ACERCA DEL AUTOR

Santiago Aguirre Pérez es programador y desarrollador web, además de entusiasta de la tecnología.

Trabajó durante cinco años en servicio técnico, y estudió las carreras de Comunicación Social y Desarrollo Web en la Universidad de La Matanza. Es desarrollador en Java, PHP, Python, JavaScript, y en tecnologías como Angular y Bootstrap. Actualmente se desempeña como redactor para esta editorial y como desarrollador web en empresas de consultoría tecnológica.

PRÓLOGO

PHP es, hoy en día, uno de los lenguajes más utilizados y demandados en el mundo del desarrollo, tanto en solitario, trabajando bajo patrones de desarrollo; como en conjunto con sus frameworks y librerías; junto con otras tecnologías; como así también bajo el uso de distintos CMS, entornos para la creación de sitios y sistemas, que permiten utilizar menor cantidad de código. Lleva muchos años al frente del desarrollo web, y la aparición de tecnologías nuevas y de diversas características, además de su comunidad creciente, parecen implicar que seguirá así por más tiempo.

Si deseas aprender a desarrollar en PHP de manera completa, con diversas herramientas y en distintos entornos, esta colección es para ti.

SOBRE ESTA OBRA

En esta colección estudiarás todos los aspectos avanzados del desarrollo en PHP, trabajando bajo el patrón de programación Modelo Vista Controlador y utilizando conceptos como relaciones entre tablas, programación orientada a objetos y elementos de las últimas versiones del lenguaje. Crearás un sistema base, sobre el cual irás agregando nuevas e interesantes características en cada capítulo y en cada entrega de la obra, así que no te lo pierdas.

En cada volumen, trabajarás con distintos aspectos de PHP y crearás funcionalidades nuevas, que siempre son demandadas y suelen requerir experiencia en el ámbito laboral.

Este aprendizaje te servirá parar manejarte con mayor velocidad, y con conocimiento que te respalde y te permita llevar a cabo cualquier tarea fácilmente.

Aprenderás a utilizar sesiones, sistemas de login y contraseñas encriptadas; trabajarás con librerías de desarrollo en PHP como PHPMailer, la librería por excelencia del lenguaje para el envío de correos electrónicos; verás cómo utilizar pasarelas de pago y sistemas de integración de tarjetas de crédito y cobros en línea; conocerás el manejo seguro de productos y envío de newsletters; crearás sistemas modernos basados en aplicaciones reales, con usuarios y clientes; y podrás gestionar sus compras.

- **Parte 1:** Trabajarás con PHP para crear un sistema base, el cual utilizarás en los distintos volúmenes de esta colección. Para comenzar, vas a desarrollar el sistema de login, que permitirá acceder al panel de control.
- **Parte 2:** Avanzarás en la creación del front-end, generando una interfaz interesante que permita vender tus productos al cliente final, desde la cual podrán acceder a distintos pagos y obtener tu contenido.

- **Parte 3:** Utilizarás PHP para crear un sistema API que te permita aprovechar todo el potencial de AJAX, las ventajas de JavaScript asíncrono, que divide la aplicación en front-end y back-end, y crearás un sistema de calificaciones y opiniones en el sistema.
- **Parte 4:** Aprenderás a trabajar más en profundidad con PHPMailer y el envío de correos electrónicos, así como también a realizar newsletters, mejorar todo el sistema para dejarlo listo para producción y crear un sistema documental.

Parte 1

PDO. ENCRIPTACIÓN. SISTEMA DE LOGIN

Introducción
Conceptos iniciales
Interfaz
Perfil
Cuentas

1

INTRODUCCIÓN

El objetivo de esta colección es mostrarte cómo trabajar con PHP de manera profesional, avanzada, y bajo entornos y patrones de desarrollo modernos, como los principios SOLID y el patrón de diseño de sistemas Modelo Vista Controlador.

1.1 EL LENGUAJE

PHP es uno de los lenguajes más conocidos en el mundo del desarrollo orientado a la Web. Creado como un lenguaje del lado del servidor para programar sitios dinámicos, su popularidad fue creciendo desde el momento en que salió al mercado.

Las distintas mejoras que obtuvo, y los diversos programas que se crearon con él y se popularizaron en el mundo entero le dieron una inmensa difusión y generaron una enorme comunidad que lo utiliza.

Figura 1.1. PHP es uno de los lenguajes más populares del mercado.

Además de ser una herramienta ampliamente distribuida, PHP se caracteriza por ser muy versátil, y por poseer muchas alternativas entre sus frameworks y librerías. Desde Laravel, uno de los entornos de trabajo más populares; pasando por Symphony, una opción muy interesante y utilizada en este lenguaje; hasta otros recursos como CodeIgniter, conocido como el peso pluma o framework ligero de PHP. También están los CMS, creados por lo general en PHP debido a su orientación a la Web. Se trata de programas que permiten desarrollar sitios web sobre la base de distintos contenidos, como sucede con Wordpress, Drupal o Magento, herramientas destinadas a instalar y gestionar el sitio sin necesidad de trabajar con código en todos los casos, ya que el sistema viene preparado y ready-to-work.

1.2 HERRAMIENTAS Y CONOCIMIENTOS PREVIOS

Esta colección está pensada para aquellas personas que cuentan con conocimientos básicos o intermedios en PHP, es decir, que conocen la sintaxis del lenguaje, saben crear o interpretar condicionales, bucles y variables, modificar o leer funciones, así como también escribirlas desde cero. También se requiere conocer el paradigma de Programación Orientada a Objetos, al menos en forma básica, entendiendo qué es una clase, qué son las interfaces y traits, qué es la herencia y qué son los modificadores de acceso o niveles de acceso en clases. En este capítulo introductorio, verás dónde puedes obtener este conocimiento si aún no lo tienes.

1.3 ¿QUÉ DEBO SABER?

Además de tener cierta noción sobre PHP y entender los conceptos antes mencionados, se recomienda conocer sobre la Web en general; tener una idea de cómo está creada y formada; saber qué es un servidor y un cliente; manejar los conceptos de front-end y back-end; saber qué es HTML5, CSS3 y JavaScript; y poder utilizar, al menos un poco, estos lenguajes.

1.4 PHP

Si no conoces este lenguaje y te gustaría iniciarte como desarrollador back-end o full stack, es aconsejable comenzar por el libro PHP7, de esta misma editorial, donde verás conceptos clave para introducirte en el desarrollo en general y en PHP en particular.

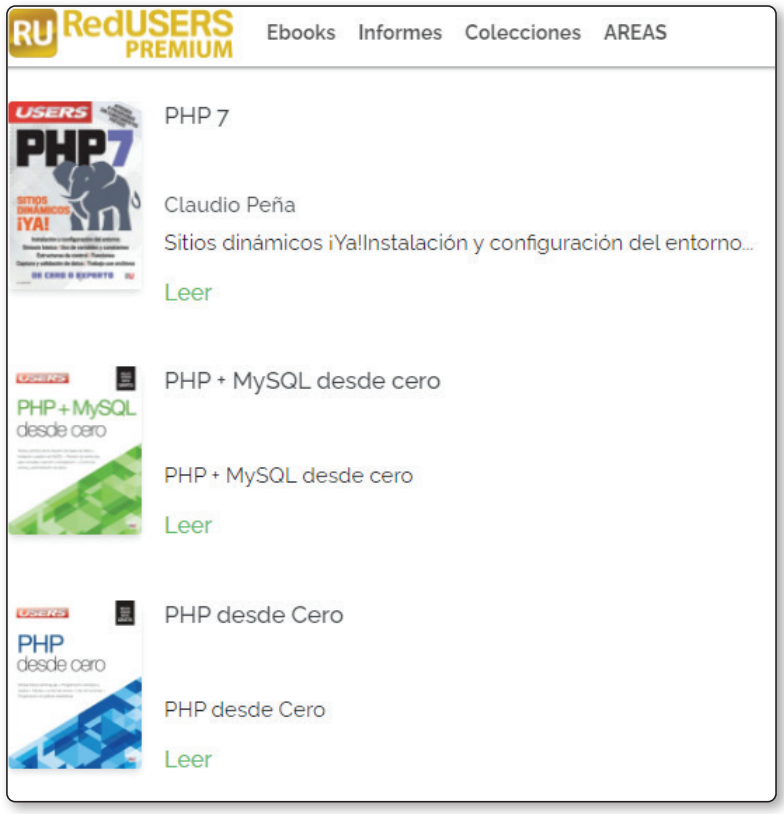


Figura 1.2. Obras de PHP desde cero en RedUSERS.

Para aprender PHP desde cero si no tienes conocimientos sobre este lenguaje de programación, RedUSERS cuenta con libros que te permitirán introducirte a este tema. Por otro lado, si ya conoces un poco sobre PHP y deseas profundizar tus bases con elementos avanzados, también hay disponibles colecciones con esta clase de contenido.

PHP7 es un volumen que te permitirá aprender conceptos muy importantes para cualquier desarrollador, y puedes tomarlo como puntapié inicial del camino para convertirte en uno de ellos.

En caso de que ya tengas conocimientos sobre el lenguaje, sepas cómo es su sintaxis y comprendas los elementos básicos, puedes pasar al siguiente nivel y aprender algunos aspectos más avanzados, como la Programación Orientada a Objetos.

Figura 1.3. Obras de PHP avanzado en RedUSERS.

En esta colección de tres volúmenes, verás conceptos más complejos de PHP, como creación de clases, constructores, campos de clase, métodos accesorios, herencia, abstracción, interfaces, y mucho más. Este roadmap, o camino de tres colecciones, te dará un gran conocimiento sobre el mundo del desarrollo que te permitirá crear cualquier tipo de sitio web.

Una vez que hayas leído las tres colecciones –PHP7, Programación orientada a objetos en PHP y PHP Avanzado–, puedes continuar aprendiendo un framework como Laravel, en RedUSERS Premium, donde encontrarás un libro completo introductorio, con conceptos básicos y avanzados, así como una colección completa del framework que integra otras tecnologías.

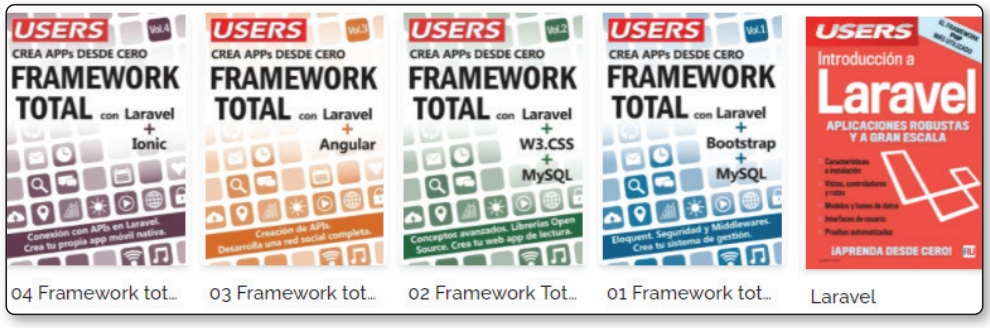


Figura 1.4. Laravel en RedUSERS.

1.5 LENGUAJE DE MARCADO HTML5

HTML es un lenguaje de etiquetas, cuya sigla en inglés proviene de HyperText MarkUp Language, o Lenguaje de Marcado de Hipertexto; fue creado para almacenar información dentro de sus etiquetas definidas. No se trata de un lenguaje de programación, sino de un lenguaje estructurado, cuyo objetivo es guardar información por medio de sus etiquetas, que se escriben con los signos <>. Dentro de ellos, se debe colocar el nombre de las etiquetas que se desea utilizar, y entre la de apertura y la de cierre, incluir la información por guardar.

El lenguaje es una de las bases más importantes para la creación de sitios web en la actualidad, siendo una de las herramientas principales que reemplazó al sistema de Adobe Flash Player. Su estándar es definido por el WWW Consortium, organismo que se dedica a la revisión y el mantenimiento del lenguaje, junto con otras tecnologías ligadas al desarrollo web.

1.6 HOJAS DE ESTILO EN CASCADA–CSS

El lenguaje CSS es un estándar creado para dar estilo visual a la información almacenada en etiquetas HTML. Así como las etiquetas HTML se encargan de almacenar información con su notación, no tienen como objetivo mostrar la información con distintos estilos visuales, sino solo en texto plano, con cierta semántica.

CSS, o Cascade Style Sheets (hojas de estilo en cascada), es un lenguaje creado para dar estilo a los elementos HTML, y así lograr interfaces visuales que pueden renderizarse en un navegador, de forma visualmente agradable.

1.7 JAVASCRIPT

JavaScript es el lenguaje estándar de programación de los navegadores web. Es un dialecto o derivado del estándar ECMAScript, el cual fue desarrollado en 1996, con inspiración en lenguajes como Java y C.

Este lenguaje es un estándar dentro de los navegadores porque todos poseen un motor de intérprete para él, que les permite ejecutar las instrucciones de los programas creados en JavaScript. JavaScript permite trabajar con HTML y CSS en conjunto, para crear páginas web dinámicas que pueden realizar todo tipo

de funcionalidades. Esto hace que sea uno de los lenguajes más solicitados en el mercado actual.

Además de poder utilizarse en navegadores, existen marcos de trabajo como Node que permiten su uso para la creación de servidores y programas fuera del mundo de los navegadores web, algo que será útil para la instalación de elementos de Bootstrap en esta colección.

Figura 1.5. La programación orientada a objetos es un concepto muy importante que todo desarrollador debe saber.

1.8 ¿DÓNDE APRENDER TODO ESTO?

El punto de partida en el cual te basarás para adquirir o reforzar conocimientos en uno o más de los temas mencionados es el sitio web RedUSERS Premium: **https://premium.redusers.com**. Con la experiencia y las décadas de evolución del ecosistema de la información, RedUSERS Premium cuenta con manuales, e-books y guías necesarias para potenciar tu conocimiento básico y así entender a la perfección esta obra.

1.8.1 HTML5 y CSS

RedUSERS Premium tiene varias obras orientadas al aprendizaje a fondo, y desde cero, de los lenguajes de la Web HTML5 y CSS.

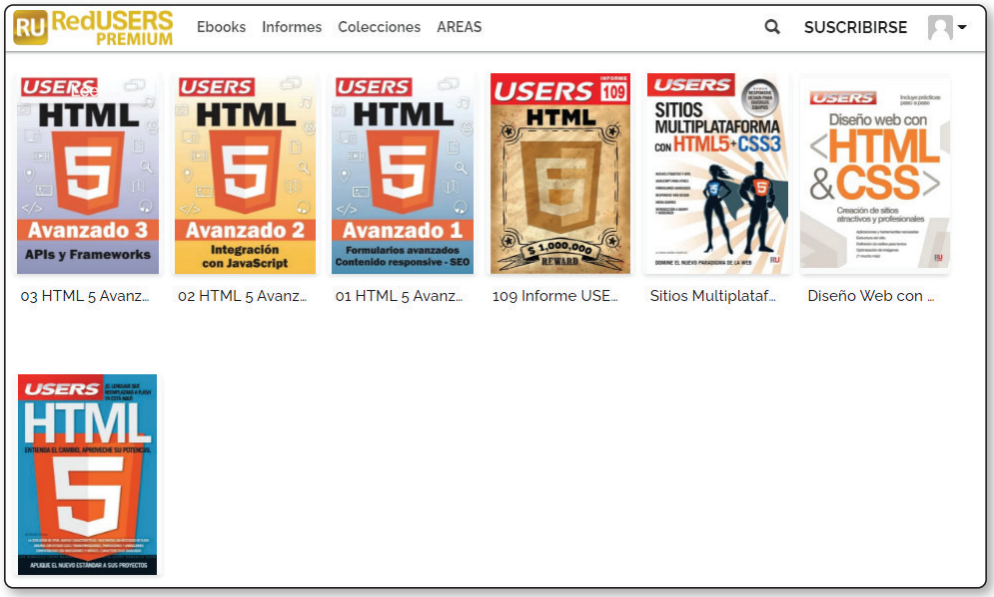


Figura 1.6. HTML5 y CSS en RedUSERS.

Figura 1.7. Sitios Multiplataforma con HTML5 y CSS3 en RedUSERS.

1.8.2 JavaScript

El lenguaje de programación de la Web es uno de los más demandados en la actualidad, y RedUSERS pone a tu disposición varias obras que te serán de utilidad para aprenderlo, tanto si tienes conocimientos como si partes desde cero.

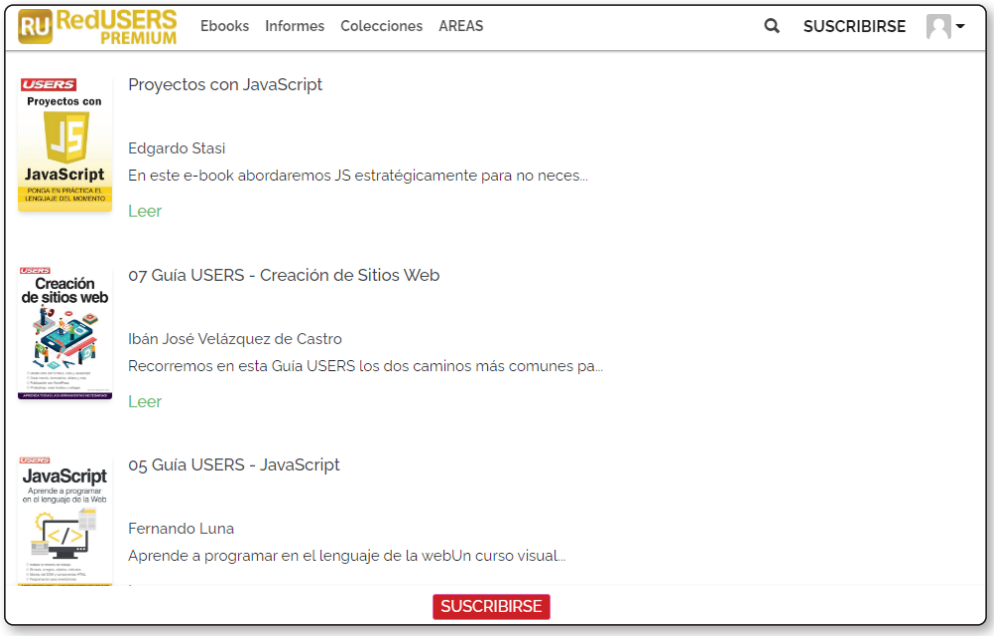


Figura 1.8. JavaScript en RedUSERS.

1.9 ACTIVIDADES

A continuación se presentan las preguntas que deberías saber responder para considerar aprendido el capítulo.

1.9.1 Test de autoevaluación

1. ¿Qué es PHP?

2. ¿Qué es HTML y qué es CSS?

3. ¿Qué es JavaScript y en dónde se ejecuta?

4. ¿Qué conceptos es necesario conocer para avanzar con esta colección?

5. ¿Qué frameworks de PHP conoces?

6. ¿Qué se recomienda conocer antes de trabajar con un framework?

2

CONCEPTOS INICIALES

PHP ha sido, desde sus comienzos, uno de los lenguajes de programación más demandados del mundo, y lo sigue siendo en la actualidad. Es utilizado para generar todo tipo de aplicaciones web, desde sistemas CMS hasta páginas dinámicas como Ecommerce, tiendas electrónicas, y más. Su versatilidad para crear servidores web lo ha llevado a ser el lenguaje más difundido dentro de Internet.

2.1 PHP

PHP es un lenguaje que permite crear toda clase de programas para la Web. Se lo utiliza dentro de sistemas para páginas dinámicas, permite crear APIs que conectan servicios entre sí, y su versatilidad para el uso orientado a objetos o mediante funciones lo ha llevado a ser el elegido para la mayoría de los sistemas en Internet. En entregas anteriores, puedes acceder a una introducción al lenguaje de programación de la Web, donde se te iniciará en la sintaxis de PHP, el uso de bucles, condicionales, funciones, variables, y otras características interesantes. Esto te permitirá aprender lo básico sobre el lenguaje para poder comenzar a trabajar creando tus propias páginas web. Si deseas consultar dicha obra, puedes hacerlo desde el siguiente *enlace*.

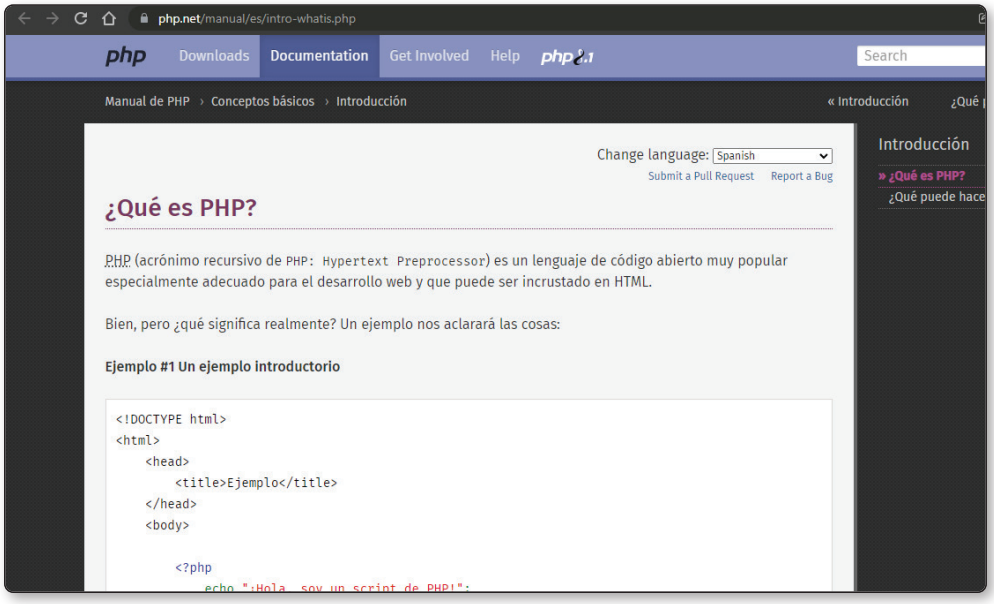


Figura 2.1. PHP es un lenguaje de código abierto, con una amplia comunidad de desarrollo.

Una vez que domines las características básicas del lenguaje, será una gran idea que aproveches la orientación a objetos que este posee. El paradigma de la programación orientada a objetos es una característica de los lenguajes modernos, como **Java**, **C#** o **JavaScript**, que permite desarrollar de manera escalable, creando código limpio, fácil de leer e interpretar, y de una forma segura. PHP posee una

fuerte orientación a este paradigma, y puedes aprender sobre clases, objetos, métodos, herencia y otras particularidades interesantes en la colección Programación Orientada a Objetos en PHP, donde verás a fondo las características más relevantes de este lenguaje y del paradigma. Accede a la colección en este *enlace*.

Luego de aprender sobre estos temas, puedes continuar conociendo las herramientas más utilizadas del lenguaje: las librerías de bases de datos avanzadas, como PDO, el uso de servidores de correo electrónico, las librerías creadas para PHP, los gestores de paquetes de software, el tratamiento de pagos y dinero online, y otras características muy demandadas dentro de la industria del software, que se tratan en esta colección.

Esta obra se centrará en aprovechar todos tus conocimientos sobre PHP, y en explotarlos al máximo para que puedas crear sistemas web avanzados, con características como las mencionadas anteriormente. De este modo, estarás preparado para afrontar cualquier desafío laboral que implique el manejo de este lenguaje.

En este caso, para comprender cómo se trabaja con cada una de estas herramientas, utilizarás un sistema base que podrás manejar a lo largo de los próximos volúmenes de esta obra. En esta primera entrega, crearás este sistema que se encargará de generar una conexión con la base de datos y el sistema de login, para que los usuarios puedan ingresar e iniciar sesión. Para hacerlo, necesitarás instalar en tu computadora una serie de tecnologías y herramientas destinadas a desarrollar en PHP con bases de datos. En primer lugar, tendrás que instalar el lenguaje de programación PHP, junto con el servidor Apache. Luego, necesitarás instalar el motor de bases de datos MySQL, para poder ejecutar tu aplicación. Para esto, si te encuentras en Windows, la forma más sencilla de trabajar con estas tecnologías es mediante un paquete de desarrollo como XAMPP o WAMP, los cuales te permiten adquirirlas en una sola instalación.

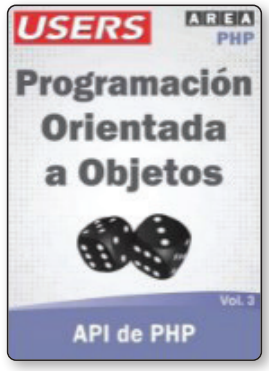
Figura 2.2. El paquete de instalación XAMPP ofrece PHP, MySQL, Perl y Apache Server.

Una vez que hayas descargado e instalado el paquete en tu equipo, es recomendable definir en el **PATH** una variable de entorno para PHP, que te permitirá utilizar Composer y PHP en la terminal de tu sistema operativo. También podrás instalar de forma individual cada tecnología, pero esta es una tarea más compleja.

Si trabajas con un sistema operativo Mac, puedes recurrir a un paquete como MAMP, que instala las mismas tecnologías pero en los sistemas de Apple.

Figura 2.3. MAMP es un paquete de software para desarrollar con PHP y MySQL en MacOS.

En sistemas Linux, tendrás que actualizar la lista de **repositorios disponibles** para tu sistema, y luego instalar el lenguaje mediante el comando **sudo apt install php8.0 libapache2-mod-php8.0**. Además, deberás instalar Apache2 y MySQL, y reiniciar el servidor con el comando correspondiente. También puedes instalar el paquete de desarrollo LAMP en tu computadora, para instalar el lenguaje PHP junto con el motor de bases de datos y el servidor Apache, todo en una misma instalación. Si deseas conocer más acerca de la manera de instalar estas tecnologías, puedes leer la primera entrega de la colección Programación Orientada a Objetos en PHP, en el siguiente *enlace*.



Una vez que tengas todo instalado en tu sistema operativo, deberás instalar Composer, el gestor de paquetes de PHP, que te permite trabajar con el lenguaje, instalar librerías o incluso frameworks como Laravel o **Symphony**, de manera rápida y cómoda.

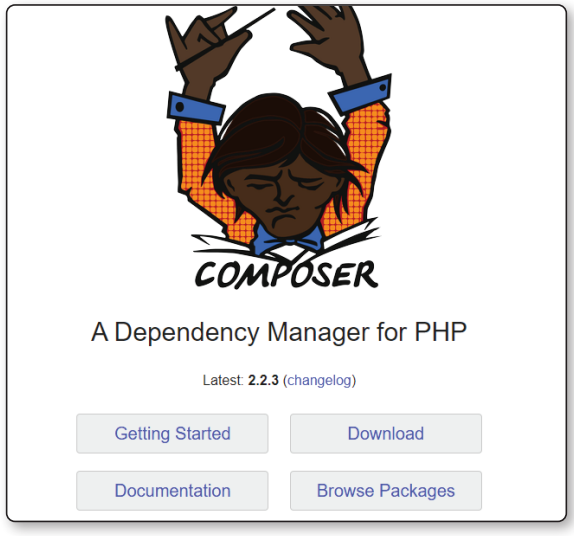


Figura 2.4. Composer es el gestor de paquetes de PHP por excelencia.

Para instalar Composer, debes dirigirte en tu navegador a *https://getcomposer.org*, donde podrás acceder a un instalador o a las instrucciones necesarias para instalar este software. Una vez que tengas el lenguaje de programación junto con el gestor de paquetes, para verificar que todo esté como corresponde, abre una terminal y ejecuta los comandos **php —version** y **composer –version**. Estos deberían indicarte qué versión de cada tecnología tienes instalada.

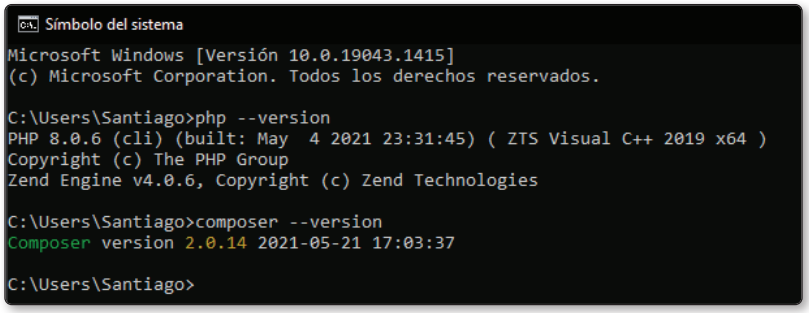


Figura 2.5. Ambos comandos deberían devolver las versiones instaladas de cada programa.

A continuación, podrás comenzar a desarrollar y a aprender las características más avanzadas del lenguaje. Para finalizar, tendrás que utilizar un editor de código destinado a desarrollar en PHP. Una alternativa ligera e interesante es Visual Studio Code, que permite desarrollar en cualquier lenguaje, e instalar plugins para ayudarte a programar con sugerencias y corrección de errores en la sintaxis.

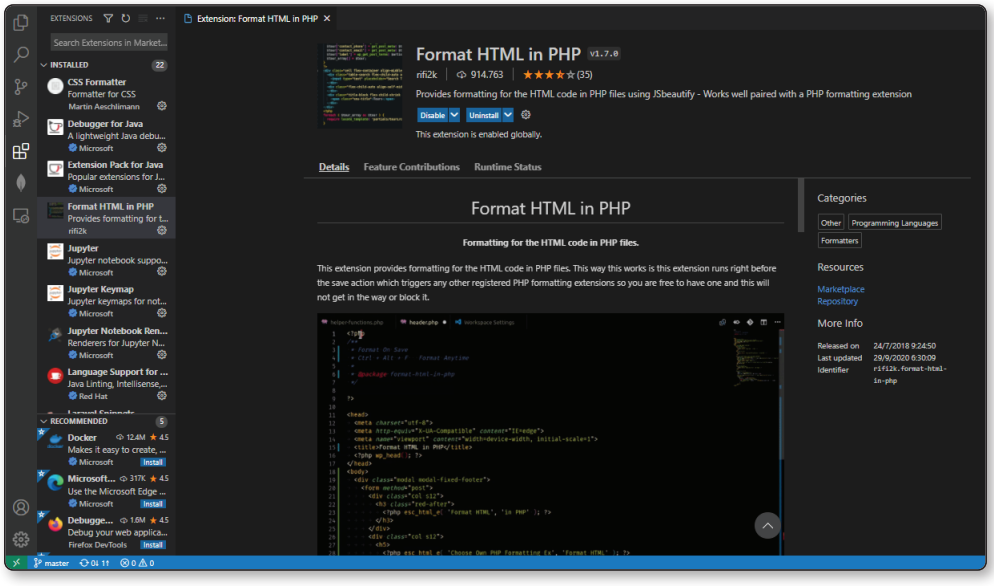


Figura 2.6. Visual Studio Code permite instalar plugins para desarrollar en PHP o en cualquier otro lenguaje.

Dentro de los entornos de desarrollo integrados, se encuentra Eclipse, un software de código abierto que posee distintas herramientas muy útiles, como aquellas destinadas a ejecución de software, integración con **PHPUnit**, trabajo con Git y repositorios remotos, entre otras.

Para finalizar, otra alternativa popular, aunque paga, es PHPStorm, un IDE de la empresa Jetbrains que ofrece una gran cantidad de herramientas, como control de código, conexión a servidores remotos y trabajo con bases de datos. Si tienes una cuenta de correo universitaria como alumno o profesor, puedes obtener acceso gratuito a la versión community del software.

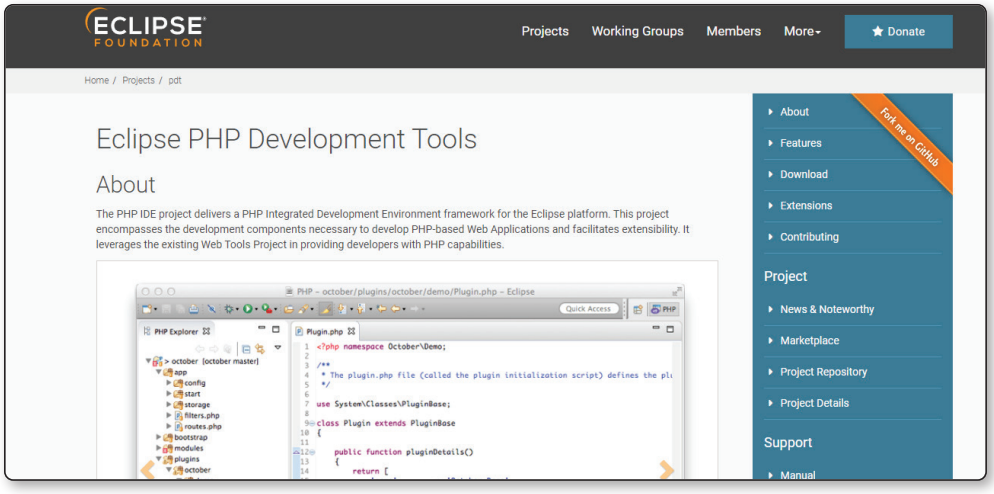


Figura 2.7. Eclipse permite desarrollar en PHP de manera cómoda.

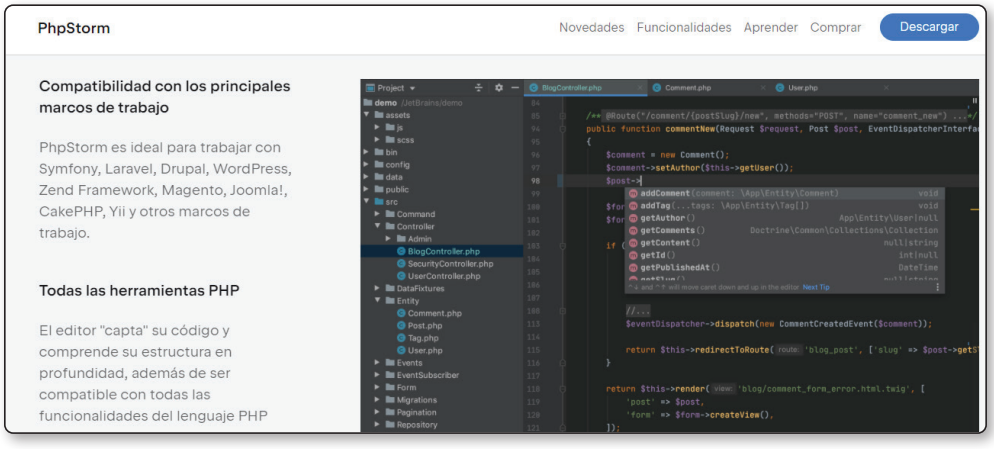


Figura 2.8. PHPStorm es uno de los IDEs más utilizados y completos dentro de la comunidad.

La elección del IDE o editor de código es totalmente personal, y depende de los gustos y las necesidades de cada desarrollador.

2.2 CREACIÓN DEL PROYECTO EN PHP

Ahora que cuentas con las herramientas necesarias para trabajar, es momento de empezar a crear un proyecto de PHP. Si trabajas con XAMPP, como en este ejemplo, sería recomendable generar una nueva carpeta dentro del paquete en Apache, en la carpeta **htdocs**, y abrirla con tu editor de código. En caso de que uses otro entorno, puedes hacerlo en la ubicación que más te convenga. Para comenzar a trabajar, tendrás que dividir tu proyecto en algunas carpetas e inicializarlo con Composer, para instalar las dependencias necesarias y declararlas en el archivo **Composer.json**.

Dentro de la terminal del sistema operativo, o en la terminal integrada del editor de código, ubicado dentro de la carpeta del proyecto, ejecuta el comando **composer require phpmailer/phpmailer**. Este se encargará de instalar en tu proyecto la librería **PHPMailer**, una de las más clásicas y utilizadas en el desarrollo en PHP, que permite crear un servidor simple de correo SMTP y enviar correos electrónicos con este lenguaje de programación, utilizando plantillas de correo en HTML, una cuenta de e-mail preparada para el envío, uso de nombre de usuario, y varias características interesantes.

Una vez que hayas instalado esta librería, verás en la carpeta de tu proyecto un archivo **composer.json**, en cuyo interior hay un código como el siguiente:

```
{
   “name”: “vendor_name/proyecto-php”,
   “description”: “description”,
   “minimum-stability”: “stable”,
   “license”: “GPLv3”,
   “authors”: [
      {
      “name”: “Santiago”,
      “email”: “aguirresantiago@gmail.com”
      }
   ],
   “require”: {
      “phpmailer/phpmailer”: “^6.5”,
      “ext-pdo”: “*”
   }
}
```

Como puedes observar en este archivo, Composer genera un código en formato JSON que permite especificar una serie de parámetros para todo aquel que desee instalar en su computadora tu proyecto, o para ti mismo si deseas subir el código de tu proyecto a GitHub o GitLab, y reinstalar las dependencias.

Dentro de este archivo, se coloca un nombre para el proyecto, una descripción, una versión mínima estable, una licencia y el nombre del autor, junto con su correo electrónico, para propósitos de contacto o soporte, casos de dudas en su uso, contacto para señalar bugs, errores o mejoras, o cuestiones similares.

Además, se establece una sección llamada Requiere, con un objeto JSON en su interior, donde ya se listan algunas cuestiones.

En tu caso, probablemente solo exista la librería **PHPMailer** con la versión instalada. Esto significa que has instalado correctamente el paquete para el envío de correos electrónicos, y que cualquier persona que utilice el proyecto en su equipo necesitará ejecutar el comando **composer install** para instalar esta dependencia y que funcione en su entorno local. Además, indica la versión mínima de la librería para trabajar en el proyecto; en este caso, se requiere al menos la versión 6.5. En tu ejemplo, la versión puede diferir, dependiendo del momento en que la hayas instalado.

Luego de instalar la librería, es momento de comenzar a trabajar con tus modelos de esta aplicación.

En primer lugar, será necesario que el proyecto tenga algún tipo de sistema de login, por lo cual precisarás un modelo de usuario que trabaje para registrar en la base de datos tus usuarios, les permita iniciar sesión chequeando sus credenciales, e inicie sesiones en el servidor cuando un usuario ingrese.

Para lograrlo, lo primero será crear una carpeta llamada **models**, y en su interior, crear una clase **Model**, de la cual heredarán las demás clases modelo:

```
<?php

namespace models;

use Controller\Connection;

abstract class Model
{
   public abstract function save(
      Connection $connection);
}
```

La clase abstracta **Model**, que no deberá ser instanciada, tendría que verse de esta manera. Debería poseer un método llamado **save()**, que te permitirá guardar métodos dentro de la base de datos (**Figura 2.9.**)

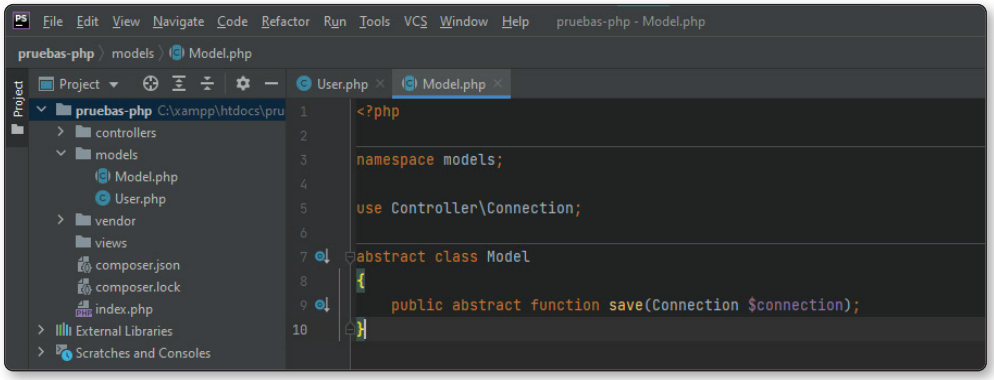


Figura 2.9. PHPStorm permite crear clases con su asistente, y especificar la herencia, un namespace y otros datos.

Este método será sobrescrito por cada modelo que herede de la clase **Model**, utilizando el código que requiera en cada caso. Además, la función necesitará como parámetro un objeto de la clase **Connection**, que aún no has creado, dentro de la carpeta **controllers**. Ahora, crea el directorio **controllers**, dentro de la raíz de tu proyecto, y en su interior, genera una nueva clase llamada **Connection**, la cual debería lucir de este modo:

```
<?php

namespace Controller;

use PDO;
use PDOException;

class Connection
{
   protected $connection;
   protected $servername;
   protected $username;
   protected $password;

   public function __construct()
   {
   }

   public function get_connection()
   {
   }
```

```
        public function close_connection()
        {
        }
    }
```

Como puedes ver, la clase **Connection** tendrá cuatro campos de clase, cada uno **protected**, y tres métodos que funcionarán para abrir una conexión a la base de datos, dentro del constructor, un método para obtener esa conexión, y otro para cerrarla.

Habrás notado que, dentro del archivo **composer.json**, se especifica la extensión PDO, la librería de conexiones a la base de datos que te permite trabajar con MySQL, **SQLite**, **PostgreSQL**, o casi cualquier otra del mercado. Esta será la librería que utilizarás para trabajar con la base de datos y MySQL, en este proyecto. PDO u Objetos de Datos en PHP, como lo define la documentación oficial, permite abstraer las características de la base de datos y trabajar de forma más segura con las conexiones, creando consultas preparadas. Al trabajar con una capa de abstracción del acceso a los datos, PDO se utiliza para cualquier base de datos, empleando las mismas funciones. Por lo tanto, si necesitas comenzar a trabajar con una base de datos distinta, es decir, migrar de MySQL a **MS SQL Server**, SQLite, u otra, las funciones serán las mismas, sin necesidad de cambiar el código general de la aplicación.

En la mayoría de las instalaciones de PHP, la librería PDO viene instalada por defecto y activada, pero para evitar errores a la hora de reinstalar el proyecto, en el archivo **composer.json** se especifica que se utilizará la extensión PDO. Si tienes que activar la extensión de PDO, puedes hacerlo desde tu archivo **php.ini**, dentro de la instalación del lenguaje en el sistema operativo.

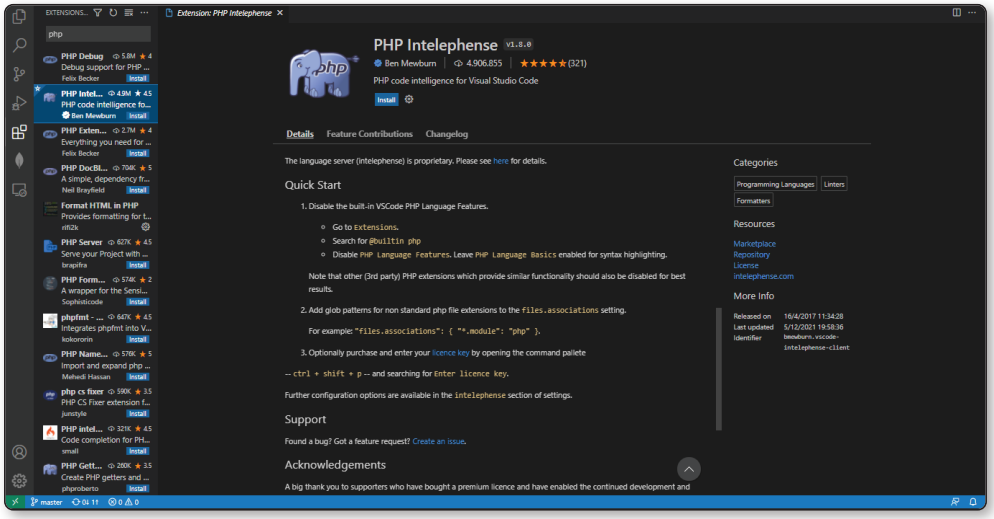


Figura 2.10. Si trabajas con VS Code, puedes instalar extensiones para formatear el código, PHP o HTML dentro de archivos de este lenguaje.

Dentro de la función **__construct()**, de la clase **Connection**, coloca el siguiente código:

```
public function __construct()
{
   try {
      $this->username = “root”;
      $this->password = “”;
      $this->servername = “localhost”;

      $this->connection = new
   PDO(“mysql:host=$this->servername;dbname=test”,    $this->username, $this-
>password);
   $this->connection
      ->setAttribute(PDO::ATTR_ERRMODE,
         PDO::ERRMODE_EXCEPTION);
   } catch(PDOException $e) {
      echo “Connection failed: “ . $e
         ->getMessage();
   }
}
```

Una vez que hayas incluido este código, al generar un nuevo objeto de la clase **Connection**, se creará una conexión a la base de datos, con los campos de clase ya definidos.

El siguiente paso será definir el código de los métodos que retornan la conexión y la cierran, de esta forma:

```
public function get_connection()
{
   return $this->connection;
}

public function close_connection()
{

   $this->connection = null;
}
```

Ahora, la clase ya está lista, y puedes comenzar a utilizarla dentro de otras clases.

A continuación, tendrás que definir una clase llamada **User**, que represente los registros de los usuarios dentro de la base de datos, la cual deberá heredar de la clase **Model**. Aquí, deberás definir cada uno de los campos que tendrá un usuario, que luego se guardarán en la base de datos:

```
<?php

namespace models;

class User extends Model
{
   protected $name;
   protected $mail;
   protected $password;
   protected $sector;
   protected $token;
   protected $mail_verified;

   public function __construct()
   {
   }

   public function save(\Controller\Connection
      $connection)
   {
   }
}
```

En este caso, los usuarios tendrán las propiedades **name**, **mail** y **password**; estas dos últimas se utilizarán para el inicio de sesión. Luego, tendrán un sector o departamento, un **token** que servirá para el reseteo de contraseñas y un campo para aquellos que verifiquen su cuenta.

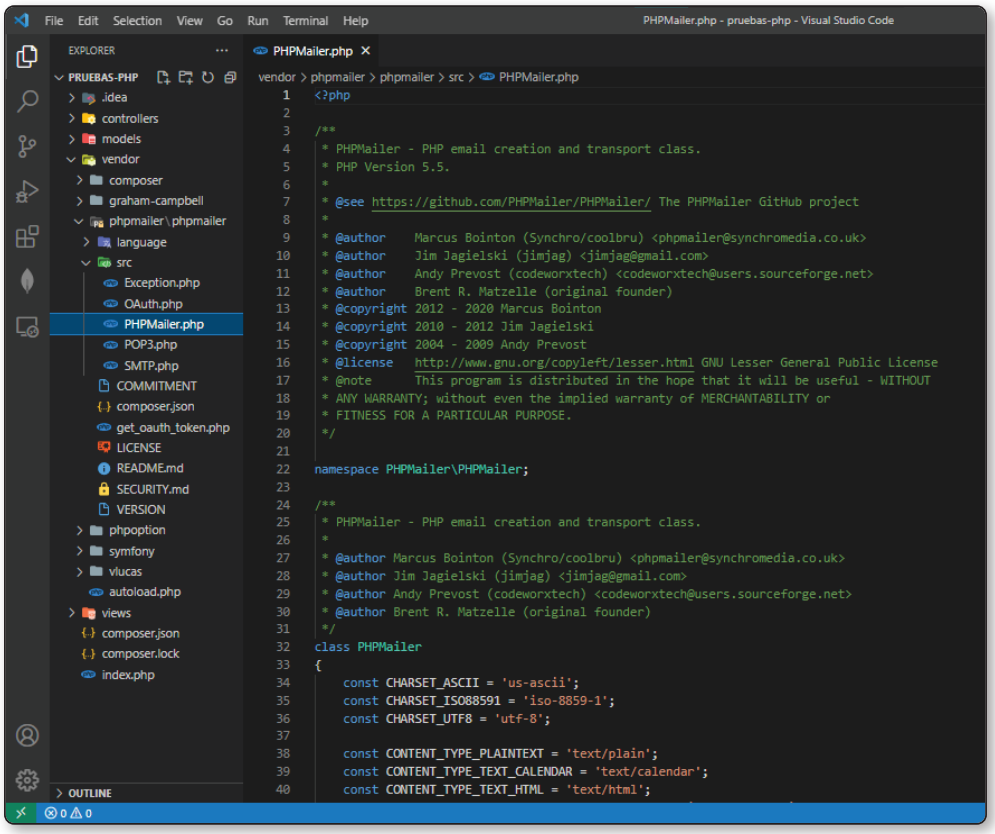


Figura 2.11. La carpeta vendor no debería commitearse, y contiene el código de las librerías instaladas con Composer.

Ahora, dentro de la función **__construct()**, tendrás que definir los valores de los campos de clase, de esta manera:

```
public function __construct($name, $mail, $sector,    $password)
{
   $this->name = $name;
   $this->mail = $mail;
   $this->password = password_hash($password,
      PASSWORD_DEFAULT);
   $this->sector = $sector;
   $this->token = null;
   $this->mail_verified = null;
}
```

Luego, debes sobrescribir el método **save()**, que hereda de la clase **Model**, para guardar dentro de la base de datos cada registro:

```
public function save(\Controller\Connection $connection)
{
   $con = $connection->get_connection();

   $stmt = $con->prepare("INSERT INTO users (name, mail, password, sector,
token, creation, mail_verified) VALUES (:name, :mail, :password, :sector, :token,
:mail_verified)");

   $stmt->bindParam(":name", $this->name);
   $stmt->bindParam(":mail", $this->mail);
   $stmt->bindParam(":password", $this->password);
   $stmt->bindParam(":sector", $this->sector);
   $stmt->bindParam(":token", $this->token);
   $stmt->bindParam(":mail_verified", $this
      ->mail_verified);

   $stmt->execute();
}
```

Por último, necesitarás un método para chequear el login de cada usuario, es decir, para verificar si el usuario ingresado y la contraseña son correctas:

```
public static function login(Connection $connection,$email, $password)
{
   $con = $connection->get_connection();

   $stmt = $con->prepare("SELECT mail, password FROM users WHERE mail= ?");
   $stmt->execute(array($email));
   $user = $stmt->fetch();

   if($user && password_verify($password,
      $user['password']))
   {
      return true;
   }
   return false;
}
```

Ahora, cuando se llame al método **login()**, se verificará si el usuario o mail y la contraseña son los que corresponden. Para continuar y probar este código, tendrás que crear la base de datos para el proyecto y una tabla. La base deberá llevar como nombre el mismo que uses en la cadena de conexión, dentro del método constructor de la clase **Connection**. Luego, crearás una tabla dentro de tu gestor de bases de datos MySQL, llamada **users**, con la siguiente sentencia SQL:

```
CREATE TABLE `users` (
   `id` int(11) PRIMARY KEY AUTO_INCREMENT NOT
      NULL,
```

```
    `name` varchar(200) DEFAULT NULL,
    `mail` varchar(200) UNIQUE DEFAULT NULL,
    `sector` varchar(200) DEFAULT NULL,
    `password` varchar(255) DEFAULT NULL,
    `token` varchar(255) DEFAULT NULL,
    `mail_verified` tinyint(1) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

Ya puedes probar el código que has generado. Dirígete a la raíz del proyecto y, allí, registra en primer lugar un usuario, para luego verificar con un inicio de sesión. Crea un archivo llamado **index.php** dentro de la raíz del proyecto, y coloca el siguiente código en su interior:

```
<?php

require_once ‘./controllers/Connection.php’;
require_once ‘./models/User.php’;
require_once ‘./models/Model.php’;
use Controller\Connection;
use Models\User;

$connection = new Connection();

$user = new User(“Santiago”, “santi@mail.com”, “IT Department”, “password”);
$user->save($connection);

echo User::login($connection, “santi@mail.com”, “password”);
```

Ahora, en caso de que exista un error, en el navegador se indicará el problema generado. De lo contrario, en la pantalla se mostrará un 1, indicando que la función **login()** dio como resultado **true** al intentar iniciar la sesión.

Como ya has probado que los métodos de las clases funcionan correctamente, puedes crear un formulario para generar estos datos en forma dinámica, y métodos para validar los datos y luego escapar algunos de los caracteres que se ingresan en el formulario, de modo de tener mayor seguridad en el sistema. Dentro de la clase **User**, crea la siguiente función:

```
public static function validate($name, $mail, $password, $sector)
{
   if($name == null || $mail == null
   || $password == null || $sector == null)
   {
      return false;
   }
   return true;
}
```

Esta función se encargará de verificar que ningún dato sea nulo o esté vacío al enviar el formulario. También puedes agregar otras reglas en este método, como solicitar un mínimo de caracteres para las contraseñas y utilizar expresiones regulares para verificar que el campo mail corresponda a un correo electrónico válido.

Luego, puedes crear un método en la clase **Model**, que será utilizado por las demás clases, para escapar los datos ingresados en el formulario, evitando así caracteres extraños que pueden llevar a vulnerabilidades de seguridad, como **inyección SQL** o ataques **XSS**.

Dentro de la clase **Model**, crea el siguiente método estático:

```
public static function escapeData($input)
{
   $input = trim($input);
   $input = htmlspecialchars($input);
   return stripslashes($input);
}
```

Ahora, en la carpeta **views** puedes crear un archivo con un formulario, para que los usuarios se registren de forma dinámica, como el siguiente:

```
<!doctype html>
<html lang=”es”>
<head>
   <meta charset=”UTF-8”>
   <meta name=”viewport”
   content=”width=device-width, user-scalable=no, initial-scale=1.0, maximum-
scale=1.0, minimum-scale=1.0”>
   <meta http-equiv=”X-UA-Compatible”  content=”ie=edge”>
   <title>Document</title>
</head>
<body>

<form action=”register_end.php” method=”post”>
   <label for=”name”>Ingrese nombre</label>
   <input type=”text” name=”name” class=”” />
   <br>

   <label for=”name”>Ingrese correo</label>
   <input type=”email” name=”email” class=”” />
   <br>

   <label for=”name”>Ingrese una contraseña</label>
   <input type=”password” name=”password” class=”” />
   <br>
```

```
    <select name="sector" id="">
    <option value="IT Department">IT
       Department</option>
    <option value="Sales">Sales</option>
    <option value="Graphic Design">Graphic
       Design</option>
    <option value="Marketing">Marketing</option>
    </select>

    <button type="submit">Registrarse</button>
</form>

</body>
</html>
```

En los siguientes capítulos, verás cómo dar estilos a este formulario y al resto de la aplicación. Ahora, crea un archivo llamado **register_end.php**, y allí coloca el siguiente código:

```
<?php
require_once '../controllers/Connection.php';
require_once '../models/User.php';
use Controller\Connection;

use Models\User;

$connection = new Connection();

$name = User::escapeData($_POST['name']);
$email = User::escapeData($_POST['email']);
$password = $_POST['password'];
$sector = User::escapeData($_POST['sector']);

if(!User::validate($name, $email, $sector, $password))
{
   echo "Error de validación, datos faltantes";
}
else
{
   $user = new User($name, $email, $sector,
      $password);
   if($user->save($connection))
   {
      echo "Registro correcto";
   }
}
```

Si pruebas a registrar un usuario, verás que en el navegador se te indica que lo has hecho de manera correcta en la base de datos, con lo cual ya puedes probar a iniciar sesión. Crea un archivo llamado **login.php** dentro de la carpeta **views**, y allí coloca otro formulario, como el siguiente:

```
<form action=”inicio.php” method=”post”>
   <label for=”name”>Ingrese su email</label>
   <input type=”email” name=”email” class=”” />
   <br>

   <label for=”name”>Ingrese una contraseña</label>
   <input type=”password” name=”password” class=”” />
   <br>

   <button type=”submit”>Iniciar sesión</button>
</form>
```

Luego, crea un archivo llamado **inicio.php** con el siguiente código:

```
<?php
require_once ‘../controllers/Connection.php’;
require_once ‘../models/User.php’;
use Controller\Connection;

use Models\User;

$connection = new Connection();

$email = User::escapeData($_POST[‘email’]);
$password = $_POST[‘password’];

if(User::login($connection, $email, $password))
{
   echo “Inicio de sesión correcto”;
}
else
{
   echo “Error en el inicio de sesión”;
}
```

En el próximo capítulo, verás cómo dar más estilos a tus formularios, mejorar el código haciendo un **refactor** y generar una sesión en el servidor para que tus usuarios puedan navegar por la aplicación.

2.3 ACTIVIDADES

A continuación se presentan las preguntas y los ejercicios que deberías saber responder y resolver, para considerar aprendido el capítulo.

2.3.1 Test de autoevaluación

1. ¿Qué es PDO?

2. ¿Para qué se utiliza Composer?

3. ¿Para qué se usa el archivo **composer.json***?*

4. ¿Qué es PHPMailer?

2.3.2 Ejercicios prácticos

1. Dentro de la clase **User***, crea un nuevo método llamado* **userDetail***.*

2. Este debe tomar como parámetro un objeto de tipo **Connection***, y un email de tipo* **string***.*

3. Llama a la conexión de la función, y retorna los datos del usuario consultado a la base de datos con el mail.