

ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Manual del Programador

Sitio Web eBBBBOOKS

www.ebbbooks.com

Autor	Versión	Fecha
María José, Burgos Ivonne, Moreano Irina, Robalino Edwin	V. 1.0	15/09/2014

MANUAL DEL PROGRAMADOR

Tabla de Contenidos

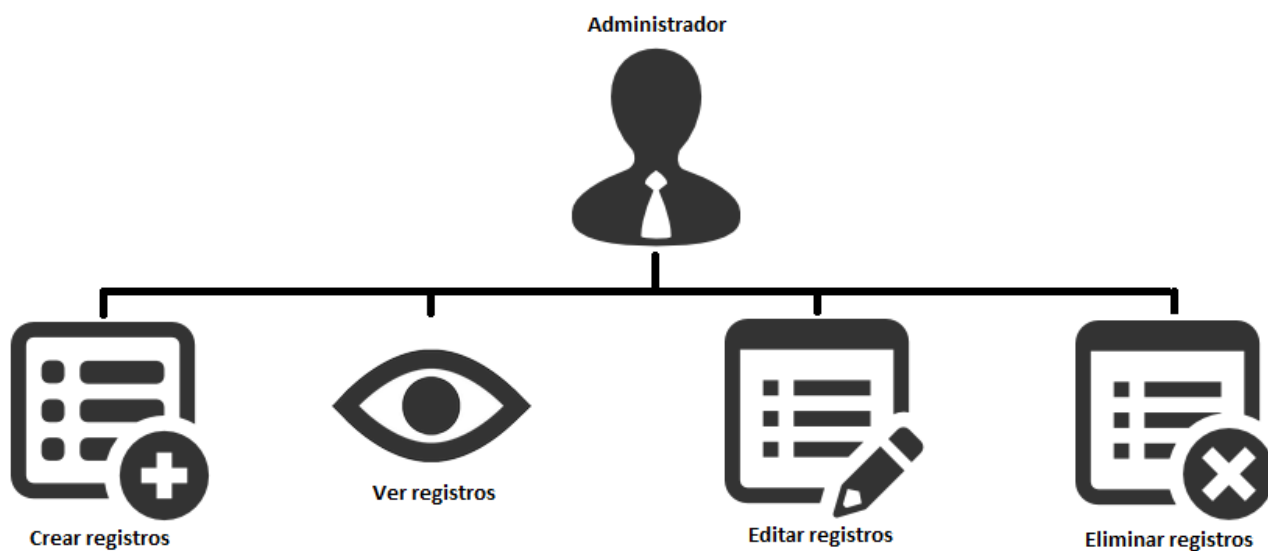
1. Introducción	2
1.1 Propósito del documento.....	2
2. Modelo de Base de Datos	3
2.1 Modelo Entidad-Relación.....	3
2.2 Diccionario de Datos	4
3. Instrucciones de Uso	7
3.1 Navegación de la Administración.....	7
3.2 Uso de archivos y funciones.....	9
4. Estadísticas del Sitio Web.....	28
4.1 Google Analytics.....	28
4.1.1 ¿Cómo se implementa?	28
5. Instalación del Sistema.....	29
5.1 Requerimientos de Software	29
5.1.1 Servidor Web.....	29
5.1.2 Base de Datos.....	29
5.1.3 Lenguaje de Programación.....	29
5.2 Instalación y configuración de paquetes necesarios	29
5.2.1 Instalación en Ubuntu V. 14.04.1. LTS	29
5.2.1.1 Actualización de repositorios	29
5.2.1.2 Instalación de librerías para PHP	30
5.2.2 Instalación en Windows 7 y 8.....	30
5.2.2.1 Descarga de Appserv	30
5.2.2.2 Instalación de Appserv	31
5.2.2.3 Habilitar la extensión PDO.....	34
5.2.2.4 Habilitar el motor InnoDB	34
5.2.2.5 Instalar la base de datos ebbbooks	34
5.3 Servicio Web Apache.....	37
5.3.1 Instalación en Ubuntu V. 14.04.1 LTS	37
5.4 Servicio de Base de Datos MySQL.....	37
5.4.1 Instalación en Ubuntu V. 10.04.1 LTS	37
5.4.1.1 Instalación de MySQL Workbench.....	38
5.5 Librerías para la comunicación de PHP con MySQL	41

1. Introducción

1.1 Propósito del documento

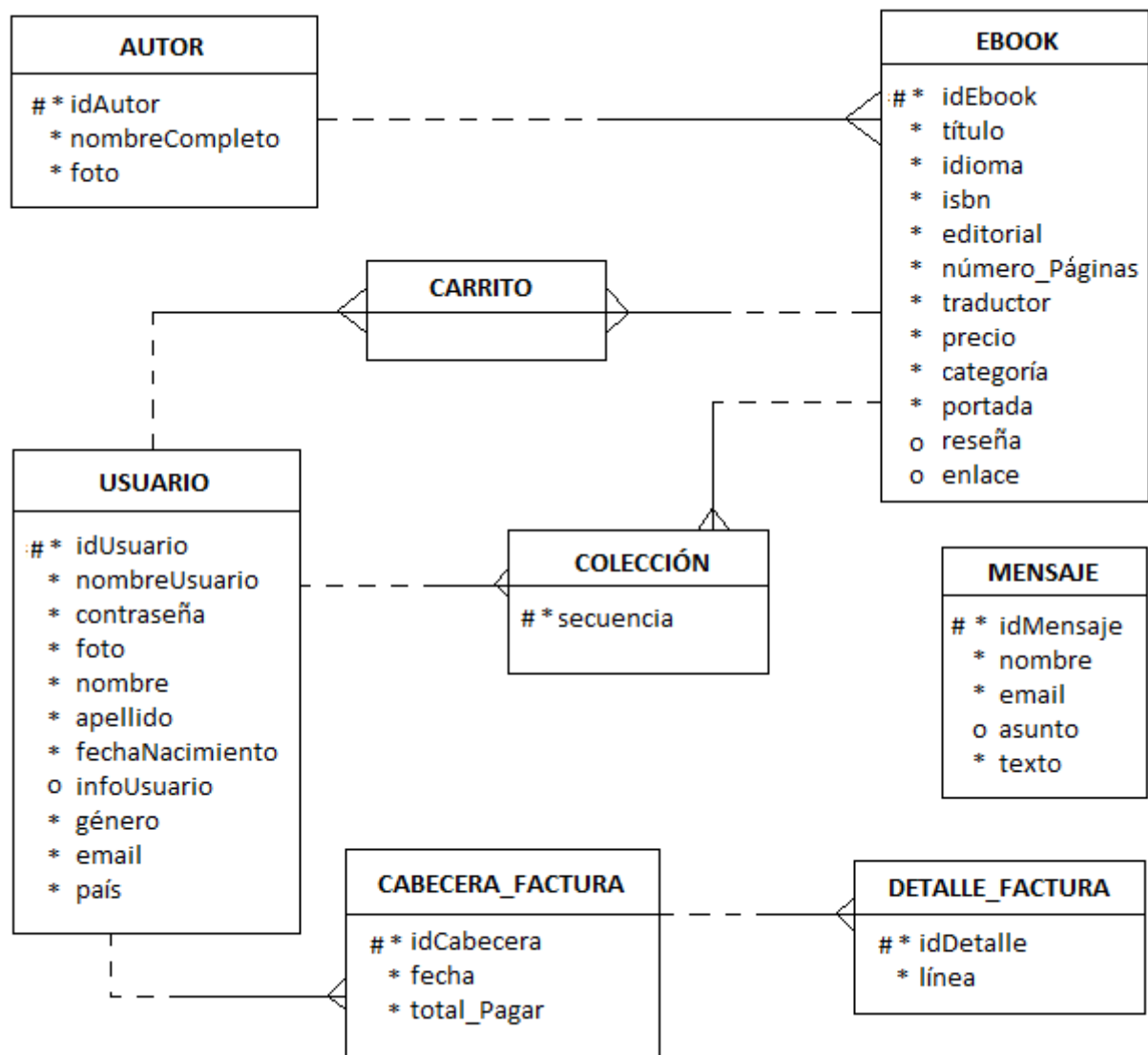
El documento a presentar está destinado a explicar el uso, contenido y funcionamiento de los archivos y páginas del sitio web para la empresa eBBBOOKS. Tiene como objetivo describir las operaciones y funciones disponibles para el administrador, junto con el proceso de instalación de los servicios necesarios para su correcto funcionamiento de manera que facilite el uso a este.

El siguiente esquema explica las funcionalidades con las que contará el administrador al momento de navegar por el sitio:



2. Modelo de Base de Datos

2.1 Modelo Entidad-Relación



2.2 Diccionario de Datos

Tabla: EBOOK

Campo	Tamaño	Tipo de Dato	Descripción
idEbook	11	Entero	Clave única para cada ebook registrado.
Título	11	Entero	Palabra o frase con que se da a conocer el nombre o asunto de una obra o de cada una de las partes o divisiones de un escrito.
Autor	45	Varchar	Persona que ha hecho alguna obra científica, literaria o artística.
Idioma	45	Varchar	Lengua de un pueblo o nación, o común a varios.
Isbn	45	Varchar	Sistema internacional de numeración de libros para su fácil y correcta identificación.
Editorial	45	Varchar	Casa editora.
número_Páginas	4	Entero	Cantidad de hojas de un escrito o manuscrito.
Traductor	45	Varchar	Que traduce una obra o escrito.
Precio	-	Doble	Valor pecuniario (relativo al dinero efectivo) en que se estima algo.
Categoría	45	Varchar	Uno de los diferentes elementos de clasificación que suelen emplearse.
Portada	50	Varchar	Cubierta delantera de un libro o de cualquier otra publicación o escrito.
Reseña	-	Texto	Narración breve de una obra literaria o científica.
Enlace	50	Varchar	Referencia en un documento de hipertexto a otro documento o recurso.
Relaciones: Autor, Colección, Carrito.			Campos Clave: idEbook. Claves foráneas: Autor.

Tabla: CARRITO

Campo	Tamaño	Tipo de Dato	Descripción
Usuario	11	Entero	Clave foránea de un usuario.
Ebook	11	Entero	Clave foránea de un ebook.
Relaciones:			Campos clave: Claves foráneas: fk_usuario y fk_ebook

Tabla: CABECERA_FACTURA

Campo	Tamaño	Tipo de Dato	Descripción
id_Cabecera	11	Entero	Clave única para cada cabecera de factura registrada.
usuario	11	Entero	Clave foránea de un usuario.
Fecha	-	Date	Tiempo en que ocurre o se hace algo.
total_pagar	-	Double	Valor total correspondiente a los artículos adquiridos.
Relaciones: Detalle_Factura			Campos clave: id_cabecera Claves foráneas: fk_usuario

Tabla: DETALLE_FACTURA

Campo	Tamaño	Tipo de Dato	Descripción
id_detalle	11	Entero	Clave única para cada detalle de factura registrado.
cabeceraFactura	11	Entero	Clave foránea de la cabecera de factura a la cual corresponde este detalle.
Línea	11	Entero	Número del objeto o servicio adquirido.
código_ebook	11	Entero	Clave foránea de un ebook.
Relaciones:			Campos clave: id_detalle Claves foráneas: fk_cabeceraFactura y código_ebook

Tabla: USUARIO

Campo	Tamaño	Tipo de Dato	Descripción
idUsuario	11	Entero	Clave única para cada usuario registrado.
nombreUsuario	45	Varchar	Apodo o sobrenombre para un usuario.
contraseña	45	Varchar	Seña secreta que permite el acceso a algo, a alguien o a un grupo de personas antes inaccesible.
Foto	50	Varchar	Imagen obtenida fotográficamente que identifica visualmente a una persona.
Nombre	45	Varchar	Palabra que designa o identifica a una persona.
Apellido	45	Varchar	Nombre de familia con que se distinguen las personas.
fechaNacimiento	-	Date	Indicación del día de nacimiento.
infoUsuario	-	Texto	Información acerca del usuario.
Género	1	Varchar	Taxón que agrupa a especies que comparten ciertos caracteres.
Email	45	Varchar	Dirección de correo electrónico.
País	45	Varchar	Nación, región, provincia o territorio.
Relaciones: Colección, Carrito y Cabecera_Factura.			Campos Clave: idUsuario.

Tabla: MENSAJE

Campo	Tamaño	Tipo de Dato	Descripción
id_mensaje	11	Entero	Clave única para cada mensaje registrado.
nombre	50	Varchar	Palabra que designa o identifica a una persona.
Email	50	Varchar	Dirección del sistema comunicación personal por ordenador a través de redes informáticas.
Asunto	50	Varchar	Tema o argumento de que se trata
Texto	-	Text	Enunciado o conjunto coherente de enunciados escritos.
Relaciones:			Campos clave: id_Mensaje

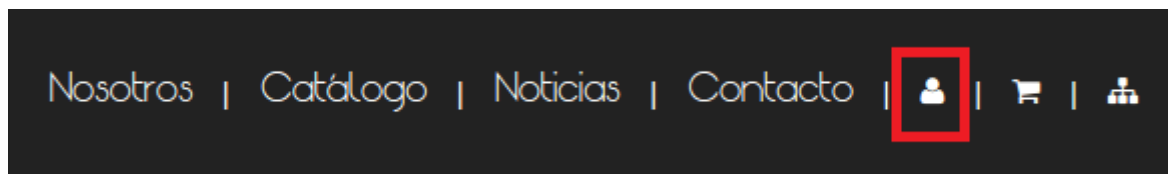
Tabla: AUTOR

Campo	Tamaño	Tipo de Dato	Descripción
idAutor	11	Entero	Clave única para cada Autor registrado.
nombreCompleto	45	Varchar	Palabra que designa o identifica seres animados o inanimados
Foto	45	Varchar	Imagen obtenida fotográficamente que identifica visualmente a una persona.
Relaciones: Ebook			Campos clave: idAutor.

3. Instrucciones de Uso

3.1 Navegación de la Administración

Para poder hacer uso de la interfaz del sitio web desde el localhost, se debe guardar la carpeta del sitio 'Proyecto-Venta-e-books' dentro del directorio '/var/www/html' y luego abrirla desde el navegador con la siguiente ruta: localhost/Proyecto-Venta-e-books/index.php (para poder acceder a la página principal del sitio). Para instalar los servicios necesarios para el correcto funcionamiento del sitio, ir a la sección **5. Instalación del Sistema**.



Dentro del sitio, se selecciona el ícono de usuario que nos lleva a la página de ingreso en la que se deben escribir las credenciales de administrador, por ejemplo:

Usuario: administrador
Contraseña: Pa\$\$w0rd



Luego de ingresar exitosamente, el perfil de administrador mostrará las opciones de tablas para poder administrarlas. Estas son: EBOOK, AUTOR, FACTURA, MENSAJE Y USUARIO.



Dentro de cada una de estas opciones, se muestran los registros completos con etiquetas

individuales para editar o eliminar los registros, así también como una opción en la parte superior para poder crear un registro nuevo.

[Nuevo](#)

Id	Usuario	Contraseña	Url de Foto	Nombre	Apellido	Fecha de Nacimiento	Información del Usuario	Género	Email	País
1	administrador	Pa\$\$w0rd	images/perfil/default.jpg	Admin	Ebbbooks	1989-08-03	Soy Admin	f	admin@gmail.com	Ecuador editar
2	connantrutman	connanmismo	images/perfil/default.jpg	Connan	Trutman	1980-05-23	asdasd	m	connanmismo@hotmail.com	Ecuador editar eliminar
3	jonathan92	venga	images/perfil/default.jpg	Jonathan	Lopez	1983-10-08	Soy un usuario de Prueba	m	jonathan92@gmail.com	Ecuador editar eliminar

[Regresar a Perfil de Administrador](#)

Tomando como ejemplo la opción de USUARIO, tenemos que el usuario con el id 1, que vendría a ser el administrador, no muestra la opción de eliminar ya que sería contradictorio que el mismo administrador pueda eliminar su propia cuenta de usuario en el sitio. Por este motivo, solo se muestra la opción de Editar.

En el caso de querer Editar algún registro, al seleccionar la opción se muestra un formulario con los campos llenos con la información del registro seleccionado para su edición. Todos los campos serán editables excepto por el id, que es generado automáticamente por la base datos al ser creado.

Si se escoge la opción de Eliminar, esta borrará los datos de ese registro no solo del sitio o vista del administrador, sino también de la base de datos por lo que se recomienda manejar con cuidado esta opción y que sólo sea manejada por un administrador autorizado del sitio.

Editar Objeto Usuario

Id:

Usuario:

Contraseña:

Ruta de foto:

Nombre:

Apellido:

Fecha de Nacimiento:

Acerca de mi:

Género:

Correo:

País:

[Cancelar](#) [Guardar](#)

Usuario:

Contraseña:

Ruta de foto:

Nombre:

Apellido:

Fecha de Nacimiento:

Acerca de mi:

Género:
Masculino ☐ Femenino ☐

Correo:

País:

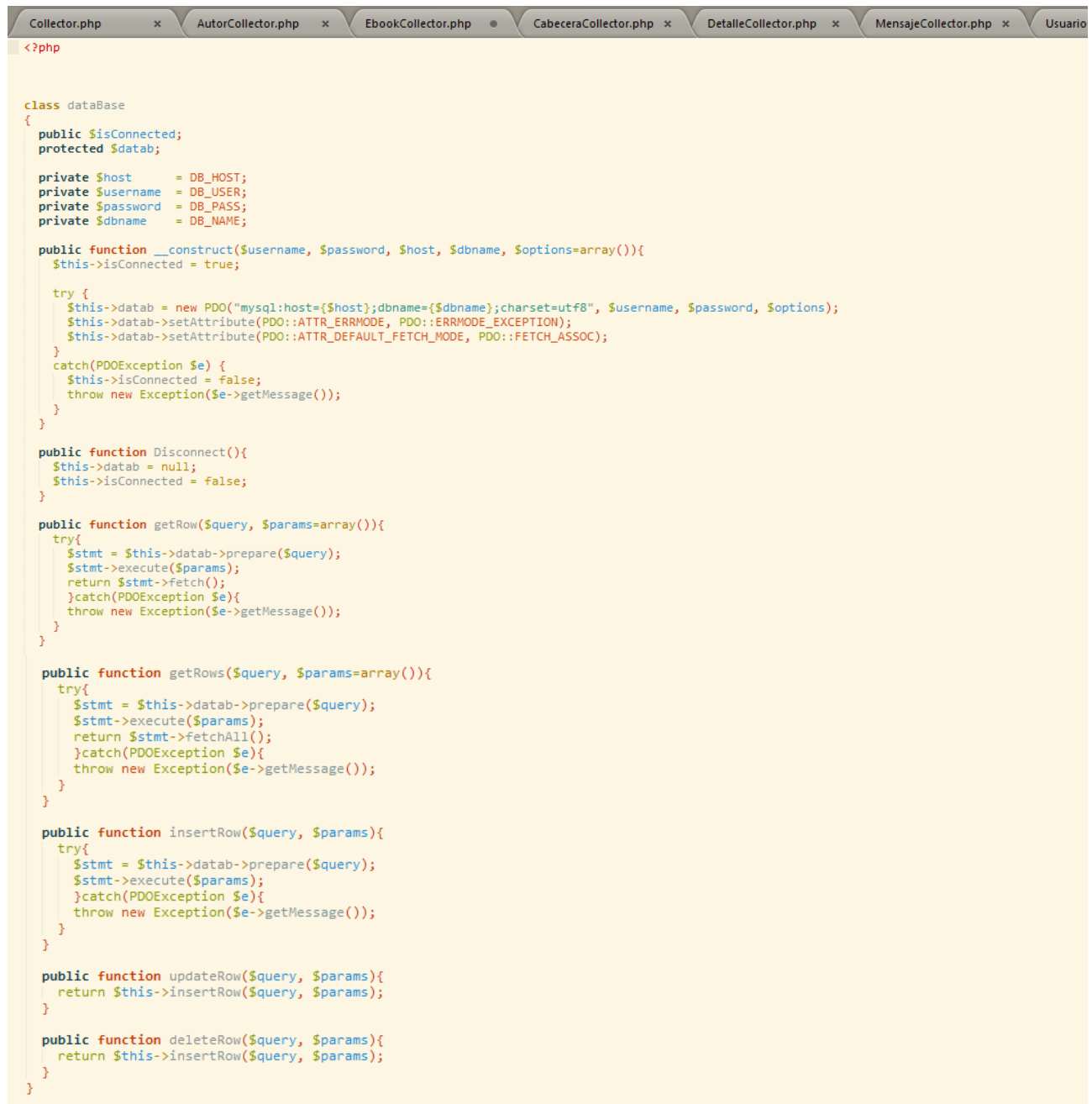
Finalmente para la opción de Nuevo registro, se mostrará un formulario vacío para poder llenar cada campo con la información necesaria para ingresarla a la base de datos al hacer clic en Enviar.

Para poder regresar al perfil de Administrador y salir del sitio debemos acceder a la opción de Regresar a Perfil de Administrador que se encuentra al final del listado de registros dentro de cada opción disponible para administrar.

3.2 Uso de archivos y funciones

Clase: dataBase.

Ubicación: Crud de todas las tablas del sitio y carpeta principal del sitio.



```
<?php

class dataBase
{
    public $isConnected;
    protected $datab;

    private $host      = DB_HOST;
    private $username  = DB_USER;
    private $password  = DB_PASS;
    private $dbname    = DB_NAME;

    public function __construct($username, $password, $host, $dbname, $options=array()){
        $this->isConnected = true;

        try {
            $this->datab = new PDO("mysql:host={$host};dbname={$dbname};charset=utf8", $username, $password, $options);
            $this->datab->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
            $this->datab->setAttribute(PDO::ATTR_DEFAULT_FETCH_MODE, PDO::FETCH_ASSOC);
        }
        catch(PDOException $e) {
            $this->isConnected = false;
            throw new Exception($e->getMessage());
        }
    }

    public function Disconnect(){
        $this->datab = null;
        $this->isConnected = false;
    }

    public function getRow($query, $params=array()){
        try{
            $stmt = $this->datab->prepare($query);
            $stmt->execute($params);
            return $stmt->fetch();
        }catch(PDOException $e){
            throw new Exception($e->getMessage());
        }
    }

    public function getRows($query, $params=array()){
        try{
            $stmt = $this->datab->prepare($query);
            $stmt->execute($params);
            return $stmt->fetchAll();
        }catch(PDOException $e){
            throw new Exception($e->getMessage());
        }
    }

    public function insertRow($query, $params){
        try{
            $stmt = $this->datab->prepare($query);
            $stmt->execute($params);
        }catch(PDOException $e){
            throw new Exception($e->getMessage());
        }
    }

    public function updateRow($query, $params){
        return $this->insertRow($query, $params);
    }

    public function deleteRow($query, $params){
        return $this->insertRow($query, $params);
    }
}
```

- La clase dataBase servirá para el funcionamiento de las demás clases ya que se encargará de permitir la conexión de la base de datos y tendrá funciones importantes como: insertar, eliminar, actualizar o enlistar.
- En la clase dataBase se guardan los parámetros de configuración de la base de datos.
 - **DB_HOST:** host.

- **DB_USER:** username.
 - **DB_PASS:** password.
 - **DB_NAME:** dbname.
- Contiene una variable publica \$isConnected que utilizara para saber si la conexión a la base esta activa.
- Se crea un constructor donde va recibir los parámetros que fueron enviados desde la clase Collector. Por medio de PDO se establece la conexión a la base de datos MySQL.
- Se crea una función **Disconnect** para poder desactivar la conexión con la base de datos.
- Se implementan las funciones principales:
 - **getRow:** Obtiene una fila de registros que se encuentre en la base de datos.
 - **getRows:** Obtiene los registros que se encuentran en la base de datos.
 - **insertRow:** Inserta nuevos registros en la base de datos.
 - **updateRow:** Actualiza los registros de la base de datos.
 - **deleteRow:** Elimina registros de la base.

Clase: Collector.

Ubicación: Crud de todas las tablas y carpeta principal del sitio.



```
Collector.php *
<?php
include_once('dataBase.php');

// Define configuration
define("DB_HOST", "localhost");
define("DB_USER", "root");
define("DB_PASS", "root");
define("DB_NAME", "ebbbbooks");

class Collector extends DataBase
{
    public static $db;
    private $host = DB_HOST;
    private $username = DB_USER;
    private $password = DB_PASS;
    private $dbname = DB_NAME;

    public function __construct()
    {
        self::$db = new DataBase($this->username, $this->password, $this->host, $this->dbname);
    }
}

?>
```

- La clase Collector incluye la clase DataBase para poder funcionar correctamente.
- Se definen los parámetros de configuración de la base de datos que se utilizara.
 - **DB_HOST:** se define el nombre del host donde está la base de datos.
 - **DB_USER:** usuario que accede a la base de datos.
 - **DB_PASS:** contraseña para acceder a la base de datos.
 - **DB_NAME:** nombre de la base de datos.
- La clase Collector extiende de la clase DataBase.
- Crea un método estático \$db. Este contiene variables privadas (\$host, \$username, \$password, \$dbname) que tienen asociado los parámetros que se definieron con anterioridad (DB_HOST, DB_USER, DB_PASS, DB_NAME).
- Se crea un constructor y dentro se establece un nuevo objeto de tipo DataBase que es asignado al método estático \$db. El constructor servirá para enviar las variables privadas que ya se definieron hacia el constructor creado en la clase DataBase que verificará los parámetros que fueron enviados y permitirá la conexión de la base.
- Esta clase Collector tiene como objetivo evitar la redundancia y optimizar el código a la hora de hacer la conexión.

Clase: AutorCollector.

Ubicación: Crud la tabla Autor y carpeta principal del sitio.



```
<?php
include_once('Autor.php');
include_once('Collector.php');

class AutorCollector extends Collector
{
    function showAutor($id) {
        $row = self::$db->getRows("SELECT * FROM autor where idAutor= ? ", array("{ $id }"));
        $ObjAutor = new Autor($row[0]['idAutor'],$row[0]['nombreCompleto']);
        return $ObjAutor;
    }

    function createAutor($nombre) {
        $insertrow = self::$db->insertRow("INSERT INTO ebbbooks.autor (idAutor, nombreCompleto) VALUES (?, ?)", array(null, "{ $nombre }"));
    }

    function readAutor() {
        $rows = self::$db->getRows("SELECT * FROM autor ");
        $arrayAutor= array();
        foreach ($rows as $c){
            $aux = new Autor($c['idAutor'],$c['nombreCompleto']);
            array_push($arrayAutor, $aux);
        }
        return $arrayAutor;
    }

    function updateAutor($id,$nombreCompleto) {
        $insertrow = self::$db->updateRow("UPDATE ebbbooks.autor SET autor.nombreCompleto = ? WHERE autor.idAutor = ? ", array( "{ $nombreCompleto }", $id));
    }

    function deleteAutor($id) {
        $deleterow = self::$db->deleteRow("DELETE FROM ebbbooks.autor WHERE idAutor= ?", array("{ $id }"));
    }
}
?>
```

- La clase AutorCollector incluye la clase Collector(definida anteriormente) y Autor(objeto) para poder funcionar correctamente.
- La clase AutorCollector extiende de la clase Collector(que a su vez extiende de dataBase).
- Se crean las funciones principales:
 - **showAutor:** Obtiene los registros que se encuentran en la tabla Autor.
 - Envía \$id como parámetro.
 - Utiliza el método getRows(definido antes en la clase dataBase) para poder extraer los registros de la base.
 - Emplea la sentencia SQL **SELECT * FROM autor WHERE idAutor=?** donde comparara el id que ha sido enviado con los registros en la base.
 - Retorna los datos obtenidos en un arreglo de tipo Autor(\$ObjAutor).
 - **createAutor:** Crea nuevos registros de autor dentro de la tabla Autor de la base datos.
 - Envía \$nombre como parámetro.
 - Utiliza el método insertRow(definido antes en la clase dataBase) para poder insertar registros.
 - Emplea la sentencia SQL **INSERT INTO ebbbooks.autor** para poder llenar en la base los nuevos valores de idAutor y nombreCompleto.

- **readAutor:** Enlista todo el contenido que se encuentran registrado en la tabla Autor.
 - No envía parámetros.
 - Utiliza el método `getRows`(definido antes en la clase `dataBase`) para poder extraer los registros de la base.
 - Emplea la sentencia SQL **SELECT * FROM autor** para obtener todos los datos registrados en la tabla.
 - Retorna los datos obtenidos en un arreglo de tipo `Autor($arrayAutor)`.
- **updateAutor:** Actualiza los registros dentro de la tabla Autor.
 - Envía `$id` y `$nombreCompleto` como parámetro.
 - Utiliza el método `updateRow`(definido antes en la clase `dataBase`) para poder actualizar registros.
 - Emplea una sentencia SQL **UPDATE ebbbooks.autor SET autor.nombreCompleto WHERE idAutor=?** y reemplaza con nuevos valores los registros que coincidan con el `id` envía como parámetro.
- **deleteAutor:** Elimina registros de la tabla Autor.
 - Envía `$id` como parámetro.
 - Utiliza el método `deleteRow`(definido antes en la clase `dataBase`) para poder eliminar registros.
 - Emplea una sentencia SQL **DELETE FROM ebbbooks.autor WHERE idAutor=?** para poder eliminar registros según el `id` que se envió.

Clase: EbookCollector.

Ubicación: Crud la tabla Ebook y carpeta principal del sitio.

```
<?php
include_once('Ebook.php');
include_once('Collector.php');
include_once('AutorCollector.php');

class EbookCollector extends Collector
{
    function showEbook_byId ($id) {
        $AutorCollectorObj = new AutorCollector();

        $row = self::$db->getRow("SELECT * FROM ebook WHERE idEbook LIKE '%" . $id . "%'");
        $autor=$AutorCollectorObj->showAutorId($id);

        $aux = new Ebook($row['idEbook'], $row['titulo'], $autor->getnombreCompleto(), $row['precio'], $row['idioma'],
            $row['numero_Paginas'], $row['categoria'], $row['isbn'], $row['editorial'], $row['traductor'], $row['portada'], $row['resena'], $row['enlace'] );
        return $aux;
    }

    function showEbook() {
        $AutorCollectorObj = new AutorCollector();

        $rows = self::$db->getRows("SELECT * FROM ebook");
        $arrayEbook= array();
        foreach ($rows as $c){

            $autor=$AutorCollectorObj->showAutorId($c['fk_autor']);

            $aux = new Ebook($c['idEbook'], $c['titulo'], $autor->getnombreCompleto(), $c['precio'], $c['idioma'],
                $c['numero_Paginas'], $c['categoria'], $c['isbn'], $c['editorial'], $c['traductor'], $c['portada'], $c['resena'], $c['enlace'] );
            array_push($arrayEbook, $aux);
        }
        return $arrayEbook;
    }

    function showEbooks($id) {
        $row = self::$db->getRows("SELECT * FROM ebook where idEbook= ? ", array("$id"));

        $ObjEbook = new Ebook($row[0]['idEbook'], $row[0]['titulo'], $row[0]['fk_autor'], $row[0]['precio'], $row[0]['idioma'], $row[0]['numero_Paginas'],
            $row[0]['categoria'], $row[0]['isbn'], $row[0]['editorial'], $row[0]['traductor'],
            $row[0]['portada'], $row[0]['resena'], $row[0]['enlace']);
        return $ObjEbook;
    }

    function createEbook($titulo, $fk_autor, $precio, $idioma, $numero_Paginas, $categoria, $isbn, $editorial, $traductor, $portada, $resena, $enlace) {
        $insertrow = self::$db->insertRow("INSERT INTO ebbbooks.ebook (idEbook, titulo, fk_autor, precio, idioma, numero_Paginas, categoria, isbn, editorial, traductor, portada, resena, enlace)
        VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)", array(null, "$titulo", "$fk_autor", "$precio", "$idioma", "$numero_Paginas", "$categoria", "$isbn", "$editorial",
            "$traductor", "$portada", "$resena", "$enlace"));
    }

    function readEbooks() {
        $rows = self::$db->getRows("SELECT * FROM ebook ");
        $arrayEbook= array();
        foreach ($rows as $c){
            $aux = new Ebook($c['idEbook'], $c['titulo'], $c['fk_autor'], $c['precio'], $c['idioma'], $c['numero_Paginas'], $c['categoria'], $c['isbn'], $c['editorial'],
                $c['traductor'], $c['portada'], $c['resena'], $c['enlace']);
            array_push($arrayEbook, $aux);
        }
        return $arrayEbook;
    }

    function updateEbook($id, $titulo, $fk_autor, $precio, $idioma, $numero_Paginas, $categoria, $isbn, $editorial, $traductor, $portada, $resena, $enlace) {
        $insertrow = self::$db->updateRow("UPDATE ebbbooks.ebook SET ebook.titulo = ?, ebook.fk_autor = ?, ebook.precio = ?, ebook.idioma = ?, ebook.numero_Paginas = ?, ebook.categoria = ?,
        ebook.isbn = ?, ebook.editorial = ?, ebook.traductor = ?, ebook.portada = ?, ebook.resena = ?, ebook.enlace = ? WHERE ebook.idEbook = ? ", array( "$titulo", "$fk_autor", "$precio",
            "$idioma", "$numero_Paginas", "$categoria", "$isbn", "$editorial", "$traductor", "$portada", "$resena", "$enlace", $id));
    }

    function deleteEbook($id) {
        $deleterow = self::$db->deleteRow("DELETE FROM ebbbooks.ebook WHERE idEbook= ?", array("$id"));
    }
}
?>
```

- La clase EbookCollector incluye la clase Collector(definida anteriormente), AutorCollector y Ebook(objeto) para poder funcionar correctamente.
- La clase EbookCollector extiende de la clase Collector(que a su vez extiende de dataBase).
- Se crean las funciones principales:
 - **showEbook_byId**: Obtiene un registro de la base de datos según un id.
 - Envía \$id como parámetro.
 - Se crea un objeto de tipo AutorCollector.
 - Utiliza el método getRow(definido antes en la clase dataBase) para poder extraer los registros de la base.
 - Emplea la sentencia SQL **SELECT * FROM ebook WHERE idEbook LIKE = \$id** donde comparara el id que ha sido enviado con los registros en la base.

- Con el objeto AutorCollector creado se llama a la función showAutorId que recibe un \$id.
 - Retorna los datos obtenidos en un arreglo de tipo Ebook(\$aux).
- **showEbook:** Obtiene todos los datos que se encuentran registrados en la tabla Cabeecera.
 - No envía parámetros.
 - Utiliza el método getRows(definido antes en la clase dataBase) para poder extraer los registros de la base.
 - Emplea la sentencia SQL **SELECT * FROM ebook** para obtener todos los datos registrados en la tabla.
 - Con el objeto AutorCollector creado se llama a la función showAutorId que recibe \$fk_autor.
 - Retorna los datos obtenidos en un arreglo de tipo Ebook(\$arrayEbook).
- **showEbooks:** Obtiene los registros que se encuentran en la tabla Ebook.
 - Envía \$id como parámetro.
 - Utiliza el método getRows(definido antes en la clase dataBase) para poder extraer los registros de la base.
 - Emplea la sentencia SQL **SELECT * FROM ebook WHERE idEbook=?** donde comparara el id que ha sido enviado con los registros en la base.
 - Retorna los datos obtenidos en un arreglo de tipo Ebook(\$ObjEbook).
- **createEbook:** Crea nuevos registros de ebooks dentro de la tabla Ebook de la base datos.
 - Envía todos los atributos del objeto Ebook como parámetros.
 - Utiliza el método insertRow(definido antes en la clase dataBase) para poder insertar registros.
 - Emplea la sentencia SQL **INSERT INTO ebbbooks.ebook** para poder llenar en la base los nuevos valores para los atributos de Ebook.
- **readEbooks:** Enlista todo el contenido que se encuentran registrado en la tabla Detalle.
 - No envía parámetros.
 - Utiliza el método getRows(definido antes en la clase dataBase) para poder extraer los registros de la base.
 - Emplea la sentencia SQL **SELECT * FROM ebook** para obtener todos los datos registrados en la tabla.
 - Retorna los datos obtenidos en un arreglo de tipo Ebook(\$arrayEbook).
- **updateEbook:** Actualiza los registros dentro de la tabla Detalle.
 - Envía todos los atributos del objeto Ebook como parámetros.
 - Utiliza el método updateRow(definido antes en la clase dataBase) para poder actualizar registros.

- Emplea una sentencia SQL **UPDATE ebbbooks.ebook SET ebook.titulo WHERE ebook.idEbook =?** y reemplaza con nuevos valores los registros que coincidan con el id envía como parámetro.
- **deleteEbook:** Elimina registros de la tabla Detalle.
 - Envía \$id como parámetro.
 - Utiliza el método deleteRow(definido antes en la clase dataBase) para poder eliminar registros.
 - Emplea una sentencia SQL **DELETE FROM ebbbooks.ebook WHERE idEbook=?** para poder eliminar registros según el id que se envió.

Clase: CabeceraCollector.

Ubicación: Crud la tabla Cabecera_Factura y carpeta principal del sitio.

```
Collector.php x AutorCollector.php x EbookCollector.php x CabeceraCollector.php x DetalleCollector.php x MensajeCollector.php x UsuarioCollector.php x
<?php
include_once ('Cabecera.php');
include_once ('Collector.php');

class CabeceraCollector extends Collector{

    function showCabecera($id) {
        $rows = self::$db->getRows("SELECT * FROM cabecera_factura where idCabecera= ? ", array($id));
        $objCabecera = new Cabecera($rows[0]['idCabecera'],$rows[0]['fk_usuario'],$rows[0]['fecha'],$rows[0]['total_pagar']);
        return $objCabecera;
    }

    function showCabeceras(){
        $rows = self::$db->getRows("SELECT * FROM cabecera_factura");
        $arrayCabecera = array();
        foreach ($rows as $c) {
            $tmp = new Cabecera($c['idCabecera'], $c['fk_usuario'], $c['fecha'], $c['total_pagar']);
            $arrayCabecera[] = $tmp;
        }
        return $arrayCabecera;
    }

    function createCabecera($fk_usuario, $fecha, $total_pagar) {
        $insertRow = self::$db->insertRow("INSERT INTO ebbbooks.cabecera_factura (idCabecera, fk_usuario, fecha, total_pagar) VALUES (?, ?, ?, ?)",
        array(null, $fk_usuario, $fecha, $total_pagar));
    }

    function readCabeceras(){
        $rows = self::$db->getRows("SELECT * FROM cabecera_factura");
        $arrayCabecera = array();
        foreach ($rows as $c) {
            $tmp = new Cabecera($c['idCabecera'], $c['fk_usuario'], $c['fecha'], $c['total_pagar']);
            $arrayCabecera[] = $tmp;
        }
        return $arrayCabecera;
    }

    function updateCabecera($id, $fk_usuario, $fecha, $total_pagar) {
        $insertRow = self::$db->updateRow("UPDATE ebbbooks.cabecera_factura SET cabecera_factura.fk_usuario = ?, cabecera_factura.fecha = ?, cabecera_factura.total_pagar = ?
        WHERE cabecera_factura.idCabecera = ?", array($fk_usuario, $fecha, $total_pagar, $id));
    }

    function deleteCabecera($id) {
        $deleteRow = self::$db->deleteRow("DELETE FROM ebbbooks.cabecera_factura WHERE idCabecera= ?", array("$id"));
    }

}
??
```

- La clase CabeceraCollector incluye la clase Collector(definida anteriormente) y Cabece-
ra(objeto) para poder funcionar correctamente.
- La clase CabeceraCollector extiende de la clase Collector(que a su vez extiende de dataBa-
se).
- Se crean las funciones principales:
 - **showCabecera:** Obtiene los registros que se encuentran en la tabla Cabecera.
 - Envía \$id como parámetro.
 - Utiliza el método getRows(definido antes en la clase dataBase) para poder extraer los registros de la base.
 - Emplea la sentencia SQL **SELECT * FROM cabecera_factura WHERE idCabe-
cera=?** donde comparara el id que ha sido enviado con los registros en la ba-
se.
 - Retorna los datos obtenidos en un arreglo de tipo Cabecera(\$ObjCabecera).
 - **showCabeceras:** Obtiene todos los datos que se encuentran registrados en la tabla Cabecera.
 - No envía parámetros.
 - Utiliza el método getRows(definido antes en la clase dataBase) para poder extraer los registros de la base.

- Emplea la sentencia SQL **SELECT * FROM cabecera_factura** para obtener todos los datos registrados en la tabla.
 - Retorna los datos obtenidos en un arreglo de tipo `Cabecera($arrayCabecera)`.
- **createCabercera:** Crea nuevos registros de cabecera dentro de la tabla Cabecera de la base datos.
 - Envía `$fk_usuario`, `$fecha`, `$total_pagar` como parámetros.
 - Utiliza el método `insertRow`(definido antes en la clase `dataBase`) para poder insertar registros.
 - Emplea la sentencia SQL **INSERT INTO ebbbooks.cabecera_factura** para poder llenar en la base los nuevos valores de `idCabecera`, `fk_usuario`, `fecha`, `$total_pagar`.
- **readCaberceras:** Enlista todo el contenido que se encuentran registrado en la tabla Cabecera.
 - No envía parámetros.
 - Utiliza el método `getRows`(definido antes en la clase `dataBase`) para poder extraer los registros de la base.
 - Emplea la sentencia SQL **SELECT * FROM cabecera_factura** para obtener todos los datos registrados en la tabla.
 - Retorna los datos obtenidos en un arreglo de tipo `Cabecera($arrayCabecera)`.
- **updateCabercera:** Actualiza los registros dentro de la tabla Cabecera.
 - Envía `$id`, `$fk_usuario`, `$fecha`, `$total_pagar` como parámetros.
 - Utiliza el método `updateRow`(definido antes en la clase `dataBase`) para poder actualizar registros.
 - Emplea una sentencia SQL **UPDATE ebbbooks.cabecera_factura SET cabecera_factura.fecha WHERE cabecera_factura.idCabecera =?** y reemplaza con nuevos valores los registros que coincidan con el id envía como parámetro.
- **deleteCabercera:** Elimina registros de la tabla Cabecera.
 - Envía `$id` como parámetro.
 - Utiliza el método `deleteRow`(definido antes en la clase `dataBase`) para poder eliminar registros.
 - Emplea una sentencia SQL **DELETE FROM ebbbooks.cabecera_factura WHERE idCabecera=?** para poder eliminar registros según el id que se envió.

Clase: DetalleCollector.

Ubicación: Crud la tabla Detalle_Factura y carpeta principal del sitio.

```
Collector.php x AutorCollector.php x EbookCollector.php x CabeceraCollector.php x DetalleCollector.php x MensajeCollector.php x UsuarioCollector.php x
<?php
include_once ('Detalle.php');
include_once ('Collector.php');

class DetalleCollector extends Collector{

    function showDetalle($id) {
        $row = self::$db->getRows("SELECT * FROM detalle_factura where idDetalle= ? ", array($id));
        $objDetalle = new Detalle($row[0]['idDetalle'],$row[0]['fk_cabeceraFactura'],$row[0]['linea'],$row[0]['codigoEbook']);
        return $objDetalle;
    }

    function showDetalles(){
        $rows = self::$db->getRows("SELECT * FROM detalle_factura");
        $arrayDetalle = array();
        foreach ($rows as $c){
            $tmp = new Detalle($c['idDetalle'], $c['fk_cabeceraFactura'], $c['linea'], $c['codigoEbook']);
            $arrayDetalle[] = $tmp;
        }
        return $arrayDetalle;
    }

    function createDetalle($fk_cabeceraFactura, $linea, $codigoEbook) {
        $insertRow = self::$db->insertRow("INSERT INTO ebbbooks.detalle_factura (idDetalle, fk_cabeceraFactura, linea, codigoEbook) VALUES (?, ?, ?, ?)",
        array(null, $fk_cabeceraFactura, $linea, $codigoEbook));
    }

    function readDetalles(){
        $rows = self::$db->getRows("SELECT * FROM detalle_factura");
        $arrayDetalle = array();
        foreach ($rows as $c){
            $tmp = new Detalle($c['idDetalle'], $c['fk_cabeceraFactura'], $c['linea'], $c['codigoEbook']);
            $arrayDetalle[] = $tmp;
        }
        return $arrayDetalle;
    }

    function updateDetalle($id, $fk_cabeceraFactura, $linea, $codigoEbook) {
        $insertRow = self::$db->updateRow("UPDATE ebbbooks.detalle_factura SET detalle_factura.fk_cabeceraFactura = ?, detalle_factura.linea = ?, detalle_factura.codigoEbook = ?
        WHERE detalle_factura.idDetalle = ?", array($fk_cabeceraFactura, $linea, $codigoEbook, $id));
    }

    function deleteDetalle($id) {
        $deleteRow = self::$db->deleteRow("DELETE FROM ebbbooks.detalle_factura WHERE idDetalle= ?", array("$id"));
    }
}
```

- La clase DetalleCollector incluye la clase Collector(definida anteriormente) y Detalle(objeto) para poder funcionar correctamente.
- La clase DetalleCollector extiende de la clase Collector(que a su vez extiende de dataBase).
- Se crean las funciones principales:
 - **showDetalle:** Obtiene los registros que se encuentran en la tabla Detalle.
 - Envía \$id como parámetro.
 - Utiliza el método getRows(definido antes en la clase dataBase) para poder extraer los registros de la base.
 - Emplea la sentencia SQL **SELECT * FROM detalle_factura WHERE idDetalle=?** donde comparara el id que ha sido enviado con los registros en la base.
 - Retorna los datos obtenidos en un arreglo de tipo Detalle(\$ObjDetalle).
 - **showDetalles:** Obtiene todos los datos que se encuentran registrados en la tabla Cabecera.
 - No envía parámetros.
 - Utiliza el método getRows(definido antes en la clase dataBase) para poder extraer los registros de la base.
 - Emplea la sentencia SQL **SELECT * FROM detalle_factura** para obtener todos los datos registrados en la tabla.
 - Retorna los datos obtenidos en un arreglo de tipo Detalle(\$arrayDetalle).
 - **createDetalle:** Crea nuevos registros de detalle dentro de la tabla Detalle de la base

datos.

- Envía \$fk_cabeceraFactura, \$linea, \$codigoEbook como parámetros.
 - Utiliza el método insertRow(definido antes en la clase dataBase) para poder insertar registros.
 - Emplea la sentencia SQL **INSERT INTO ebbbooks.detalle_factura** para poder llenar en la base los nuevos valores de fk_cabeceraFactura, linea, codigoEbook.
- **readDetalles:** Enlista todo el contenido que se encuentran registrado en la tabla Detalle.
 - No envía parámetros.
 - Utiliza el método getRows(definido antes en la clase dataBase) para poder extraer los registros de la base.
 - Emplea la sentencia SQL **SELECT * FROM detalle_factura** para obtener todos los datos registrados en la tabla.
 - Retorna los datos obtenidos en un arreglo de tipo Detalle(\$arrayDetalle).
 - **updateDetalle:** Actualiza los registros dentro de la tabla Detalle.
 - Envía \$id, \$fk_cabeceraFactura, \$linea, \$codigoEbook como parámetros.
 - Utiliza el método updateRow(definido antes en la clase dataBase) para poder actualizar registros.
 - Emplea una sentencia SQL **UPDATE ebbbooks.detalle_factura SET detalle_factura.linea WHERE detalle_factura.idDetalle =?** y reemplaza con nuevos valores los registros que coincidan con el id envía como parámetro.
 - **deleteDetalle:** Elimina registros de la tabla Detalle.
 - Envía \$id como parámetro.
 - Utiliza el método deleteRow(definido antes en la clase dataBase) para poder eliminar registros.
 - Emplea una sentencia SQL **DELETE FROM ebbbooks.detalle_factura WHERE idDetalle=?** para poder eliminar registros según el id que se envió.

Clase: MensajeCollector.

Ubicación: Crud la tabla Mensaje y carpeta principal del sitio.

```
Collector.php x AutorCollector.php x EbookCollector.php x CabeceraCollector.php x DetalleCollector.php x MensajeCollector.php x Usu
<?php
include_once ('Mensaje.php');
include_once ('Collector.php');

class MensajeCollector extends Collector{

    function showMensaje($id){
        $rows = self::$db->getRows("SELECT * FROM mensaje WHERE idMensaje = ?", array($id));
        $objMensaje = new Mensaje($row[0]['idMensaje'],$row[0]['nombre'],$row[0]['email'],$row[0]['asunto'],$row[0]['texto']);
        return $objMensaje;
    }

    function showMensajes(){
        $rows = self::$db->getRows("SELECT * FROM mensaje");
        $arrayMensaje = array();
        foreach ($rows as $c){
            $tmp = new Mensaje($c['idMensaje'], $c['nombre'], $c['email'], $c['asunto'], $c['texto']);
            $arrayMensaje[] = $tmp;
        }
        return $arrayMensaje;
    }

    function createMensaje($nombre, $email, $asunto, $texto){
        $insertRow = self::$db->insertRow("INSERT INTO ebbbooks.mensaje (idMensaje, nombre, email, asunto, texto) VALUES (?, ?, ?, ?, ?)",
        array(null, $nombre, $email, $asunto, $texto));
    }

    function readMensajes(){
        $rows = self::$db->getRows("SELECT * FROM mensaje");
        $arrayMensaje = array();
        foreach ($rows as $c){
            $tmp = new Mensaje($c['idMensaje'], $c['nombre'], $c['email'], $c['asunto'], $c['texto']);
            $arrayMensaje[] = $tmp;
        }
        return $arrayMensaje;
    }

    function updateMensaje($id, $nombre, $email, $asunto, $texto){
        $insertRow = self::$db->updateRow("UPDATE ebbbooks.mensaje SET mensaje.nombre = ?, mensaje.email = ?, mensaje.asunto = ?,
        mensaje.texto = ? WHERE mensaje.idMensaje = ?", array($nombre, $email, $asunto, $texto, $id));
    }

    function deleteMensaje($id){
        $deleteRow = self::$db->deleteRow("DELETE FROM ebbbooks.mensaje WHERE idMensaje= ?", array("{$id}"));
    }
}
```

- La clase MensajeCollector incluye la clase Collector(definida anteriormente) y Mensaje(objeto) para poder funcionar correctamente.
- La clase MensajeCollector extiende de la clase Collector(que a su vez extiende de dataBase).
- Se crean las funciones principales:
 - **showMensaje:** Obtiene los registros que se encuentran en la tabla Mensaje.
 - Envía \$id como parámetro.
 - Utiliza el método getRows(definido antes en la clase dataBase) para poder extraer los registros de la base.
 - Emplea la sentencia SQL **SELECT * FROM mensaje WHERE idMensaje=?** donde comparara el id que ha sido enviado con los registros en la base.
 - Retorna los datos obtenidos en un arreglo de tipo Mensaje(\$ObjMensaje).
 - **showMensajes:** Obtiene todos los datos que se encuentran registrados en la tabla Mensaje.
 - No envía parámetros.
 - Utiliza el método getRows(definido antes en la clase dataBase) para poder extraer los registros de la base.

- Emplea la sentencia SQL **SELECT * FROM mensaje** para obtener todos los datos registrados en la tabla.
 - Retorna los datos obtenidos en un arreglo de tipo Mensaje (\$arrayMensaje).
- **createMensaje:** Crea nuevos registros de detalle dentro de la tabla Mensaje de la base datos.
 - Envía \$nombre, \$email, \$asunto, \$texto como parámetros.
 - Utiliza el método insertRow(definido antes en la clase dataBase) para poder insertar registros.
 - Emplea la sentencia SQL **INSERT INTO ebbbooks.mensaje** para poder llenar en la base los nuevos valores de nombre, email, asunto, texto.
- **readMensajes:** Enlista todo el contenido que se encuentran registrado en la tabla Mensaje.
 - No envía parámetros.
 - Utiliza el método getRows(definido antes en la clase dataBase) para poder extraer los registros de la base.
 - Emplea la sentencia SQL **SELECT * FROM mensaje** para obtener todos los datos registrados en la tabla.
 - Retorna los datos obtenidos en un arreglo de tipo Mensaje(\$arrayMensaje).
- **updateMensaje:** Actualiza los registros dentro de la tabla Mensaje.
 - Envía \$id, Envía \$nombre, \$email, \$asunto, \$texto como parámetros.
 - Utiliza el método updateRow(definido antes en la clase dataBase) para poder actualizar registros.
 - Emplea una sentencia SQL **UPDATE ebbbooks.mensaje SET mensaje.nombre WHERE mensaje.idMensaje =?** y reemplaza con nuevos valores los registros que coincidan con el id envía como parámetro.
- **deleteMensaje:** Elimina registros de la tabla Mensaje.
 - Envía \$id como parámetro.
 - Utiliza el método deleteRow(definido antes en la clase dataBase) para poder eliminar registros.
 - Emplea una sentencia SQL **DELETE FROM ebbbooks.mensaje WHERE id-Mensaje=?** para poder eliminar registros según el id que se envió.

Clase: UsuarioCollector.

Ubicación: Crud la tabla Usuario y carpeta principal del sitio.

```
Collector.php x AutorCollector.php — crudAutor x EbookCollector.php • AutorCollector.php — Ebook x CabeceraCollector.php x DetalleCollector.php x Mens

<?php
include_once('Usuario.php');
include_once('Collector.php');

class UsuarioCollector extends Collector
{
    function showUsuario($id) {
        $row = self::$db->getRows("SELECT * FROM usuario where idUsuario= ? ", array("$id"));
        $ObjUsuario = new Usuario($row[0]['idUsuario'],$row[0]['nombreUsuario'],$row[0]['contrasena'],$row[0]['foto'],$row[0]['nombre'],
        $row[0]['apellido'],$row[0]['fechaNacimiento'],$row[0]['infoUsuario'],$row[0]['genero'],$row[0]['email'],$row[0]['pais']);
        return $ObjUsuario;
    }

    function createUsuario($nombre,$contrasena) {
        $insertrow = self::$db->insertRow("INSERT INTO ebbbooks.usuario (idUsuario, nombreUsuario, contrasena, foto, nombre, apellido, fechaNacimiento,
        infoUsuario, genero, email, pais) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)", array(null, "{$nombreUsuario}", "{$contrasena}", "{$foto}", "{$nombre}",
        "{$apellido}", "{$fechaNacimiento}", "{$infoUsuario}", "{$genero}", "{$email}", "{$pais}"));
    }

    function readUsuarios() {
        $rows = self::$db->getRows("SELECT * FROM usuario ");
        $arrayUsuario= array();
        foreach ($rows as $c){
            $aux = new Usuario($c['idUsuario'],$c['nombreUsuario'],$c['contrasena'],$c['foto'],$c['nombre'],$c['apellido'],$c['fechaNacimiento'],
            $c['infoUsuario'],$c['genero'],$c['email'],$c['pais']);
            array_push($arrayUsuario, $aux);
        }
        return $arrayUsuario;
    }

    function updateUsuario($id,$nombreUsuario,$contrasena,$foto,$nombre,$apellido,$fechaNacimiento,$infoUsuario,$genero,$email,$pais) {
        $insertrow = self::$db->updateRow("UPDATE ebbbooks.usuario SET usuario.nombreUsuario = ?, usuario.contrasena = ?, usuario.foto = ?, usuario.nombre = ?,
        usuario.apellido = ?, usuario.fechaNacimiento = ?, usuario.infoUsuario = ?, usuario.genero = ?, usuario.email = ?, usuario.pais = ?
        WHERE usuario.idUsuario = ? ", array( "{$nombreUsuario}", "{$contrasena}", "{$foto}", "{$nombre}", "{$apellido}", "{$fechaNacimiento}", "{$infoUsuario}",
        "{$genero}", "{$email}", "{$pais}", $id));
    }

    function deleteUsuario($id) {
        $deleterow = self::$db->deleteRow("DELETE FROM ebbbooks.usuario WHERE idUsuario= ? ", array("$id"));
    }
}
?>
```

- La clase UsuarioCollector incluye la clase Collector(definida anteriormente) y Usuario(objeto) para poder funcionar correctamente.
- La clase UsuarioCollector extiende de la clase Collector(que a su vez extiende de dataBase).
- Se crean las funciones principales:
 - **showUsuario:** Obtiene los registros que se encuentran en la tabla Usuario.
 - Envía \$id como parámetro.
 - Utiliza el método getRows(definido antes en la clase dataBase) para poder extraer los registros de la base.
 - Emplea la sentencia SQL **SELECT * FROM usuario WHERE idUsuario=?** donde comparara el id que ha sido enviado con los registros en la base.
 - Retorna los datos obtenidos en un arreglo de tipo Usuario (\$ObjUsuario).
 - **createUsuario:** Crea nuevos registros de usuario dentro de la tabla Usuario de la base datos.
 - Envía todos los atributos del objeto Usuario como parámetros.
 - Utiliza el método insertRow(definido antes en la clase dataBase) para poder insertar registros.
 - Emplea la sentencia SQL **INSERT INTO ebbbooks.usuario** para poder llenar en la base los nuevos valores para los atributos de Usuario.

- **readUsuarios:** Enlista todo el contenido que se encuentran registrado en la tabla Usuario.
 - No envía parámetros.
 - Utiliza el método `getRows`(definido antes en la clase `dataBase`) para poder extraer los registros de la base.
 - Emplea la sentencia SQL **SELECT * FROM usuario** para obtener todos los datos registrados en la tabla.
 - Retorna los datos obtenidos en un arreglo de tipo `Usuario($arrayUsuario)`.
- **updateUsuario:** Actualiza los registros dentro de la tabla Usuario.
 - Envía todos los atributos del objeto `Usuario` como parámetros.
 - Utiliza el método `updateRow`(definido antes en la clase `dataBase`) para poder actualizar registros.
 - Emplea una sentencia SQL **UPDATE ebbbooks.usuario SET usuario.contrasena WHERE usuario.idUsuario =?** y reemplaza con nuevos valores los registros que coincidan con el id envía como parámetro.
- **deleteUsuario:** Elimina registros de la tabla Usuario.
 - Envía `$id` como parámetro.
 - Utiliza el método `deleteRow`(definido antes en la clase `dataBase`) para poder eliminar registros.
 - Emplea una sentencia SQL **DELETE FROM ebbbooks.usuario WHERE idUsuario=?** para poder eliminar registros según el id que se envió.

Clase: ColeccionCollector.

Ubicación: Crud la tabla Coleccion y carpeta principal del sitio.

```
1 <?php
2
3 include_once('coleccion.php');
4 include_once('Collector.php');
5 //include_once('login_logout.php');
6 //$idUserio= $_SESSION["id"];
7
8 class ColeccionCollector extends Collector
9 {
10     function guardarColeccion() {
11         $idUserio=$_SESSION["idUserio"];
12         $rows = self::$db->getRows("SELECT * FROM coleccion WHERE fk_usuario LIKE '".$idUserio."'");
13         $arrayColeccion= array();
14         foreach ($rows as $c){
15             $aux = new Coleccion($c['fk_ebook'],$c['fk_usuario'],$c['secuencia']);
16             array_push($arrayColeccion, $aux);
17         }
18         return $arrayColeccion;
19     }
20
21     function contarRows(){
22         $idUserio=$_SESSION["idUserio"];
23         $rows = self::$db->getRows("SELECT * FROM coleccion WHERE fk_usuario LIKE '".$idUserio."'");
24         $arrayColeccion= array();
25         foreach ($rows as $c){
26             $aux = new Coleccion($c['fk_ebook'],$c['fk_usuario'],$c['secuencia']);
27             array_push($arrayColeccion, $aux);
28         }
29         return count($arrayColeccion);
30     }
31
32     function createColeccion($fk_usuario, $fk_ebook, $secuencia){
33         $insertRow = self::$db->insertRow("INSERT INTO ebbbooks.coleccion (fk_usuario, fk_ebook, secuencia) VALUES (?, ?, ?)", array($fk_usuario, $fk_ebook, $secuencia));
34     }
35 }
36 ?>
```

- La clase coleccionCollector incluye la clase Collector(definida anteriormente) y coleccion(objeto) para poder funcionar correctamente.
- La clase coleccionCollector extiende de la clase Collector(que a su vez extiende de dataBase).
- Se crean las funciones principales:
 - **guardarColeccion:** Obtiene los registros que se encuentran en la tabla Coleccion.
 - Obtiene la id del usuario que se encuentre en sesión.
 - Emplea la sentencia SQL **SELECT * FROM coleccion WHERE fk_usuario LIKE** donde comparara el id que ha sido enviado con los registros en la base.
 - Retorna los datos obtenidos en un arreglo de tipo coleccion(\$arrayColeccion)
 - **contarRows:** Obtiene la cantidad de registros que se encuentran en la tabla Coleccion.
 - Obtiene la id del usuario que se encuentre en sesión.
 - Emplea la sentencia SQL **SELECT * FROM coleccion WHERE fk_usuario LIKE** donde comparara el id que ha sido enviado con los registros en la base.
 - Guarda los datos obtenidos en un arreglo de tipo coleccion(\$arrayColeccion)
 - Hace un conteo de los registros en el arreglo y regresa el resultado.
 - **createColeccion:** Crea un nuevo registro en la tabla Coleccion.
 - Emplea la sentencia SQL **INSERT INTO coleccion VALUES** para ingresar un nuevo registro valiéndose de los parámetros enviados.

Clase: CarritoCollector.

Ubicación: Crud la tabla Carrito y carpeta principal del sitio.

```
1 <?php
2 include_once ('Carrito.php');
3 include_once ('Collector.php');
4
5 class CarritoCollector extends Collector {
6     function showCarritosPorUsuario($idUser){
7         $rows = self::$db->getRows("SELECT * FROM carrito WHERE fk_usuario = ?", array($idUser));
8         $arrayCarrito = array();
9         foreach ($rows as $c) {
10             $objCarrito = new Carrito($c['fk_usuario'],$c['fk_ebook']);
11             $arrayCarrito[] = $objCarrito;
12         }
13         return $arrayCarrito;
14     }
15
16     function createCarrito($fk_usuario, $fk_ebook){
17         $insertRow = self::$db->insertRow("INSERT INTO ebbbooks.carrito (fk_usuario, fk_ebook) VALUES (?, ?)", array($fk_usuario, $fk_ebook));
18     }
19
20     function readCarritos(){
21         $rows = self::$db->getRows("SELECT * FROM carrito");
22         $arrayCarrito = array();
23         foreach ($rows as $c){
24             $tmp = new Carrito($c['fk_usuario'],$c['fk_ebook']);
25             $arrayCarrito[] = $tmp;
26         }
27         return $arrayCarrito;
28     }
29
30     function updateCarrito($fk_usuario, $fk_ebook){
31         $insertRow = self::$db->updateRow("UPDATE ebbbooks.carrito SET carrito.fk_ebook = ? WHERE carrito.fk_usuario = ?", array($fk_ebook, $fk_usuario));
32     }
33
34     function deleteCarrito($fk_usuario, $fk_ebook){
35         $deleterow = self::$db->deleteRow("DELETE FROM ebbbooks.carrito WHERE fk_usuario= ? AND fk_ebook= ?", array($fk_usuario, $fk_ebook));
36     }
37
38     function deleteAllCarritos($fk_usuario){
39         $deleterow = self::$db->deleteRow("DELETE FROM ebbbooks.carrito WHERE fk_usuario= ?", array($fk_usuario));
40     }
41 }
42 >>
```

- La clase carritoCollector incluye la clase Collector(definida anteriormente) y carrito(objeto) para poder funcionar correctamente.
- La clase carritoCollector extiende de la clase Collector(que a su vez extiende de dataBase).
- Se crean las funciones principales:
 - **showCarritosPorUsuario:** Obtiene los registros que se encuentran en la tabla carrito que contengan la misma id que se envía.
 - Emplea la sentencia SQL **SELECT * FROM carrito WHERE fk_usuario LIKE** donde comparara el id que ha sido enviado con los registros en la base.
 - Retorna los datos obtenidos en un arreglo de tipo carrito(\$arrayCarrito)
 - **createCarrito:** Crea un nuevo registro en la tabla carrito.
 - Emplea la sentencia SQL **INSERT INTO coleccion VALUES** para ingresar un nuevo registro valiéndose de los parámetros enviados.
 - **readCarritos:** Obtiene todos los registros de la tabla carrito.
 - Emplea la sentencia SQL **SELECT * FROM carrito** lo cual recibe como resultado todos los registros de la tabla.
 - Retorna los datos obtenidos en un arreglo de tipo carrito(\$arrayCarrito)
 - **updateCarrito:** Actualiza los registros dentro de la tabla carrito.

- Envía \$fk_usuario y \$fk_ebook como parámetros.
 - Utiliza el método updateRow(definido antes en la clase dataBase) para poder actualizar registros.
 - Emplea una sentencia SQL **UPDATE ebbbooks.carrito SET carrito.fk_ebook WHERE carrito.fk_usuario=?** y reemplaza con nuevos valores los registros que coincidan con el id envía como parámetro.
- **deleteCarrito:** Elimina un registro de la tabla carrito que cumpla con los parámetros enviados.
 - Envía \$fk_usuario y \$fk_ebook como parámetros.
 - Utiliza el método deleteRow(definido antes en la clase dataBase) para poder eliminar registros.
 - Emplea una sentencia SQL **DELETE FROM ebbbooks.carrito WHERE fk_usuario= ? AND fk_ebook= ?** para poder eliminar registros según los id que se envió.
- **deleteAllCarritos:** Elimina todos los registros de la tabla carrito que cumplan con el parámetro enviado.
 - Envía \$fk_usuario como parámetro.
 - Utiliza el método deleteRow(definido antes en la clase dataBase) para poder eliminar registros.
 - Emplea una sentencia SQL **DELETE FROM ebbbooks.carrito WHERE fk_usuario= ?** para poder eliminar registros según el id que se envió.

4. Estadísticas del Sitio Web

Para llevar un seguimiento de quienes accedan al sitio, se optó por el servicio de estadísticas e informes de sitios web de Google, llamado Google Analytics.

4.1 Google Analytics

Google Analytics permite saber quién visita el sitio, identificando así el buscador, dispositivo y el lugar del mundo desde donde lo hace. Junto con los informes de tráfico, muestra también qué llevó a ese usuario al sitio para, de esta forma, mejorar la experiencia del usuario.

Ciertas características como el Flujo de Visualización o el Rastreo de Eventos, ayudan a monitorizar las distintas páginas de un sitio para así saber con cuáles el usuario interactúa más, en las que permanece más tiempo, y en las que decide dejar el sitio.

La herramienta llamada In-Page Analytics permite hacer una evaluación visual de lo que los usuarios miran más y qué es lo que más les gusta del sitio, para así poder replantear o mejorar las estrategias de marketing de las empresas que emplean este servicio.



4.1.1 ¿Cómo se implementa?

Para implementar el servicio de Google Analytics en un sitio web, se necesita incluir un código JavaScript en las páginas que se deseen analizar, llamado GATC (Google Analytics Tracking Code o Código de Rastreo de Google Analytics). Éste código carga archivos del servidor de Google que monitorean la página y almacenan informes en la cuenta de usuario de Google Analytics que tenga autorización para verlos.

Este código carga un archivo de aproximadamente 18 KB conocido como "ga.js" que se descarga al inicio de la visita y se almacena en caché para el resto de la sesión del usuario. Si el usuario visitara otros sitios que también emplean este servicio, se utiliza el mismo archivo almacenado en caché para el monitoreo de estos ya que todos utilizan el mismo archivo maestro de Google, por lo que el tiempo de carga de las páginas no se ve afectado.

Actualmente hay otras opciones de implementación de este servicio, como lo son el uso de widgets o gadgets en el sitio web, tal y como lo hacen Blogger o Wordpress.

5. Instalación del Sistema

5.1 Requerimientos de Software

Estos son algunos de programas que serán necesarios para acceder a todos los servicios y funciones del sitio:

5.1.1 Servidor Web



Apache Web Server es un servidor HTTP de código abierto, desarrollado por Apache Software Foundation y que funciona en todos los Sistemas Operativos (Linux, Windows o Mac) para procesar una aplicación desde un servidor, y así luego realizar la conexión con un cliente para que éste pueda compilar y ejecutar la aplicación recibida a través de un navegador web.

5.1.2 Base de Datos



MySQL es un Sistema de Gestión de Base de Datos, desarrollado por Sun Microsystems, que permite el almacenamiento, edición y extracción de información o datos guardados en una Base de Datos a través de comandos o mediante una interfaz gráfica.

5.1.3 Lenguaje de Programación



PHP es un lenguaje de programación de código abierto, desarrollado por PHP Group, especialmente para el desarrollo web. Es un lenguaje rápido, sencillo y pragmático, que se encuentra tanto en el blog más pequeño como en los sitios web más populares del mundo.

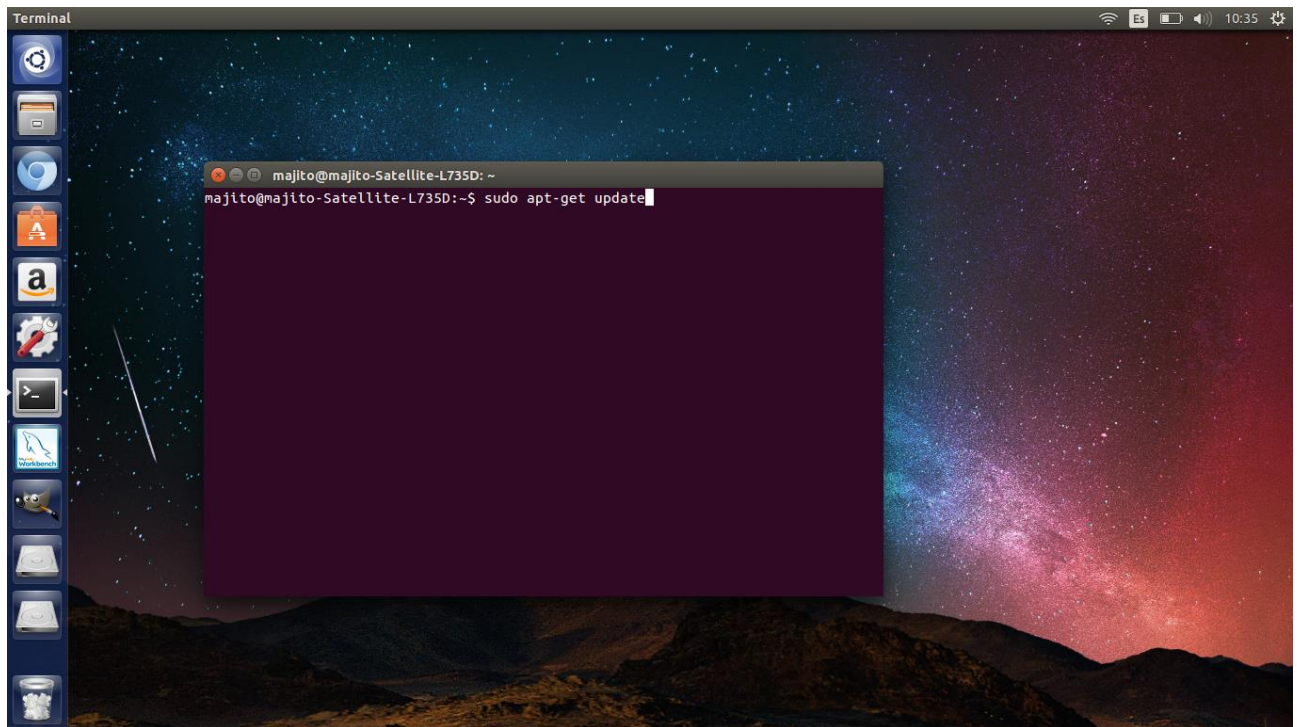
5.2 Instalación y configuración de paquetes necesarios

Por efectos de prueba y revisión de sintaxis, se realizan pruebas en Linux Ubuntu V. 14.04.1 LTS y en Microsoft Windows 7 y 8 (con Appserv).

5.2.1 Instalación en Ubuntu V. 14.04.1 LTS

5.2.1.1 Actualización de repositorios

1. Para empezar, se debe asegurar de tener actualizados los repositorios con el siguiente comando:



2. Y luego de esto, se los instala si los hay, de la siguiente forma:

```
majito@majito-Satellite-L735D: ~  
majito@majito-Satellite-L735D:~$ sudo apt-get upgrade  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
Calculating upgrade... Done  
The following packages will be upgraded:  
  python3-distupgrade ubuntu-release-upgrader-core ubuntu-release-upgrader-gtk  
  unity-settings-daemon  
4 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.  
Need to get 625 kB of archives.  
After this operation, 24,6 kB disk space will be freed.  
Do you want to continue? [Y/n]
```

3. Después de darle el permiso necesario, ya se tendrá los paquetes y repositorios actualizados y se podrá proceder a la instalación de los servicios que se necesitan que son: PHP, Apache y MySQL.

5.2.1.2 Instalación del librerías para PHP

1. Para la instalación de PHP, se utiliza el siguiente comando:

```
majito@majito-Satellite-L735D: ~  
majito@majito-Satellite-L735D:~$ sudo apt-get install php5 libapache2-mod-php5 p  
hp5-cli php5-mysql  
[sudo] password for majito:
```

Se instala la versión 5 se PHP al ser esta la última versión funcional y con soporte completo.

‘libapache2-mod-php5’: este paquete crea un módulo de PHP5 para el servidor web Apache.

‘php5-cli’: es un paquete que provee de un intérprete de línea de comandos (/usr/bin/php5), útil para hacer pruebas con los scripts de PHP (archivos PHP). Para una explicación del comando ‘php5-mysql’, ir a la sección **5.5 Librerías para la comunicación de PHP con MySQL**.

5.2.2 Instalación en Windows 7 y 8

5.2.2.1 Descarga de Appserv

En Windows se utilizan los siguientes componentes para el funcionamiento de Ebbbooks.

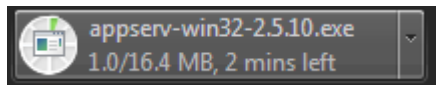
- PHP5
- MySql
- Apache 2.2
- phpMyAdmin-2.10.3

Es posible conseguir e instalar todos estos paquetes por separado. Pero “AppServ Open Project” ofrece una alternativa más sencilla a esto conocido como “Appserv”. Se puede encontrar en la siguiente dirección: <http://www.appservnetwork.com/>



Aquí se mostraran noticias sobre el proyecto, incluyendo las mas recientes versiones del mismo junto con enlaces para su descarga. Seleccionar la versión de AppServ 2.5.10.

Esto llevara a una página donde el programa se descargara.



5.2.2.2 Instalación de Appserv

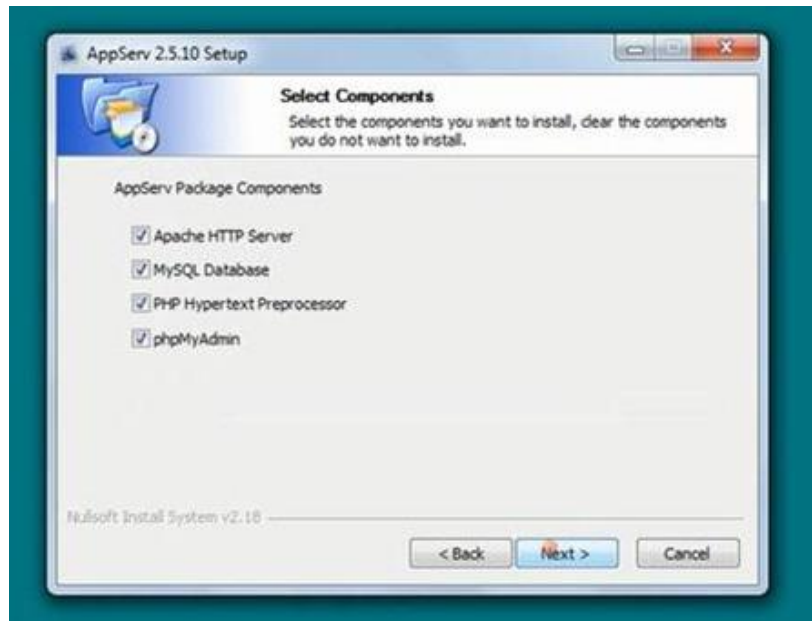
1. Ejecutar el archivo de instalación y aceptar licencia.



2. Seleccionar la carpeta de destino, por defecto se selecciona el disco C



3. Seleccionar las todas opciones de instalación



4. Llenar los campos que se piden a continuación:

- Server Name: El nombre del servidor, generalmente localhost.
- Administrator's email Address : Direccion del correo del administrador
- Apache Http port: Puerto a usarse.



5. Se ingresa la contraseña para el super usuario "root". También se activa la opción de habilitar InnoDB, el cual es un motor para la gestión de datos.



6. Seleccionar Install y esperar a que termine la instalación. Una vez finalizada, click en Finish para activar los servicios.



5.2.2.3 Habilitar la extensión PDO

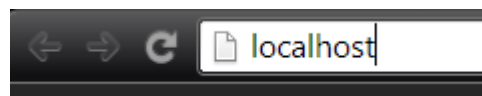
1. En la barra de inicio escribir “ini” y seleccionar la opción de editar el archivo php.ini.
2. Buscar y descomentar(remove el ;) las siguientes líneas:
 1. extension=php_pdo.dll
 2. extension=php_pdo_mysql.dll
 3. extension=php_pdo_sqlite.dll
 4. extension=php_mysql.dll
 5. extension=php_mysqli.dll
3. Guardar cambios.
4. Reiniciar el servicio de php.

5.2.2.4 Habilitar el motor InnoDB

1. Abrir el archivo my.ini (\AppServ\mysql)
2. Buscar y descomentar las siguientes líneas
 - default-storage-engine=INNODB
 - skip-innodb
3. Guardar cambios
4. Reiniciar Mysql

5.2.2.5 Instalar la base de datos ebbbooks

1. Dentro de un navegador Web, escribir el nombre del servidor. Ej: localhost



2. Seleccionar phpMyAdmin e ingresar el nombre y contraseña del súper usuario root


The AppServ Open Project - 2.5.10 for Windows

phpMyAdmin Database Manager Version 2.10.3
PHP Information Version 5.2.6

About AppServ Version 2.5.10 for Windows
AppServ is a merging open source software installer package for Windows includes :

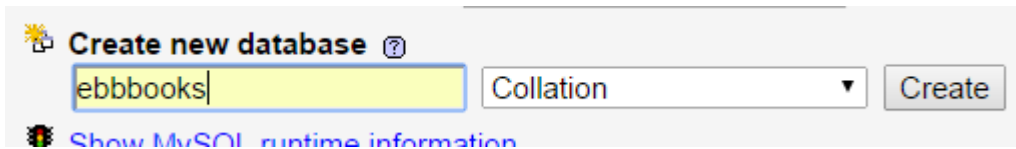
- Apache Web Server Version 2.2.8
- PHP Script Language Version 5.2.6
- MySQL Database Version 5.0.51b
- phpMyAdmin Database Manager Version 2.10.3

• ChangeLog
• README
• AUTHORS
• COPYING
• Official Site : <http://www.AppServNetwork.com>
• Hosting support by : <http://www.AppServHosting.com>

Change Language :  

 Easy way to build Webserver, Database Server with AppServ :-)

3. Crear una nueva base de datos con el nombre “ebbbooks”

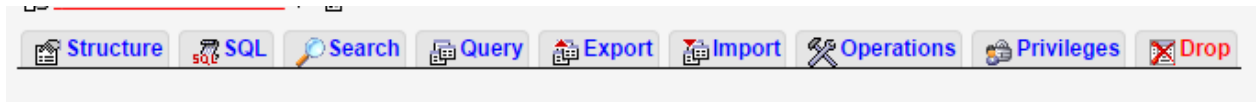


Create new database ⓘ

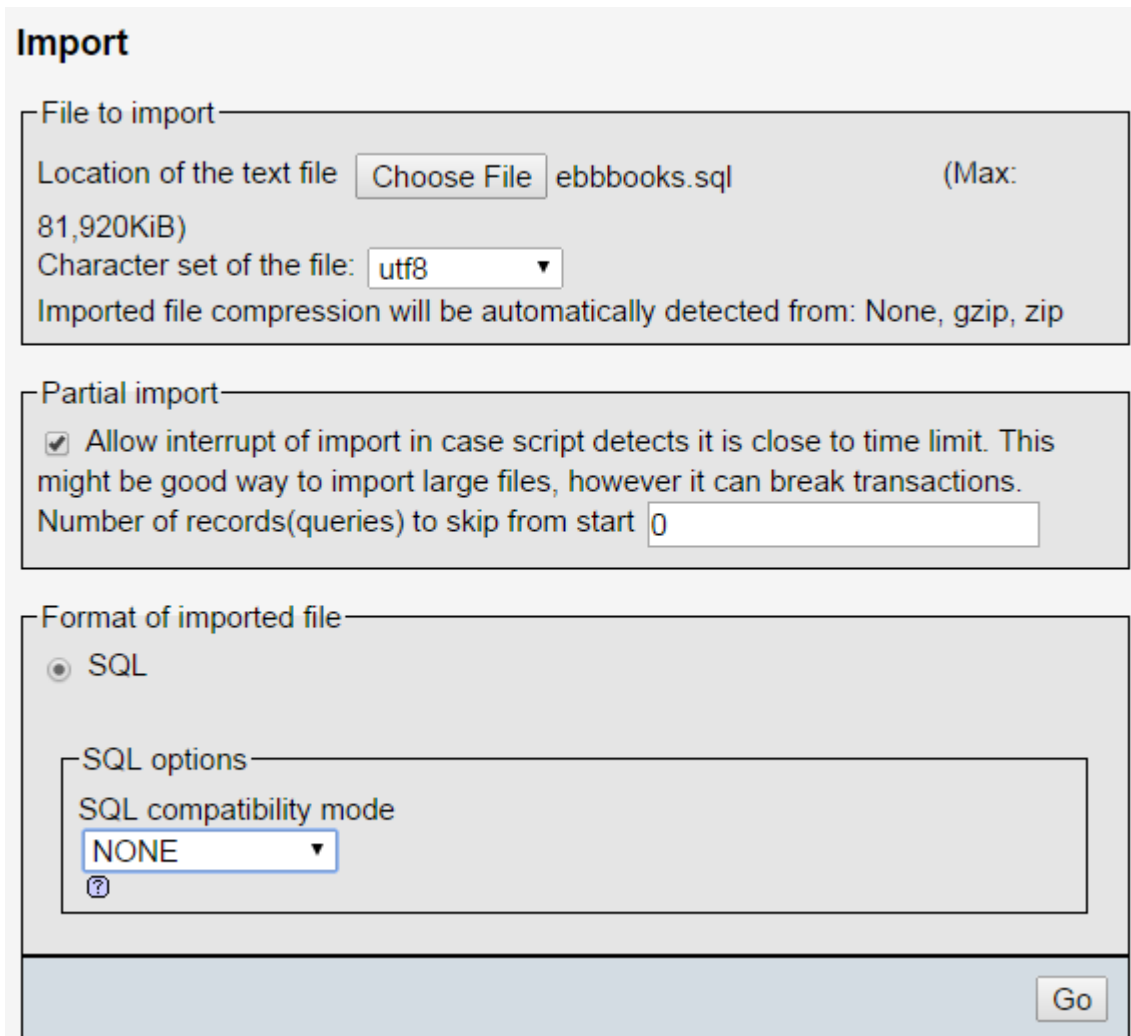
ebbbooks Collation ▼ Create

Show MySQL runtime information ⓘ

4. Seleccionar la opción de Importar



5. Seleccionar el archivo “ebbbooks.sql” del repositorio y dar clic en “Go”. Una vez hecho esto la base de datos esta lista para ser usada



Import

File to import

Location of the text file ebbbooks.sql (Max: 81,920KiB)

Character set of the file: utf8 ▼

Imported file compression will be automatically detected from: None, gzip, zip

Partial import

☒ Allow interrupt of import in case script detects it is close to time limit. This might be good way to import large files, however it can break transactions.

Number of records(queries) to skip from start

Format of imported file

☒ SQL

SQL options

SQL compatibility mode

NONE ▼ ⓘ

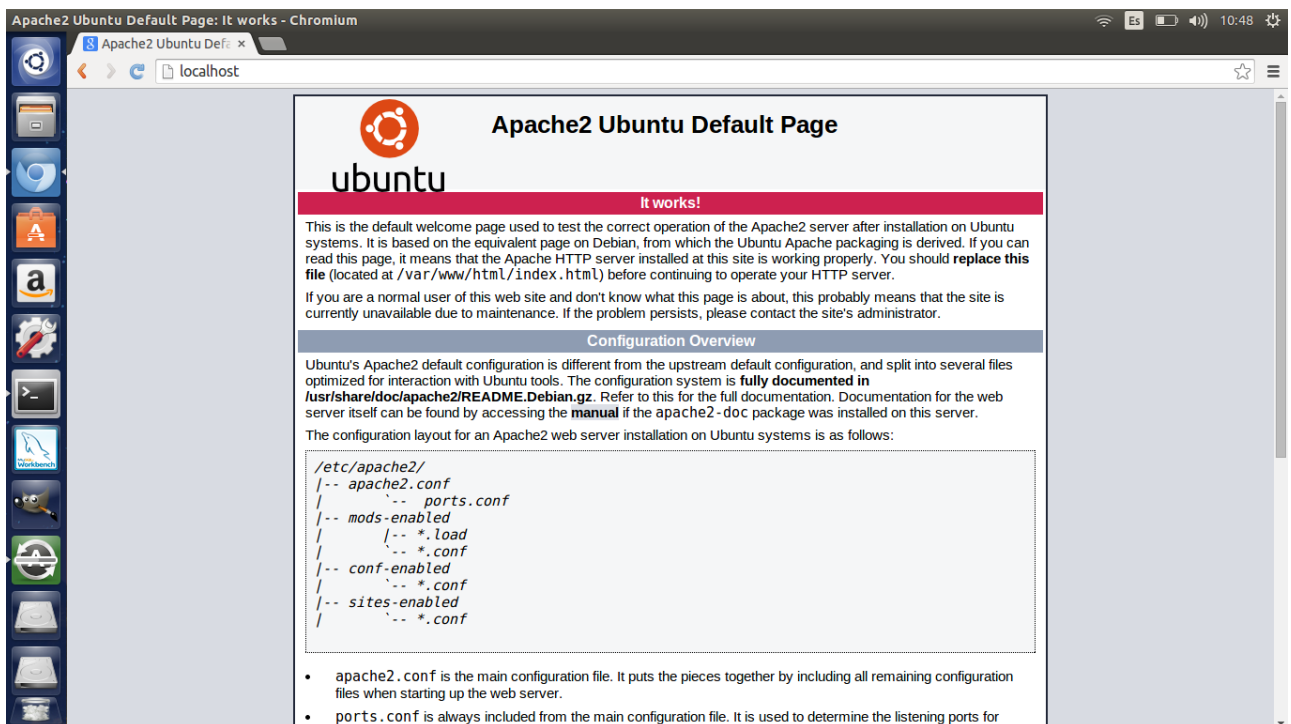
5.3 Servicio Web Apache

5.3.1 Instalación en Ubuntu V. 14.04.1 LTS

1. Se empieza la instalación del servidor Apache con el siguiente comando en la terminal:

```
majito@majito-Satellite-L735D: ~  
majito@majito-Satellite-L735D:~$ sudo apt-get install apache2
```

2. Luego dar el permiso respectivo de instalación. El servicio se iniciará automáticamente cada vez que se inicie la máquina. El directorio en el que se deberá guardar el sitio web será `/var/www`.
3. Para comprobar que el servicio funciona, primero debemos dar permisos a este directorio con el comando `'sudo chmod 777 -R /var/www'`.
4. Luego de esto, en el navegador que se desee, se coloca `http://localhost` o simplemente `localhost` y se debería mostrar la siguiente página con información sobre la versión instalada de Apache:



5.4 Servicio de Base de Datos MySQL

5.4.1 Instalación en Ubuntu V. 14.04.1 LTS

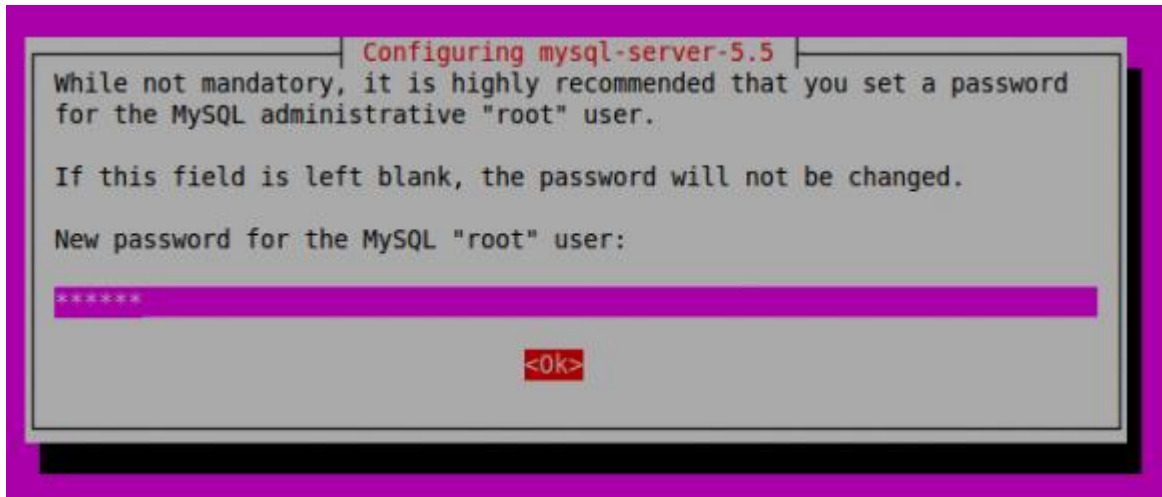
1. El proceso de instalación del servicio de gestión de base datos MySQL empieza con el siguiente comando en la terminal:

```
majito@majito-Satellite-L735D: ~  
majito@majito-Satellite-L735D:~$ sudo apt-get install mysql-server mysql-client  
libmysqlclient-dev
```

‘mysql-server’ y ‘mysql-client’ son paquetes vacíos que se encargan de buscar la última versión estable de MySQL disponible para este sistema. Se los coloca de esta manera en caso de que no se conozca cuál es la versión que se necesita para cliente y servidor.

‘mysqlclient-dev’ es un paquete que incluye librerías de desarrollo y archivos header.

2. Se mostrará una pantalla que nos pedirá la clave del super usuario ‘root’:



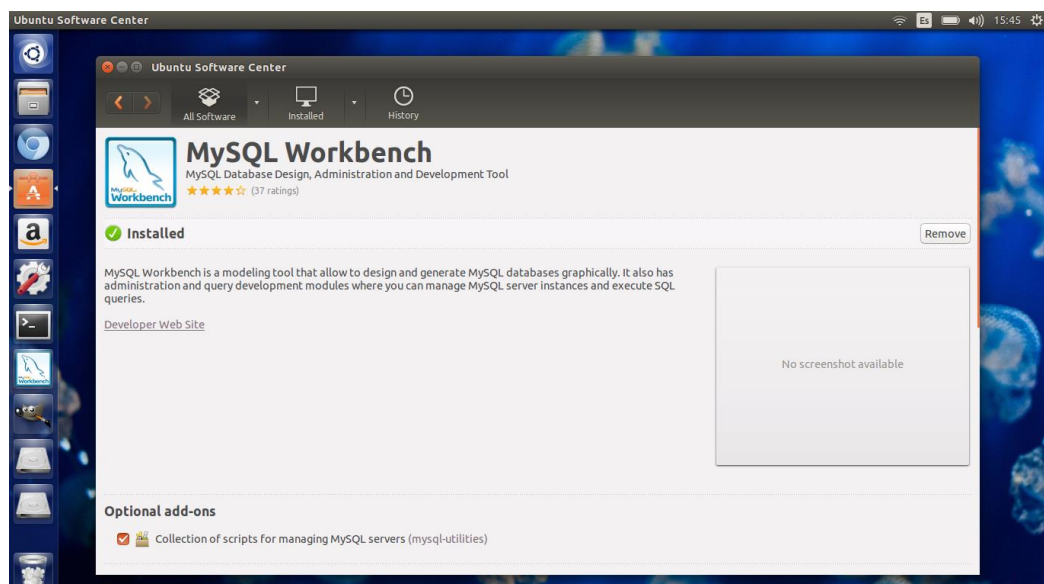
servicio se iniciará siempre que se inicie la máquina y siempre que se desee acceder a él se necesitarán estas credenciales.

5.4.1.1 Instalación de MySQL Workbench

Para efectos de facilitar el manejo de la base de datos del sitio, se decidió instalar también MySQL Workbench. Esta herramienta proporcionará un ambiente más amigable para la creación, administración y manejo de los datos dentro del sitio.

Su instalación no tiene complicación alguna, ya que se puede encontrar en el mismo Centro de Software de Ubuntu:

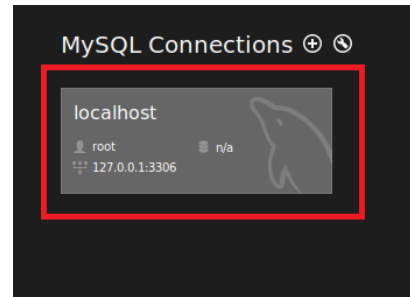
1. Si se utiliza la barra de búsqueda dentro de esta aplicación y se teclea “MySQL Workbench”, aparecerá esta opción, que es la que se debe instalar:



Una vez instalada, simplemente se ingresa con las credenciales de 'root' y ya se puede empezar a utilizar.

Para poder manejar la base de datos del sitio, se debe seguir el siguiente proceso:

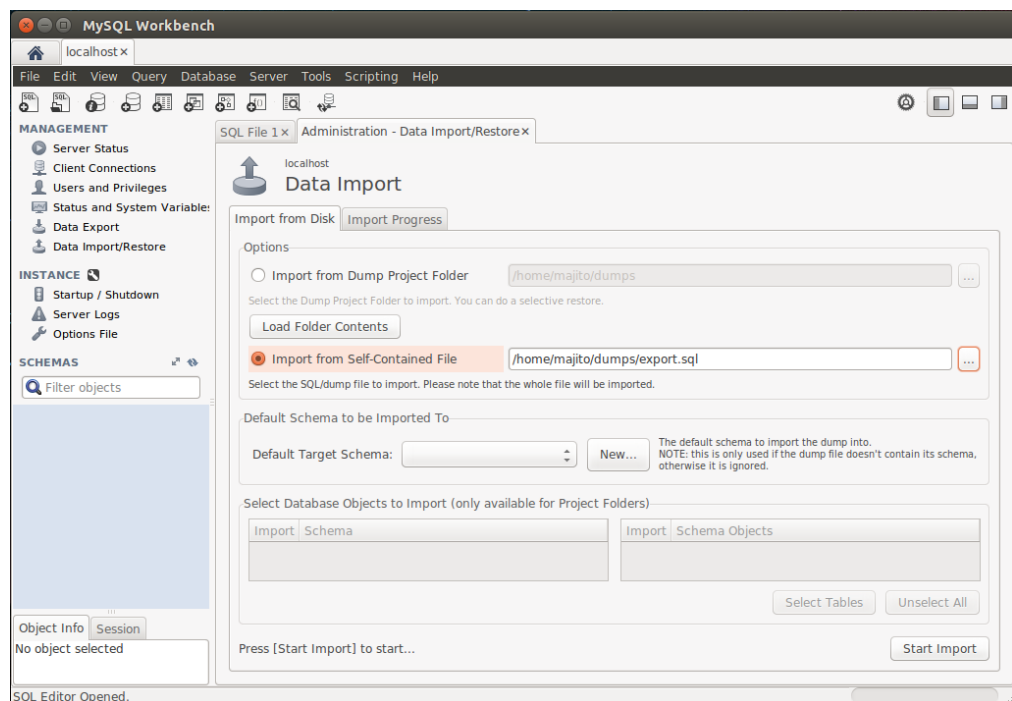
1. Ingresar a el programa MySQL Workbench y seleccionar la conexión principal que se muestra:



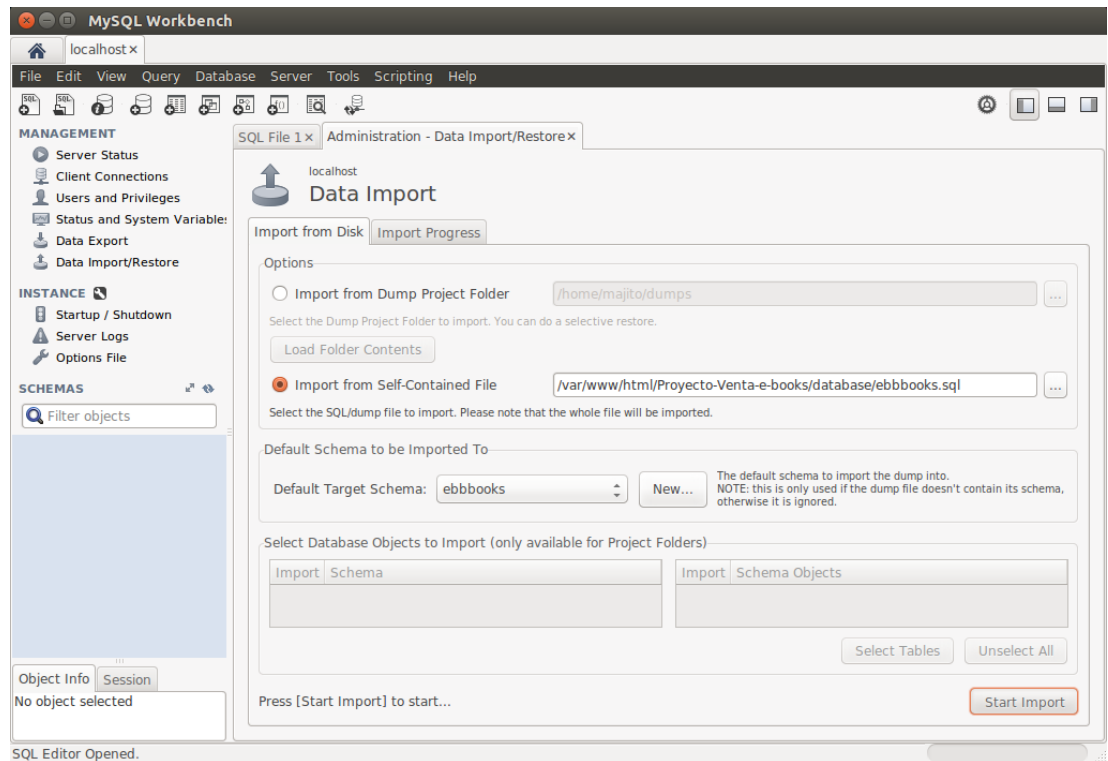
2. Se pedirá ingresar con las credenciales de root, y luego se dará clic en 'ok'.



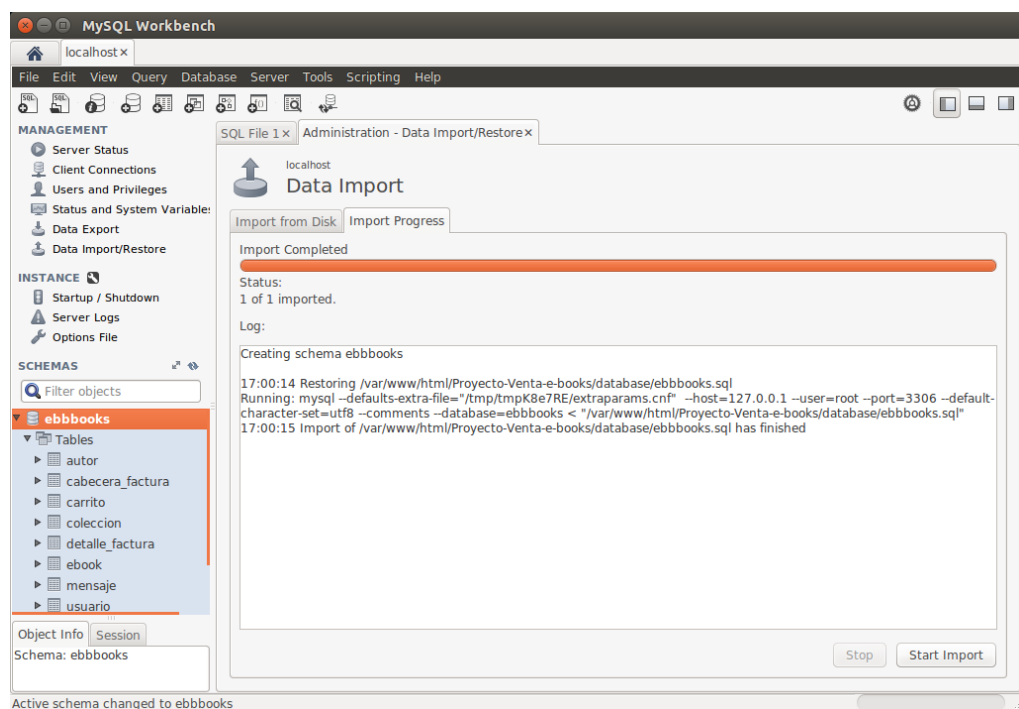
3. En la ventana principal luego de haber iniciado la conexión, se debe acceder al menú se 'Server' y seleccionar la opción 'Data Import', a lo que aparecerán estas opciones luego de pedirnos nuevamente nuestras credenciales de root:
4. Se selecciona la opción de 'Import from Self-Contained File' y a la derecha se busca el archivo con la extensión .sql que contenga las tablas y datos del sitio.



5. Luego de encontrar el archivo en la parte inferior se selecciona la opción 'New' para crear un nuevo esquema en base a ese archivo, en este caso se lo llamará 'ebbbooks' por ser este el nombre del sitio en mención.
6. Teniendo todo esto listo, se dará clic en 'Start Import'.



7. Este proceso tardará unos segundos. Luego de esto, debemos actualizar los esquemas en la sección izquierda junto a la etiqueta de SCHEMAS.
8. Finalmente, el esquema creado se mostrará en la lista de esquemas en la parte inferior izquierda y dándole doble clic al nombre estaremos listo para usarlo.



5.5 Librerías para la comunicación de PHP con MySQL

La librería 'php5-mysql' instalada a la vez que PHP5 es la librería que permite la comunicación de este lenguaje con el servicio de MySQL. Provee módulos de conexión directa a la base de datos MySQL desde un script de PHP. Incluye el módulo genérico 'mysql' para poder conectarse a cualquier versión de MySQL, el módulo mejorado 'mysqli' para la versión 4.1 o posterior de MySQL y finalmente el módulo 'pdo_mysql' para utilizarlo con la extensión Data Object de PHP (extensión de una capa de abstracción de acceso a datos para PHP5, sirve para hacer consultas y obtener datos de, en este caso, MySQL).