**PROYECTO S.I.G.E.N 2024**

****

**PRIMERA ENTREGA**

**Solicitante:** Instituto tecnológico de Informática

**Grupo:** 3°BB

**Horario:** Matutino

**Materia:** Programación Web

**Nombre del docente**: Marcos Lucia

**Miembros del equipo:**

Ismael Bergara, Facundo Recagno, Thiago calfani y Diego García

**Fecha de culminación:** 15/07/2024

Indice

[**Introducción**](#_heading=h.30j0zll) **3**

[**Objetivos**](#_heading=h.1fob9te) **3**

[Objetivo General](#_heading=h.3znysh7) 3

[Objetivos Específicos](#_heading=h.2et92p0) 3

[**Alcance**](#_heading=h.tyjcwt) **4**

[**Patrón de arquitectura del proyecto**](#_heading=h.3dy6vkm) **4**

[Modelo vista controlador(MVC)](#_heading=h.1t3h5sf) 4

[Separación de Preocupaciones](#_heading=h.4d34og8) 4

[Facilita el Desarrollo Colaborativo](#_heading=h.2s8eyo1) 5

[Mejor Mantenimiento y Escalabilidad](#_heading=h.17dp8vu) 5

[Reutilización de Código](#_heading=h.3rdcrjn) 5

[Facilita las Pruebas](#_heading=h.26in1rg) 5

[Mejora la Estructura del Código](#_heading=h.lnxbz9) 5

[Conclusión](#_heading=h.35nkun2) 5

[**Principales clases del programa**](#_heading=h.1ksv4uv) **6**

[Descripción de la Clase Database](#_heading=h.44sinio) 6

[Descripción de la Clase UsuarioRepository](#_heading=h.2jxsxqh) 6

[Razones para Utilizar la Clase UsuarioRepository](#_heading=h.z337ya) 7

[Descripción de la Clase UsuarioService](#_heading=h.3j2qqm3) 8

[Razones para Utilizar la Clase UsuarioService](#_heading=h.1y810tw) 9

[Descripción de la Clase UsuarioModel](#_heading=h.4i7ojhp) 9

[Razones para Utilizar la Clase UsuarioModel](#_heading=h.2xcytpi) 10

[Descripción de la Clase UsuarioController](#_heading=h.1ci93xb) 11

[**Razones para Utilizar MySQLi**](#_heading=h.3whwml4) **12**

[**Importancia de Usar GitHub para el Desarrollo en Equipo**](#_heading=h.2bn6wsx) **13**

[Repositorio del Proyecto](#_heading=h.qsh70q) 14

[**Instructivo de Ejecución**](#_heading=h.3as4poj) **14**

[Prerrequisitos](#_heading=h.1pxezwc) 14

[Paso 1: Instalación de Composer](#_heading=h.49x2ik5) 14

[Paso 2: Configurar el VirtualHost en Apache](#_heading=h.2p2csry) 14

[Paso 4: Reiniciar Apache](#_heading=h.147n2zr) 15

[Paso 5: Uso de Composer](#_heading=h.3o7alnk) 15

[Nota Importante sobre la Configuración del VirtualHost](#_heading=h.23ckvvd) 15

[Detalles](#_heading=h.ihv636) 15

[Implicaciones](#_heading=h.32hioqz) 15

[**Bibliografía**](#_heading=h.2naadhqcjj6j) **16**

# 

# 

# Introducción

El presente documento detalla la arquitectura, diseño e implementación del Sistema Informático de Gestión de Entrenamiento (SIGEN). Este sistema tiene como objetivo principal optimizar la gestión de planes de entrenamiento, fisioterapia y administración en centros deportivos y de rehabilitación.

El desarrollo de SIGEN se basa en un enfoque modular y escalable, utilizando tecnologías web modernas y una arquitectura Modelo-Vista-Controlador (MVC) para garantizar un sistema eficiente, seguro y fácil de mantener. A través de una implementación cuidadosa y pruebas rigurosas, se busca entregar un producto de alta calidad que cumpla con los requisitos y expectativas de los usuarios.

# Objetivos

## Objetivo General

Desarrollar un sistema informático de gestión de entrenamiento (SIGEN) que permita gestionar, organizar y monitorear el desempeño y recuperación de deportistas, así como medir su evolución a lo largo del tiempo.

## Objetivos Específicos

Implementar una arquitectura MVC eficiente y escalable: Utilizar el patrón Modelo-Vista-Controlador para separar las responsabilidades del sistema y facilitar el desarrollo, mantenimiento y evolución del código.

Desarrollar un sistema seguro y confiable: Implementar medidas de seguridad para proteger los datos de los usuarios y garantizar la disponibilidad y confiabilidad del sistema.

Crear una interfaz de usuario intuitiva y fácil de usar: Diseñar una interfaz que permita a los usuarios interactuar con el sistema de manera eficiente y sin dificultades.

Integrar el sistema con una base de datos: Utilizar una base de datos para almacenar y gestionar los datos de usuarios, planes de entrenamiento, ejercicios y registros de evolución.

Realizar pruebas exhaustivas: Someter el sistema a pruebas rigurosas para identificar y corregir errores, asegurando un producto final de alta calidad.

# Alcance

El alcance del desarrollo del sistema SIGEN abarca las siguientes actividades:

Implementación del modelo de datos: Diseño y creación de las tablas y relaciones en la base de datos para almacenar la información del sistema.

Desarrollo de la lógica de negocio: Implementación de las clases y funciones necesarias para procesar los datos y realizar las operaciones del sistema.

Creación de la interfaz de usuario: Diseño y desarrollo de las páginas web y componentes visuales para la interacción con el usuario.

Integración con la base de datos: Conexión del sistema con la base de datos para almacenar y recuperar información.

Pruebas y depuración: Realización de pruebas unitarias, de integración y de sistema para asegurar el correcto funcionamiento del sistema.

# Patrón de arquitectura del proyecto

## Modelo vista controlador(MVC)

El Modelo-Vista-Controlador (MVC) es un patrón de arquitectura de software ampliamente utilizado en el desarrollo de aplicaciones, especialmente en el desarrollo web.

## Separación de Preocupaciones

El MVC divide una aplicación en tres componentes principales: Modelo, Vista y Controlador. Cada componente se encarga de aspectos específicos de la funcionalidad:

Modelo: Maneja la lógica de datos y negocio. Se ocupa de acceder, almacenar y procesar los datos.

Vista: Responsable de la presentación de los datos. Define cómo se muestran los datos al usuario.

Controlador: Actúa como un intermediario que maneja la entrada del usuario, interactúa con el modelo y selecciona la vista apropiada para mostrar la salida.

Esta separación de responsabilidades facilita el mantenimiento y la escalabilidad de la aplicación, ya que cada componente puede desarrollarse, probarse y modificarse de manera independiente.

## Facilita el Desarrollo Colaborativo

La división en componentes claros permite que diferentes equipos o desarrolladores trabajen en paralelo sin interferencias. Por ejemplo, los diseñadores pueden trabajar en la vista mientras los desarrolladores de backend se centran en la lógica del modelo.

## Mejor Mantenimiento y Escalabilidad

La estructura modular de MVC facilita la localización de problemas y la implementación de mejoras. Si un cambio es necesario en la lógica de negocio, se puede modificar el modelo sin afectar las vistas y controladores, y viceversa. Esta modularidad también hace que la aplicación sea más fácil de escalar.

## Reutilización de Código

El MVC promueve la reutilización de código. Por ejemplo, el modelo y el controlador pueden ser reutilizados con diferentes vistas para distintas interfaces de usuario (web, móvil, escritorio), lo cual ahorra tiempo y esfuerzo en el desarrollo de aplicaciones multiplataforma.

## Facilita las Pruebas

El patrón MVC permite pruebas unitarias más efectivas y sencillas. Cada componente se puede probar de manera aislada. Los modelos se pueden probar sin necesidad de cargar vistas o manejar eventos de entrada, y los controladores pueden ser probados para verificar la lógica de flujo de la aplicación.

## Mejora la Estructura del Código

El uso de MVC proporciona una estructura clara y organizada al código de la aplicación, lo que facilita su comprensión y gestión, especialmente en proyectos grandes y complejos. Esta claridad también ayuda a nuevos desarrolladores a integrarse en el proyecto más rápidamente.

## Conclusión

El uso del patrón MVC en el desarrollo de software no solo mejora la organización y eficiencia del desarrollo, sino que también proporciona una estructura robusta y flexible que facilita el mantenimiento, escalabilidad y pruebas de la aplicación. Su adopción en proyectos de software contribuye significativamente a la calidad y sostenibilidad del producto final.

# Principales clases del programa

## Descripción de la Clase Database

La clase Database se encarga de gestionar la conexión a una base de datos MySQL y está diseñada para ser flexible y segura. A continuación, se describen sus principales características y propiedades:

$servername: El nombre del servidor donde se encuentra la base de datos.

$username: El nombre de usuario para acceder a la base de datos.

$password: La contraseña para acceder a la base de datos.

$dbname: El nombre de la base de datos a la que se va a acceder.

$conn: Un objeto de conexión a la base de datos, inicialmente nulo.

El método \_\_construct() actúa como el constructor de la clase y se invoca automáticamente cuando se crea un objeto de la clase. Este constructor toma cuatro argumentos opcionales, que se utilizan para establecer las propiedades necesarias para la conexión a la base de datos.

El método connect() es responsable de establecer la conexión utilizando las propiedades definidas anteriormente. Este método garantiza que la conexión se gestione de manera eficiente y segura.

## Descripción de la Clase UsuarioRepository

La clase UsuarioRepository se encuentra en el archivo UsuarioRepository.php y su principal responsabilidad es interactuar con la base de datos para realizar operaciones relacionadas con los usuarios. A continuación, se describen sus componentes y funcionalidades clave:

use App\Models\UsuarioModel;

Esto importa la clase UsuarioModel desde el directorio App\Models, la cual define la estructura de datos de un usuario.

public function \_\_construct() {

$this->database = new Database();

}

Este es el constructor de la clase. Al crear una nueva instancia de UsuarioRepository, se crea una nueva instancia de la clase Database y se asigna a la variable $database.

public function guardar(UsuarioModel $usuarioModel)

Esta es una función pública llamada guardar que toma un objeto UsuarioModel como argumento. Su propósito es guardar un usuario en la base de datos.

$sql = "INSERT INTO Usuario (nroDocumento, rol, passwd) VALUES (?,?, ?)";

Esta es una consulta SQL que inserta un nuevo usuario en la tabla Usuario.

$nroDocumento = $usuarioModel->getNroDocumento();

$rol = $usuarioModel->getRol();

$passwd = $usuarioModel->getPasswd();

Estas líneas obtienen los datos del usuario desde el objeto UsuarioModel.

$stmt = $this->database->getConnection()->prepare($sql);

Esto prepara la consulta SQL para su ejecución.

$stmt->bind\_param("iis", $nroDocumento, $rol, $passwd);

Esto vincula los parámetros a la consulta SQL preparada. Los parámetros son los datos del usuario que se obtuvieron del objeto UsuarioModel.

## Razones para Utilizar la Clase UsuarioRepository

El uso de la clase UsuarioRepository proporciona varios beneficios clave para la gestión de datos de usuarios en la aplicación:

**Encapsulación de la Lógica de Acceso a Datos:** La clase UsuarioRepository encapsula toda la lógica de acceso a datos en un solo lugar, lo que mejora la organización y mantenibilidad del código. Esto permite que otros componentes de la aplicación interactúen con la base de datos a través de una interfaz clara y consistente.

**Reutilización de Código:** Al centralizar las operaciones de base de datos con la tabla usuario en la clase UsuarioRepository, se evita la duplicación de código y se facilita la reutilización de estas operaciones en diferentes partes de la aplicación.

**Seguridad y Prevención de Inyecciones SQL:** El uso de consultas preparadas y la vinculación de parámetros en MySQLi ayudan a prevenir ataques de inyección SQL, mejorando así la seguridad de la aplicación.

**Facilidad de Mantenimiento y Evolución:** Cualquier cambio en la lógica de acceso a datos, como modificaciones en la estructura de la base de datos o mejoras en la consulta, se puede realizar en un solo lugar. Esto simplifica el mantenimiento y evolución del código.

**Separación de Preocupaciones:** Separar la lógica de acceso a datos de la lógica de negocio y de presentación facilita el desarrollo colaborativo y mejora la claridad del código. Los desarrolladores pueden trabajar en distintas capas de la aplicación de manera más eficiente.

## Descripción de la Clase UsuarioService

La clase UsuarioService encapsula la lógica de negocio relacionada con los usuarios y utiliza una instancia de UsuarioRepository para interactuar con la base de datos. A continuación, se describen sus componentes y funcionalidades clave:

public function crearUsuario(UsuarioModel $usuarioModel) {

$this->usuarioRepository->guardar($usuarioModel);

}

Este es un método público llamado crearUsuario que toma una instancia de UsuarioModel como argumento. Luego, llama al método guardar en $usuarioRepository, pasando el UsuarioModel. Este método crea un nuevo usuario en la base de datos.

public function autenticar($documento, $passwd) {

return $this->usuarioRepository->autenticar($documento, $passwd);

}

Este es otro método público llamado autenticar que toma dos argumentos, $documento y $passwd. Luego, llama al método autenticar en $usuarioRepository, pasando estos dos argumentos. Este método verifica si el usuario con el documento y la contraseña proporcionados existe en la base de datos.

## Razones para Utilizar la Clase UsuarioService

La implementación de la clase UsuarioService aporta múltiples beneficios clave en la gestión de la lógica de negocio relacionada con los usuarios:

**Separación de Preocupaciones:** La clase UsuarioService separa claramente la lógica de negocio de la lógica de acceso a datos. Esto facilita la gestión del código y mejora la organización del proyecto.

**Facilidad de Mantenimiento y Evolución:** Al encapsular la lógica de negocio en una clase dedicada, cualquier cambio en las reglas de negocio se puede implementar de manera centralizada y sin afectar al resto de la aplicación. Esto simplifica el mantenimiento y la evolución del código.

**Reutilización de Código:** La clase UsuarioService permite la reutilización de la lógica de negocio en diferentes partes de la aplicación. Cualquier componente que necesite realizar operaciones relacionadas con los usuarios puede utilizar UsuarioService, promoviendo la reutilización y evitando la duplicación de código.

**Mejora de la Seguridad:** La clase UsuarioService puede implementar validaciones y otras reglas de negocio antes de interactuar con la base de datos. Esto proporciona una capa adicional de seguridad, asegurando que solo los datos válidos se persistan en la base de datos.

## Descripción de la Clase UsuarioModel

La clase UsuarioModel se encuentra en el archivo UsuarioModel.php bajo el espacio de nombres App\Models. Está diseñada para encapsular los datos de un usuario y proporcionar métodos para acceder y modificar estos datos de manera segura. A continuación, se describen sus componentes y funcionalidades clave:

public function \_\_construct($nroDocumento, $rol, $passwd) {

$this->nroDocumento = $nroDocumento;

$this->rol = $rol;

$passwdHASH = password\_hash($passwd, PASSWORD\_DEFAULT);

$this->passwd = $passwdHASH;

}

El constructor de la clase toma tres argumentos: $nroDocumento, $rol y $passwd. Al crear una nueva instancia de UsuarioModel, se asignan los valores proporcionados a las propiedades correspondientes. La contraseña se cifra utilizando la función password\_hash con el algoritmo PASSWORD\_DEFAULT para asegurar su almacenamiento seguro.

## Razones para Utilizar la Clase UsuarioModel

La implementación de la clase UsuarioModel proporciona varios beneficios clave en la gestión de datos de usuarios:

**Encapsulación de Datos:** La clase UsuarioModel encapsula los datos del usuario, proporcionando una estructura clara y organizada para su almacenamiento y manipulación. Esto facilita la gestión de los datos en diferentes partes de la aplicación.

**Seguridad en el Almacenamiento de Contraseñas:** Al utilizar la función password\_hash para cifrar las contraseñas, se asegura que las contraseñas se almacenen de manera segura. Esto protege los datos sensibles de los usuarios contra accesos no autorizados y ataques de fuerza bruta.

**Acceso y Modificación Controlados:** Los métodos getter y setter permiten el acceso controlado y la modificación de las propiedades del usuario. Esto asegura que cualquier cambio en los datos del usuario se realice de manera consistente y controlada.

**Mantenibilidad y Claridad del Código**: Al centralizar la lógica relacionada con los datos del usuario en una sola clase, se mejora la claridad y mantenibilidad del código. Cualquier cambio en la estructura de datos del usuario se puede realizar en un solo lugar, simplificando el mantenimiento y evolución de la aplicación.

**Reutilización de Código:** La clase UsuarioModel puede ser reutilizada en diferentes partes de la aplicación, promoviendo la reutilización de código y evitando la duplicación. Esto mejora la eficiencia del desarrollo y facilita la consistencia en la gestión de datos de usuarios.

**Flexibilidad y Escalabilidad:** La clase UsuarioModel proporciona una base flexible para la gestión de datos de usuarios. Si se necesitan agregar nuevas propiedades o cambiar la manera en que se gestionan los datos, estos cambios se pueden realizar de manera sencilla y sin afectar otras partes de la aplicación.

## Descripción de la Clase UsuarioController

La clase UsuarioController se encarga de recibir las solicitudes del usuario, procesarlas y devolver la respuesta adecuada. Está diseñada para interactuar con UsuarioService y manejar las operaciones relacionadas con los usuarios. A continuación, se describen sus componentes y funcionalidades clave:

public function crearUsuario() {

$usuario = new UsuarioModel(

$\_POST['nroDocumento'],

$\_POST['rol'],

$\_POST['passwd']

);

$this->usuarioService->crearUsuario($usuario);

}

Este método maneja la creación de un nuevo usuario. Toma los datos del formulario POST, crea una instancia de UsuarioModel y luego llama al método crearUsuario en UsuarioService para persistir el nuevo usuario en la base de datos.

public function autenticar() {

if ($this->usuarioService->autenticar($\_POST['documento'], $\_POST['passwd']) == false) {

echo "algo anduvo mal";

} else {

echo "<script>

localStorage.setItem('documento', '" . $this->usuarioService->autenticar($\_POST['documento'], $\_POST['passwd']). "');

window.location.href = '../../Public/inicio.html';

</script>";

$\_SESSION['logged'] = true;

$\_SESSION['nroDocumento'] = $\_POST['documento'];

echo "Sesion iniciada";

}

}

Este método maneja la autenticación del usuario. Toma los datos de inicio de sesión del formulario POST y llama al método autenticar en UsuarioService. Si la autenticación falla, muestra un mensaje de error. Si tiene éxito, guarda el documento del usuario en localStorage, inicia la sesión y redirige al usuario a la página de inicio.

public function logout() {

session\_destroy();

echo "<script>

localStorage.removeItem('documento');

window.location.href = '../../Public/inicio.html'; // Redirigir a la página de inicio

</script>";

}

Este método maneja el cierre de sesión del usuario. Destruye la sesión actual y elimina el documento del usuario de localStorage, luego redirige al usuario a la página de inicio.

# Razones para Utilizar MySQLi

La extensión MySQLi (MySQL Improved) es una versión mejorada de la extensión original MySQL de PHP, proporcionando una interfaz orientada a objetos para interactuar con bases de datos MySQL. A continuación, se detallan las razones clave para utilizar MySQLi:

**Interfaz Orientada a Objetos:** MySQLi proporciona una interfaz orientada a objetos, lo que facilita una programación más estructurada y modular. Esto es especialmente útil en aplicaciones grandes y complejas, donde la reutilización y el mantenimiento del código son esenciales.

**Consultas Preparadas:** Una de las características más importantes de MySQLi es el soporte para consultas preparadas. Las consultas preparadas ayudan a prevenir ataques de inyección SQL, ya que separan la lógica de la consulta de los datos que se envían a la base de datos. Esto mejora significativamente la seguridad de la aplicación.

**Compatibilidad con Características Recientes de MySQL:** MySQLi es compatible con las características más recientes de MySQL, lo que permite aprovechar las mejoras de rendimiento y las nuevas funcionalidades que ofrece MySQL. Esto es especialmente relevante si se utiliza una versión reciente de MySQL.

**Mejora en el Rendimiento:** MySQLi está optimizado para mejorar el rendimiento en comparación con la extensión original de MySQL. Esto se traduce en una mayor eficiencia en la gestión de conexiones y en la ejecución de consultas, lo cual es crucial para aplicaciones que requieren alta disponibilidad y rapidez en el acceso a los datos.

# Importancia de Usar GitHub para el Desarrollo en Equipo

GitHub es esencial para el desarrollo en equipo por varias razones clave:

1. Control de Versiones: Permite a los desarrolladores rastrear y revertir cambios en el código, facilitando el trabajo en paralelo y la integración de nuevas funcionalidades mediante ramas (branches) y fusiones (merges).
2. Colaboración: Pull requests y revisiones de código permiten la revisión, discusión y mejora del código antes de su integración, fomentando una cultura de colaboración y aprendizaje.
3. Seguridad y Control de Acceso: Controles de acceso detallados y análisis de seguridad protegen el código y los datos del proyecto.
4. Documentación y Comunicación: Facilita la creación de documentación y proporciona herramientas para una comunicación clara y eficiente entre los miembros del equipo.
5. Transparencia y Trazabilidad: Almacena todo el historial de cambios y discusiones, proporcionando trazabilidad completa y facilitando auditorías y la resolución de problemas.

GitHub mejora la eficiencia, la calidad del código y la colaboración en equipo, siendo fundamental para el éxito de los proyectos de software.

## Repositorio del Proyecto

Para centralizar todos los aspectos del desarrollo, hemos creado un repositorio en GitHub donde todo el equipo puede colaborar. El repositorio del proyecto se encuentra en la siguiente dirección:

[Repositorio del Proyecto](https://github.com/Proyecto3Fidat/ProyectoEgreso)

# Instructivo de Ejecución

## Prerrequisitos

1. Apache instalado y funcionando en tu equipo.
2. Composer instalado en tu equipo.

## Paso 1: Instalación de Composer

Instalación de Composer

1. Descarga Composer desde el [sitio oficial de Composer](https://getcomposer.org/).
2. Sigue las instrucciones de instalación proporcionadas en el sitio.

## Paso 2: Configurar el VirtualHost en Apache

1. Abre el archivo de configuración de Apache para VirtualHosts. Este archivo suele encontrarse en: C:\xampp\apache\conf\extra\httpd-vhosts.conf
2. Añade la siguiente configuración al archivo httpd-vhosts.conf. Asegúrate de cambiar las rutas en DocumentRoot y <Directory> por la ruta donde se encuentra tu programa.

<VirtualHost \*:80>

ServerAdmin webmaster@dummy-host.example.com

DocumentRoot "ruta del proyecto"

ServerName proyecto.localhost

ServerAlias proyecto.localhost

ErrorLog "logs/proyecto.localhost-error.log"

CustomLog "logs/proyecto.localhost-access.log" common

<Directory "ruta del proyecto">

Options Indexes FollowSymLinks

AllowOverride All

Require all granted

</Directory>

</VirtualHost>

## Paso 4: Reiniciar Apache

## Paso 5: Uso de Composer

1. Abre una consola de comandos (cmd).
2. Navega al directorio del proyecto
3. Ejecuta Composer para instalar las dependencias del proyecto: composer install

## Nota Importante sobre la Configuración del VirtualHost

Mientras se tenga configurado el VirtualHost como se ha descrito anteriormente, todas las comunicaciones dirigidas al puerto 80 en tu equipo serán redirigidas a proyecto.localhost.

## Detalles

* VirtualHost Configurado: Con la configuración de VirtualHost establecida en Apache, cualquier solicitud HTTP que llegue a tu máquina a través del puerto 80 se manejará según las reglas definidas para proyecto.localhost.
* ServerName y ServerAlias: Las directivas ServerName proyecto.localhost y ServerAlias proyecto.localhost aseguran que el dominio proyecto.localhost se resuelva y sea atendido por el servidor Apache configurado.
* Redirección de Comunicaciones: Como resultado, cualquier navegador web o aplicación que intente acceder a http://proyecto.localhost se conectará al DocumentRoot especificado.

## Implicaciones

* Desarrollo Local: Esta configuración es ideal para un entorno de desarrollo local, permitiendo probar y desarrollar la aplicación web sin conflictos de dominio.
* Desactivación Temporal: Si en algún momento deseas desactivar esta configuración, puedes comentar o eliminar la sección correspondiente en httpd-vhosts.conf y reiniciar Apache para que los cambios surtan efecto.

Recuerda que estos ajustes afectan únicamente el entorno local de tu máquina y no impactan en servidores de producción u otros dispositivos en la red a menos que se configure específicamente para ello.

# **Bibliografía**

* **Modelo Vista Controlador (MVC):**
  + MDN Web Docs - Glosario de MDN MVC <https://developer.mozilla.org/es/docs/Glossary/MVC>
  + TutorialsPoint. (s.f.). MVC Framework Introduction.<https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm>
  + Microsoft MVC de ASP.NET Core <https://learn.microsoft.com/es-es/aspnet/mvc/overview/older-versions-1/overview/asp-net-mvc-overview>
* **MySQLi:**
  + PHP.net. (s.f.). MySQLi Introduction.<https://www.php.net/manual/en/mysqli.intro.php>
  + W3Schools. (s.f.). PHP MySQLi.<https://www.w3schools.com/php/php_mysql_intro.asp>
  + Introduccion a MySQLi <https://www.ionos.mx/digitalguide/hosting/cuestiones-tecnicas/introduccion-a-mysqli/#:~:text=Continuar-,%C2%BFQu%C3%A9%20es%20MySQLi%20de%20PHP%3F,m%C3%A1s%20populares%20a%20nivel%20mundial>
* **GitHub:**
  + GitHub Guides. (s.f.). Hello World.<https://guides.github.com/activities/hello-world/>
  + GitHub Docs. (s.f.). Understanding the GitHub flow.<https://docs.github.com/en/get-started/quickstart/github-flow>
* **Apache:**
  + Apache HTTP Server Documentation. (s.f.). Virtual Hosts.<https://httpd.apache.org/docs/2.4/vhosts/>
  + DigitalOcean. (s.f.). How To Set Up Apache Virtual Hosts on Ubuntu 20.04.<https://www.digitalocean.com/community/tutorials/how-to-set-up-apache-virtual-hosts-on-ubuntu-20-04>
* **Composer:**
  + Composer Documentation. (s.f.). Getting Started.<https://getcomposer.org/doc/00-intro.md>
  + Packagist. (s.f.). The PHP Package Repository.<https://packagist.org/>
  + DesarrolloWeb - Composer guia tutorial <https://desarrolloweb.com/manuales/tutorial-composer.html>