminal se nos instalará en nuestro equipo:

pip install Django==3.1.3

Una vez instalado, no cerramos el terminal ya que nos hará falta posteriormente.

Con el siguiente comando nos creará un proyecto:

Django-admin startproject + nombre del archivo

Una vez ejecutado el comando anterior nos creará un proyecto con los siguientes archivos:



Ilustración : Archivos creados al ejecutar Django

\***\_\_init\_\_.py**: Es un fichero vacío creado aquí para que Django/Python reconozca la carpeta como un [Paquete Python](https://docs.python.org/3/tutorial/modules.html#packages) y te permita usar sus objetos dentro de otras partes del proyecto.

\***settings.py**:  contiene todos los ajustes del sitio. Es donde registramos todas las aplicaciones que creamos, la localización de nuestros ficheros estáticos, los detalles de configuración de la base de datos, etc.

\***urls.py**: Este archivo es una especie e índice o tabla de contenido con las urls que necesitaremos para el correcto funcionamiento de la página.

\***wsgi.py**: se usa para ayudar a la aplicación Django a comunicarse con el servidor web.

El script **manage.py** se usa para crear aplicaciones, trabajar con bases de datos y empezar el desarrollo del servidor web.

Una vez creado el proyecto, para lanzar el servidor basta con poner en la terminal el siguiente comando:

Python manage.py runserver

A partir de aquí ya se estaría ejecutando correctamente el servidor.

# 3. Base de datos

La base de datos que hemos utilizado para almacenar los datos introducidos por los usuarios es PostgreSQL con la interfaz gráfica Pgadmin.

## 3.1 Sistema Gestor de Bases de Datos

Un sistema gestor de base de datos es un conjunto de programas que permiten el almacenamiento, modificación y extracción de la información en una base de datos.

Para esta página hemos utilizado PostgreSQL, en específico Pgadmin para crear la base de datos de forma gráfica.

## 3.1.1 PostgreSQL

 PostgreSQL es un [sistema de gestión de bases de datos](https://es.wikipedia.org/wiki/Sistema_de_gesti%C3%B3n_de_bases_de_datos) [relacional](https://es.wikipedia.org/wiki/Modelo_relacional) [orientado a objetos](https://es.wikipedia.org/wiki/Base_de_datos_orientada_a_objetos) y de [código abierto](https://es.wikipedia.org/wiki/Software_de_c%C3%B3digo_abierto). Como características podemos observar una alta concurrencia y una amplia variedad de tipos nativos (número de precisión arbitraria, texto de largo ilimitado).



Ilustración : Logo PostgreSQL

A la hora de elegir un SGBD hemos escogido PostgreSQL por dos motivos:

* Su facilidad de usar e implementar la base de datos ya que lo podemos hacer de forma gráfica.
* Es la base de datos de código abierto más avanzada del mundo.

## 3.2 Diseño conceptual

Para el diseño conceptual hemos utilizado un programa gratuito llamado “DER editor” utilizado anteriormente en la asignatura “Bases de datos” del segundo curso de ingeniería informática.

El diseño conceptual que hemos realizado es el siguiente:

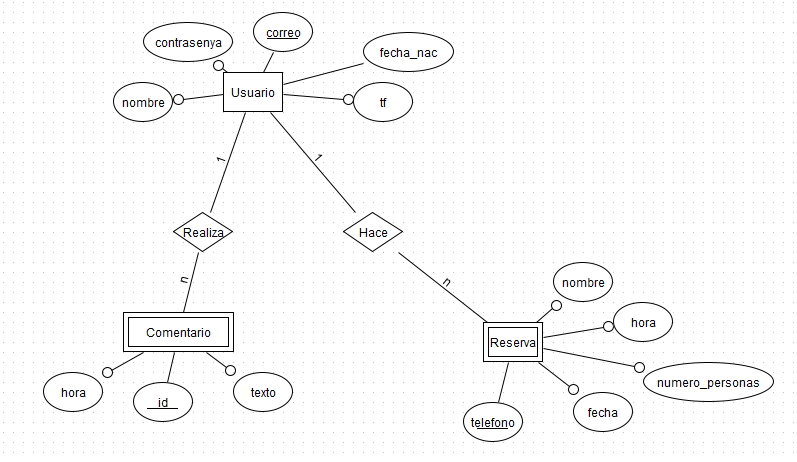


Ilustración :Diseño lógico de la base de datos

En este diseño se puede ver tres entidades y dos relaciones:

\***Entidad usuario**: Esta entidad representa como su propio nombre indica al usuario.

Como atributos tenemos el correo que se usará como clave primaria, diferenciándolo de los demás usuarios, la contraseña, el nombre y el teléfono que obligatoriamente el usuario deberá rellenar y la fecha de nacimiento.

\***Entidad Blog**: Esta entidad representa el apartado de la página relacionado con el blog.

Tiene una clave primaria llamada id y un atributo llamado comentario que obligatoriamente deberá rellenar ya que no tiene sentido enviar un mensaje vacío.

\***Entidad Reserva**: Esta entidad abarca todo lo que conlleva reservar una mesa en nuestro restaurante:

Para reservar una mesa el usuario debe poner un número de teléfono que lo usaremos como clave primaria. También necesariamente el usuario deberá introducir el número de personas que van a asistir a dicha comida-cena, el nombre, la fecha y la hora.

\***Relación Hace**: Relaciona la entidad Usuario con la entidad Reserva. Un usuario puede hacer n reservas por lo que esta relación es 1:n. Esta relación no tiene atributos

\***Relación Realiza**: Relaciona la entidad Usuario con la entidad Comentario. Un usuario puede hacer n reservas por lo que esta relación es 1:n. Esta relación no tiene atributos

## 3.3 Diseño lógico

El diseño lógico es el paso siguiente al diseño conceptual y el previo a la implementación. El diseño lógico de nuestra base de datos es el siguiente:

**Entidad Usuario:**

Usuario (correo CHAR, tf:INT, nombre:CHAR, contrasenya:CHAR, fecha\_nac:CHAR )

Clave Primaria { correo }

Valor No Nulo { tf }

Valor No Nulo { nombre }

Valor No Nulo { contrasenya }

**Entidad Comentario:**

Comentario (id null, correo CHAR, hora:null, texto:char )

Clave Primaria { id, correo }

Valor No Nulo { hora }

Valor No Nulo { texto }

Clave Ajena { correo } hace referencia a Realiza

**Relacion Realiza:**

Realiza (correo CHAR, id null, correo CHAR )

Clave Primaria { id, correo }

Valor No Nulo { correo }

Clave Ajena { correo } hace referencia a Usuario

Clave Ajena { id, correo } hace referencia a Comentario

**Entidad Reserva:**

Reserva (telefono null, hora:null, fecha:null, nombre:null, numero\_personas:null )

Clave Primaria { telefono }

Valor No Nulo { hora }

Valor No Nulo { fecha }

Valor No Nulo { nombre }

Valor No Nulo { numero\_personas }

**Relacion Hace:**

Hace (correo CHAR, telefono null )

Clave Primaria { telefono }

Valor No Nulo { correo }

Clave Ajena { correo } hace referencia a Usuario

Clave Ajena { telefono } hace referencia a Reserva

## 3.4 Tablas

Las tablas nos facilitan entender de forma visual cada entidad para poder implementarlas en la base de datos de forma correcta:

**-----------------Tabla usuario------------------**

CREATE TABLE usuario (

Correo VARCHAR (30) NOT NULL,

  nombre VARCHAR (20) NOT NULL,

contrasenya VARCHAR (20) NOT NULL,

fecha\_nac VARCHAR (20),

tlf INTEGER (9) NOT NULL,

CONSTRAINT usuarioPK PRIMARY KEY (correo)).

**------------------Tabla Comentario-----------------------**

CREATE TABLE comentario (

id BIGINT NOT NULL AUTO\_INCREMENT,

  texto VARCHAR (20) NOT NULL,

usuario VARCHAR (20),

CONSTRAINT textoPK PRIMARY KEY (textoId)

CONSTRAINT usuarioFK FOREIGN KEY (usuario)

CONSTRAINT comentarioPK PRIMARY KEY (id))

REFERENCES usuario(login));

**-------------------Tabla Reserva--------------------**

CREATE TABLE Reserva (

telefono INTEGER (9) NOT NULL,

hora INTEGER (4) NOT NULL,

fecha INTEGER (4) NOT NULL,

nombre VARCHAR (20) NOT NULL,

num\_personas INTEGER (2) NOT NULL,

CONSTRAINT telefonoPK PRIMARY KEY (telefono),

REFERENCES usuario(login));

# 4. Conclusión

Al realizar esta práctica nos hemos dado cuenta de la importancia que tienen los servidores y las bases de datos que hay detrás de una página web:

-Sin un servidor donde alojar la página web nos sería imposible interactuar con dicha página desde un dispositivo distinto al usado como servidor local.

-Un correcto enlace con la base de datos nos facilita enormemente el mantenimiento de la página y nos aumentan exponencialmente las funcionalidades que puede abarcar dicha página.

# 5. Bibliografía

- <https://es.wikipedia.org/wiki/PostgreSQL>

- <https://www.djangoproject.com/>

- <https://plantillashtmlgratis.com/>

- <https://es.wikipedia.org/wiki/HTML>

- <https://es.python.org/>

- <https://es.wikipedia.org/wiki/Sistema_de_gesti%C3%B3n_de_bases_de_datos>