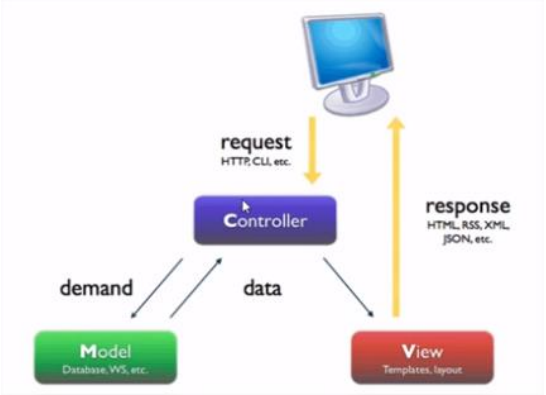
Definición de la arquitectura del proyecto formativo

Para la siguiente actividad los aprendices deberán tener muy presentes los conceptos ya vistos de Programación Orientada a Objetos y el patrón de diseño de arquitectura de software: MVC (Modelo Vista Controlador)



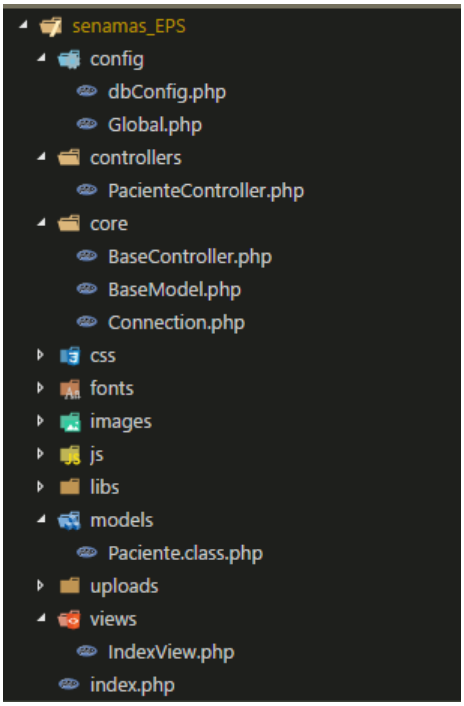
Actividad 1

Los aprendices deberán seguir el siguiente paso a paso para construir un proyecto PHP utilizando Programación Orientada a Objetos y una Patrón de Arquitectura de Software MVC. Con la ayuda del instructor deberá quedar muy clara toda la implementación ya que se utilizará como base para desarrollar los proyectos formativos con esta estructura. Para ello, los aprendices crearán un nuevo proyecto llamado: "Senamás_EPS"

Nuestro proyecto tendrá las siguientes carpetas, las cuales pueden ser creadas desde el explorador de Windows o desde del IDE seleccionado:

- **config**: aquí irán los archivos de configuración globales, etc.
- **controllers**: como sabemos en la arquitectura MVC los controladores se encargarán de recibir y filtrar datos que le llegan de las vistas, llamar a los modelos y pasar los datos de estos a las vistas. En este directorio colocaremos los controladores
- **core**: aquí colocaremos las clases base de las que heredarán por ejemplo controladores y modelos, y también podríamos colocar más librerías hechas por nosotros o por terceros, esto sería el núcleo del proyecto.

- **models**: aquí irán los modelos, para ser fieles al paradigma orientado objetos tenemos que tener una clase por cada tabla o entidad de la base de datos y estas clases servirán para crear objetos de ese tipo de entidad(por ejemplo crear un objeto usuario para crear un usuario en la BD). También tendremos modelos de consulta a la BD que contendrán consultas más complejas que estén relacionadas con una o varias entidades.
- **views**: aquí irán las vistas, es decir, donde se imprimirán los datos y lo que verá el usuario.
- **uploads**: en esta carpeta irán todas las imágenes que suba el usuario o que la aplicación requiera crear dinámicamente
- **css**: carpeta para adicionar todos los estilos de la aplicación
- **js**: carpeta para adicionar todos los JavaScripts de la aplicación
- **fonts**: carpeta que contendrá todas las fuentes necesarias en el sitio web
- **images:** carpeta que contendrá todas las imágenes de los diferentes layouts o plantillas de la aplicación
- **libs:** carpeta que manejará todas las librerías externas que utilicemos en nuestra aplicación
- **index.php** será el controlador frontal por el que pasará absolutamente todo en la aplicación.



Crear ficheros de configuración

En el directorio **config**, crearemos un archivo **Global.php** en el que irán constantes que luego nos servirán por ejemplo para establecer controladores y acciones por defecto (y todo lo que queramos ingresarle).

```php
<?php
define("CONTROLADOR_DEFECTO", "Paciente");
define("ACCION_DEFECTO", "index");
//Más constantes de configuración
```

Luego crearemos un archivo llamado **dbConfig.php**, el cual contendrá las variables que permitirán realizar la conexión a la Base de Datos

```php
<?php
define("DB_DRIVER", "mysql"); // database driver, mysql
define("DB_HOST", "localhost"); // db host
define("DB_NAME", "senamas_eps"); // database name
define("DB_USER", "root"); // database username
define("DB_PASS", ""); // database password
```

Crear las clases del núcleo

Primero crearemos la clase **Connection** que nos servirá para conectarnos a la base de datos utilizando el driver **PDO**. Luego también se podría implementar en otra versión conexión a bases de datos con un constructor de consultas como por ejemplo Eloquent (el que utiliza Laravel o algún otro ORM).

```php
<?php
include_once 'config/dbConfig.php';
abstract class Connection
{
    // Atributo que contiene la conexión a la BD con PDO
    protected $dbConnection;

    public function __construct()
    {
        try
        {
            // Se crea un objeto de la clase PDO para la conexión a la BD
            $this->dbConnection = new PDO(DB_DRIVER . ':host=' . DB_HOST . ';dbname=' .
DB_NAME, DB_USER, DB_PASS);
            $this->dbConnection->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        } catch (PDOException $ex) {
            echo $ex->getMessage();
            die();
        }
    }
}
```

Seguimos creando el archivo **BaseModel.php**, de esta clase heredarán los modelos que representen entidades, en el constructor llamaremos al constructor de la clase padre y tendremos tantos métodos como queramos para ayudarnos con las peticiones a la BD a través de los objetos que iremos creando. Manejarlo de este modo tiene la ventaja de que estos métodos pueden ser reutilizados en otras clases ya que le indicamos la tabla con la que queremos trabajar.

```php
<?php

class BaseModel extends Connection
{

    protected $table;

    public function __construct()
```

```
    {
        // Se llama al constructor de la clase Padre
        parent::__construct();
    }

    public function getAll()
    {
        try {
            // FETCH_OBJ
            $sql = $this->dbConnection->prepare("SELECT * FROM ".$this->table);

            // Ejecutamos
            $sql->execute();
            $resultSet = null;
            // Ahora vamos a indicar el fetch mode cuando llamamos a fetch:
            while ($row = $sql->fetch(PDO::FETCH_OBJ)) {
                $resultSet[] = $row;
            }
            return $resultSet;
        } catch (PDOException $ex) {
            echo $ex->getMessage();
            die();
        }
    }

    /*
    * Aqui podemos incluir los demás métodos que nos ayuden
    * a hacer operaciones con la base de datos de forma común
    */
}
```

La siguiente clase que crearemos es **ControladoresBase** de la cual heredarán los controladores, esta clase carga BaseModel y todos los modelos creados dentro de la carpeta model.

```
<?php

class BaseController
{

    public function __construct()
    {
        require_once 'Conection.php';
        require_once 'BaseModel.php';

        //Incluir todos los modelos
        /*foreach (glob("models/*.php") as $file) {
            require_once $file;
        }*/
        require_once 'models/Paciente.class.php';
    }

    // Crear los Métodos que sean comunes para todos los controladores
}
```

index.php

La única página que visita el usuario realmente es esta (index.php).

```php
<?php

# Configuración global
require_once 'config/Global.php';

# Base para los controladores
require_once 'core/BaseController.php';

# Se capturan el controlador y la acción que vienen por método GET
$controller = isset($_GET["controller"]) ? $_GET["controller"] : "";
$action = isset($_GET["action"]) ? $_GET["action"] : "";

# Se evalua el controlador que llega por URL, en caso que no llegue nada, se toma el
Controlador por defecto
switch (ucwords($controller)) {
    # EN la medidad que se incluyan más controladores, se deben referenciar en un case
    case 'Paciente':
        $controlador = 'PacienteController';
        break;
    default:
        $controlador = CONTROLADOR_DEFECTO;
        break;
}
$strFileController = 'controllers/' . $controlador . '.php';

# Si el archivo existe se procede a crear un objeto de dicha clase controlador
if (is_file($strFileController)) {
    # Se incluye el archivo del controlador para poder ser instanciado
    require_once $strFileController;
    # Se crea un objeto de la clase del controlador segun la peticion del usuario
    $controllerObj = new $controlador();

    # Si no llega ninguna accion por GET se toma la acción por defecto
    if ($action == "") {
        $action = ACCION_DEFECTO;
    }

    // Se comprueba que la acción o método que solicita el usuario existe en ese
controlador
    if (method_exists($controllerObj, $action)) {
```

```
        // Se carga la accion requerida, es decir, se llama al metodo del controlador
        $controllerObj->$action();
    } else {
        echo "No existe el método en el controlador";
    }
}
```

Modelos y objetos

Si queremos seguir el paradigma de la Programación Orientada a Objetos teóricamente deberíamos tener una clase por cada tabla de la base de datos que haga referencia a un objeto de la vida real, en este caso el objeto que crearíamos sería “Paciente” y el usuario tendría un nombre, un telefono, un estado, etc, esos serían los atributos del objeto y tendríamos un método get y set por cada atributo que servirán para establecer el valor de las propiedades y para conseguir el valor de cada atributo. Esta clase hereda de **BaseModel** y tiene un método **save** para guardar el usuario en la base de datos, podríamos tener otro método **update** que sería similar, etc. En el IDE seleccionado se puede generar los getters y setters

```
<?php

class Paciente extends BaseModel {
    private $documento;
    private $nombre;
    private $direccion;
    private $telefono;
    private $fecha_nacimiento;
    private $estado;
    private $genero;
    private $eps;
    private $email;
    private $password;

    public function __construct($doc=null, $nom=null, $dir=null, $tel=null,
$fec=null, $est=null, $gen=null, $_eps=null, $_email=null, $_pass=null){
        $this->table = "paciente"; // tabla asociada en la base de datos
        $this->documento = $doc;
        $this->nombre = $nom;
        $this->direccion = $dir;
        $this->telefono = $tel;
        $this->fecha_nacimiento = $fec;
        $this->estado = $est;
        $this->genero = $gen;
        $this->eps = $_eps;
        $this->email = $_email;
        $this->password = $_pass;
        parent::__construct();
    }
```

```php
    /**
     * Get the value of documento
     */
    public function getDocumento()
    {
        return $this->documento;
    }

    /**
     * Set the value of documento
     *
     * @return  self
     */
    public function setDocumento($documento)
    {
        $this->documento = $documento;
        return $this;
    }

    /**
     * Get the value of nombre
     */
    public function getNombre()
    {
        return $this->nombre;
    }

    /**
     * Set the value of nombre
     *
     * @return  self
     */
    public function setNombre($nombre)
    {
        $this->nombre = $nombre;
        return $this;
    }

    /**
     * Get the value of direccion
     */
    public function getDireccion()
    {
        return $this->direccion;
    }

    /**
     * Set the value of direccion
     *
     * @return  self
```

```php
     */
    public function setDireccion($direccion)
    {
        $this->direccion = $direccion;
        return $this;
    }

    /**
     * Get the value of telefono
     */
    public function getTelefono()
    {
        return $this->telefono;
    }

    /**
     * Set the value of telefono
     *
     * @return  self
     */
    public function setTelefono($telefono)
    {
        $this->telefono = $telefono;
        return $this;
    }

    /**
     * Get the value of fecha_nacimiento
     */
    public function getFecha_nacimiento()
    {
        return $this->fecha_nacimiento;
    }

    /**
     * Set the value of fecha_nacimiento
     *
     * @return  self
     */
    public function setFecha_nacimiento($fecha_nacimiento)
    {
        $this->fecha_nacimiento = $fecha_nacimiento;
        return $this;
    }

    /**
     * Get the value of estado
     */
    public function getEstado()
    {
        return $this->estado;
```

```php
    }

    /**
     * Set the value of estado
     *
     * @return  self
     */
    public function setEstado($estado)
    {
        $this->estado = $estado;
        return $this;
    }

    /**
     * Get the value of genero
     */
    public function getGenero()
    {
        return $this->genero;
    }

    /**
     * Set the value of genero
     *
     * @return  self
     */
    public function setGenero($genero)
    {
        $this->genero = $genero;
        return $this;
    }

    /**
     * Get the value of eps
     */
    public function getEps()
    {
        return $this->eps;
    }

    /**
     * Set the value of eps
     *
     * @return  self
     */
    public function setEps($eps)
    {
        $this->eps = $eps;
        return $this;
    }
```

```php
    /**
     * Get the value of email
     */
    public function getEmail()
    {
        return $this->email;
    }

    /**
     * Set the value of email
     *
     * @return  self
     */
    public function setEmail($email)
    {
        $this->email = $email;

        return $this;
    }

    /**
     * Get the value of password
     */
    public function getPassword()
    {
        return $this->password;
    }

    /**
     * Set the value of password
     *
     * @return  self
     */
    public function setPassword($password)
    {
        $this->password = $password;

        return $this;
    }

    public function save()
    {
        // Preparar la consulta para insertar un paciente en la BD
         $sql = $this->dbConnection->prepare("INSERT INTO paciente (documento, nombre, direccion, telefono, fecha_nacimiento, estado, genero, eps, email, password) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)");

        $documento = $this->getDocumento();
        $nombre = $this->getNombre();
        $direccion = $this->getDireccion();
        $telefono = $this->getTelefono();
```

```
        $fecha_nac = $this->getFecha_nacimiento();
        $estado = $this->getEstado();
        $genero = $this->getGenero();
        $eps = $this->getEps();
        $email = $this->getEmail();
        $pass = $this->getPassword();

        $sql->bindParam(1, $documento);
        $sql->bindParam(2, $nombre);
        $sql->bindParam(3, $direccion);
        $sql->bindParam(4, $telefono);
        $sql->bindParam(5, $fecha_nac);
        $sql->bindParam(6, $estado);
        $sql->bindParam(7, $genero);
        $sql->bindParam(8, $eps);
        $sql->bindParam(9, $email);
        $sql->bindParam(10, $pass);
        // Excecute
        $sql->execute();
    }
}
```

Los controladores

Los crearemos en la carpeta **controller**, en este caso crearemos **PacienteController**.

```
<?php

class PacienteController extends BaseController
{

    public function __construct()
    {
        parent::__construct();
    }

    public function index()
    {
        //Creamos el objeto usuario
        $paciente_obj = new Paciente();
        //Conseguimos todos los usuarios
        $allPacientes = $paciente_obj->getAll();

        $especialidad = new Especialidad();
        $lista_especialidades = $especialidad->getAll();
        require_once 'views/indexView.php';
    }

    public function create()
    {
```

```
        if (isset($_POST["txtNombre"])) {
            $nombre = isset($_POST['txtNombre']) ? $_POST['txtNombre'] : "";
            $documento = isset($_POST['txtDocumento']) ? $_POST['txtDocumento'] : "";
            $fecha_nac = isset($_POST['txtFechaNacimiento']) ? $_POST['txtFechaNacimiento'] : "";
            $direccion = isset($_POST['txtDireccion']) ? $_POST['txtDireccion'] : "";
            $telefono = isset($_POST['txtTelefono']) ? $_POST['txtTelefono'] : "";
            $estado = isset($_POST['txtEstado']) ? $_POST['txtEstado'] : "";
            $genero = isset($_POST['txtGenero']) ? $_POST['txtGenero'] : "";
            $eps = isset($_POST['txtEps']) ? $_POST['txtEps'] : "";
            $email = isset($_POST['txtEmail']) ? $_POST['txtEmail'] : "";
            $password = isset($_POST['txtPassword']) ? $_POST['txtPassword'] : "";

            // Crear Objeto Paciente con los datos del formulario
            $paciente_obj = new Paciente($documento, $nombre, $direccion, $telefono,
$fecha_nac, $estado, $genero, $eps, $email, $password);

            // Se llama al metodo que inserta en la base de datos.
            $paciente_obj->save();

        }
        header("Location:index.php?controller=Paciente&action=index");
    }

    public function hola()
    {
        echo "Hola ADSI día";
    }
}
```

Las vistas

En este caso crearemos la vista **IndexView.php**

```
<!doctype html>
<html lang="es">

<head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="libs/bootstrap-dist/css/bootstrap.min.css">
    <link rel="stylesheet" href="css/mycss.css">
</head>
    <body>
        <header class="container-fluid backg1">
            HEADER MENU
        </header></br>
```

```
        <div class="container">
            <div class="row">
                <div class="col-lg-4">
                    <form action = "<?php echo "index.php?controller=paciente&action=create"; ?>"
method="post">
                        <h2>Añadir Paciente</h2>
                        <hr/>
                        <div class="form-group">
                            <label for="txtDocumento">Documento: </label>
                            <input type="text" class="form-control" id="txtDocumento"
name="txtDocumento" placeholder="">
                        </div>
                        <div class="form-group">
                            <label for="txtNombre">Nombre: </label>
                            <input type="text" class="form-control" id="txtNombre" name="txtNombre"
placeholder="">
                        </div>
                        <div class="form-group">
                            <label for="txtFechaNacimiento">Fecha Nacimiento: </label>
                            <input type="date" class="form-control" id="txtFechaNacimiento"
name="txtFechaNacimiento" placeholder="">
                        </div>
                        <div class="form-group">
                            <label for="txtDireccion">Dirección: </label>
                            <input type="text" class="form-control" id="txtDireccion"
name="txtDireccion" placeholder="">
                        </div>
                        <div class="form-group">
                            <label for="txtTelefono">Telefono: </label>
                            <input type="text" class="form-control" id="txtTelefono"
name="txtTelefono" placeholder="">
                        </div>
                        <div class="form-group">
                            <label for="txtGenero">Genero: </label>
                            <select class="form-control" id="txtGenero" name="txtGenero">
                                <option value="M">Masculino</option>
                                <option value="F">Femenino</option>
                            </select>
                        </div>
                        <div class="form-group">
                            <label for="txtEstado">Estado: </label>
                            <select class="form-control" id="txtEstado" name="txtEstado">
                                <option value="activo">Activo</option>
                                <option value="inactivo">Inactivo</option>
                                <option value="multado">Con multa</option>
                            </select>
                        </div>
                        <div class="form-group">
                            <label for="txtEps">EPS: </label>
                            <input type="text" class="form-control" id="txtEps" name="txtEps"
placeholder="">
```

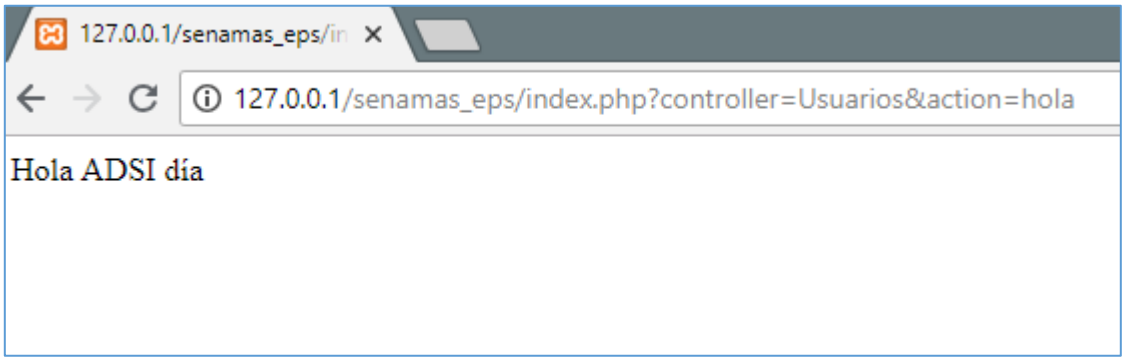
```
                        </div>
                        <div class="form-group">
                            <label for="txtEmail">Email: </label>
                            <input type="text" class="form-control" id="txtEmail" name="txtEmail"
placeholder="">
                        </div>
                        <div class="form-group">
                            <label for="txtPassword">Contraseña: </label>
                            <input type="password" class="form-control" id="txtPassword"
name="txtPassword" placeholder="">
                        </div>
                        <button type="submit" class="btn btn-success">Guardar</button>
                    </form>
                </div>
                <div class="col-lg-8">
                    <div>
                        <h2>Pacientes</h2>
                        <hr/>
                    </div>

                    <div class="row">
                        <div class="col-lg-2 ">Documento</div>
                        <div class="col-lg-3 ">Nombre</div>
                        <div class="col-lg-3 ">Telefono</div>
                        <div class="col-lg-2 ">Eps</div>
                        <div class="col-lg-2 ">Opciones</div>
                    </div>
                    <?php
                    if (isset($allPacientes)) {
                        foreach ($allPacientes as $row) { //recorremos el array de objetos y
obtenemos el valor de las propiedades
                            ?>
                                <div class="row">
                                    <div class="col-lg-2"><?php echo $row->documento ?></div>
                                    <div class="col-lg-3"><?php echo $row->nombre; ?></div>
                                    <div class="col-lg-3"><?php echo $row->telefono; ?></div>
                                    <div class="col-lg-2"><?php echo $row->eps; ?></div>
                                    <div class="col-lg-2">
                                        <div class="right col-1">
                                            <a href = "<?php echo
"index.php?controller=Paciente&action=borrar&id=" . $row->documento; ?>" class="btn btn-danger">Borrar</a>
                                        </div>
                                    </div>
                                </div><br>
                            <?php
                        }
                    }
                    ?>
                </div>
            </div>  <!-- row -->
            <br><br>
```

```
        </div>
        <footer class="container-fluid backg1">
            Senamás - Ejemplo PHP con POO/MVC/PDO - <?php echo date("Y"); ?>
        </footer>

        <!-- Optional JavaScript -->
        <!-- jQuery first, then Popper.js, then Bootstrap JS -->
        <script src="js/jquery.min.js"></script>
        <script src="js/popper.min.js"></script>
        <script src="libs/bootstrap-dist/js/bootstrap.min.js"></script>
    </body>
</html>
```

Si ejecutamos el sitio web en estos momentos cargaría el controlador y acción por defecto. Para acceder a otros controladores y acciones por la url.



Actividad 2

Con el fin de interiorizar y comprender mejor la estructura del proyecto a realizar, los aprendices deberán crear un diagrama de clases que represente la estructura del proyecto; cada clase deberá tener los atributos y métodos identificados (incluyendo constructores y métodos SET y GET