

- [PedalPoint](#)
 - [Día 1: 19/11/2025](#)
 - [Día 2: 20/11/2025](#)

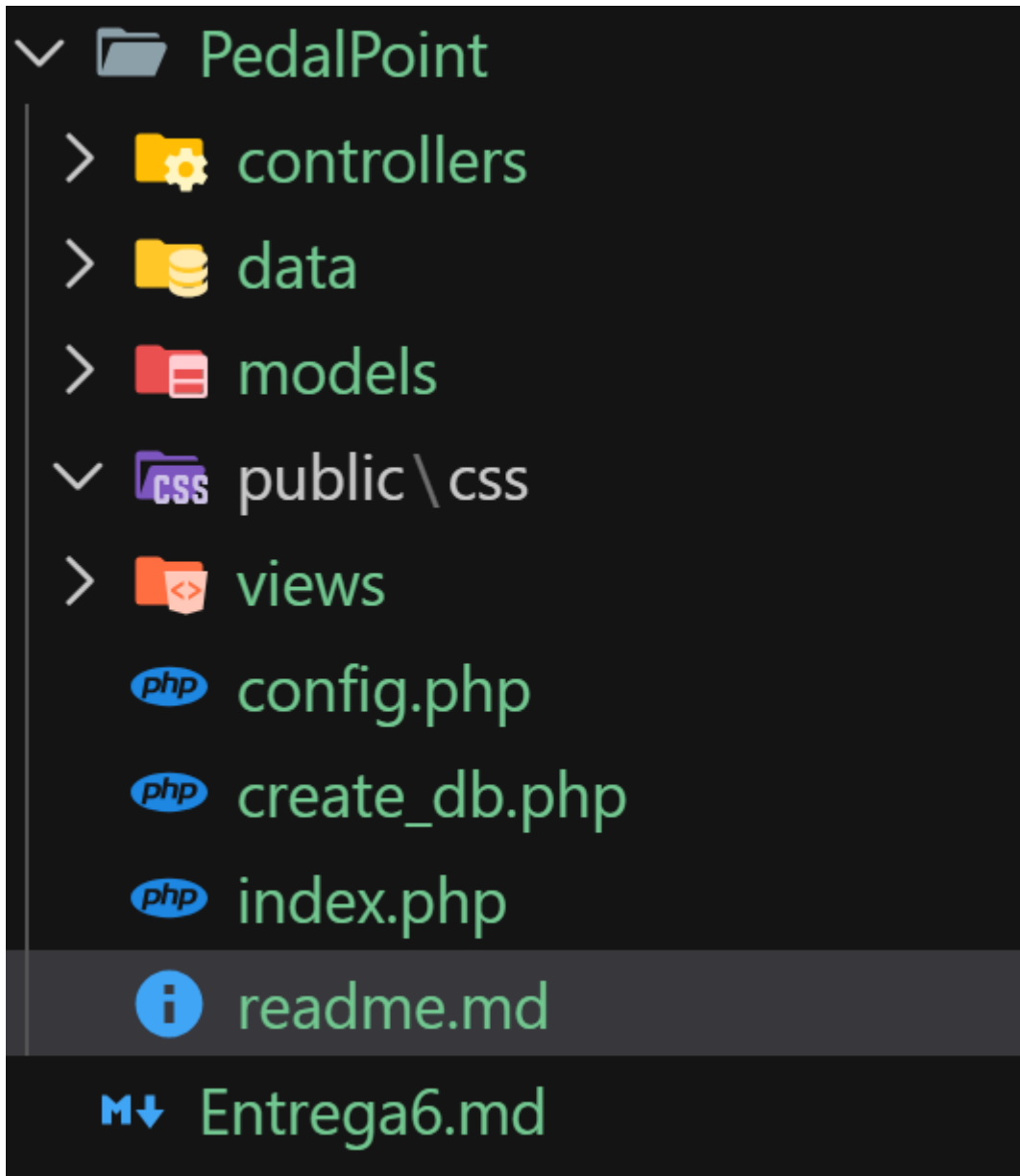
PedalPoint

Para este proyecto se nos pidió realizar un CRUD con MVC y acceso a base de datos con SQLITE. A lo largo de este readme, voy a ir desglosando mi trabajo día a día

Día 1: 19/11/2025

Este día para mí es el que más me gusta especialmente, cogí papel y boli y me creé las clases necesarias con los atributos, para que a la hora de crear la base de datos lo tuviese todo más claro, también me apunte los primeros pasos para programar mi proyecto:

1º: Creamos la estructura del proyecto






2º Creamos los archivos de configuración, y de base de datos con un usuario administrados por defecto ya en ella

Dwes_Maria > UD5HerramientasWeb > PedalPoint > create_db.php

```
1  <?php
2  declare(strict_types=1);
3
4  require_once __DIR__ . '/config.php';
5
6  //Creamos la carpeta data si no existe
7  $dataDir = __DIR__ . '/data';
8  if (!is_dir(filename: $dataDir)) {
9      mkdir(directory: $dataDir, permissions: 0777, recursive: true);
10 }
11
12 $pdo = getPdo();
13
14 //Creación tabla usuarios
15 $pdo->exec(statement: "
16     CREATE TABLE IF NOT EXISTS users (
17         id INTEGER PRIMARY KEY AUTOINCREMENT,
18         username TEXT NOT NULL UNIQUE,
19         password_hash TEXT NOT NULL
20     )
21 ");
22
23 // Crear tabla de bicicletas
24 $pdo->exec(statement: "
25     CREATE TABLE IF NOT EXISTS bikes (
26         id INTEGER PRIMARY KEY AUTOINCREMENT,
27         marca TEXT NOT NULL,
28         modelo TEXT NOT NULL,
29         tipo TEXT NOT NULL,
30         color TEXT NOT NULL,
31         price REAL NOT NULL
32     )
33 ");
34
35
36
37 //Creación de admin por defecto si no existe
38
39 $stmt = $pdo->prepare(query: 'SELECT COUNT(*) FROM users WHERE username = :u');
40 $stmt->execute(params: [':u' => 'adminMaria']);
41 $exists = (int)$stmt->fetchColumn();
42
43 if ($exists === 0) {
44     $passwordHash = password_hash(password: 'adminMaria', algo: PASSWORD_DEFAULT);
45     $insert = $pdo->prepare(query: 'INSERT INTO users (username, password_hash) VALUES (:u, :p)');
46     $insert->execute(params: [
47         ':u' => 'adminMaria',
48         ':p' => $passwordHash,
49     ]);
50     echo "Base de datos creada y usuario 'adminMaria' con contraseña 'adminMaria' generado." . PHP_EOL;
51 } else {
52     echo "Base de datos ya existente. Usuario 'admin' ya creado." . PHP_EOL;
53 }
54
```

3º Ejecutamos el archivo para que se nos cree la base de datos

> bikes			Filte 	Filte 	Filter... 
> sqlite_sequence		1	1	adminMaria	\$2y\$10\$YyhnRBGsG1eL/a6q9F4pV.ucu6cr6GNMHWIb3
> users	+	2			

Una vez creado esto, ya podemos programar como sería la entrada a nuestra aplicación, Iniciar Sesión

Tenemos ahora un modelo admin

```
Dwes_Maria > UD5HerramientasWeb > PedalPoint > models > Admin.php > ...  
1  <?php  
2  declare(strict_types=1);  
3  
3 references | 0 implementations  
4  class Admin {  
5      2 references  
6      private int $id;  
7      2 references  
8      private string $username;  
9      2 references  
10     private string $password_hash;  
11  
1 reference | 0 overrides  
12     public function getId(): int {  
13         return $this->id;  
14     }  
15  
1 reference | 0 overrides  
16     public function getUsername(): string {  
17         return $this->username;  
18     }  
19  
1 reference | 0 overrides  
20     public function getPasswordHash(): string {  
21         return $this->password_hash;  
22     }  
23  
1 reference | 0 overrides  
24     public static function findByUsername(PDO $pdo, string $username): ?Admin  
25     {  
26         $stmt = $pdo->prepare(query: 'SELECT * FROM users WHERE username = :u LIMIT 1');  
27         $stmt->execute(params: ['u' => $username]);  
28         $row = $stmt->fetch(mode: PDO::FETCH_ASSOC);  
29  
30         if (!$row) {  
31             return null;  
32         }  
33  
34         $admin = new Admin();  
35         $admin->id = (int)$row['id'];  
36         $admin->username = $row['username'];  
37         $admin->password_hash = $row['password_hash'];  
38  
39         return $admin;  
40     }  
41  
42 }  
43  
44 ?>  
45
```

Con su correspondiente controller, con todas las funciones necesarias para el admin como loguearse o desloguearse

```

1  <?php
2  declare(strict_types=1);
3  require_once __DIR__ . '/../models/Admin.php';
4
5  1 reference | 0 implementations
6  class AuthAdminController{
7      2 references
8      private PDO $pdo;
9
10     1 reference | 0 overrides
11     public function __construct(PDO $pdo)
12     {
13         $this->pdo = $pdo;
14     }
15
16     /**
17      * Muestra el formulario de login.
18      */
19     1 reference | 0 overrides
20     public function login(): void
21     {
22         $error = $_GET['error'] ?? null;
23         require __DIR__ . '/../views/authAdmin/login.php';
24     }
25
26     /**
27      * Procesa el login.
28      */
29     1 reference | 0 overrides
30     public function doLogin(): void
31     {
32         ensureSession();
33
34         $username = trim(string: $_POST['username'] ?? '');
35         $password = trim(string: $_POST['password'] ?? '');
36
37         if ($username === '' || $password === '') {
38             $error = 'Debes rellenar usuario y contraseña.';
39             require __DIR__ . '/../views/authAdmin/login.php';
40             return;
41         }
42
43         $admin = Admin::findByUsername(pdo: $this->pdo, username: $username);
44
45         if (!$admin || !password_verify(password: $password, hash: $admin->getPasswordHash())) {
46             $error = 'Usuario o contraseña incorrectos.';
47             require __DIR__ . '/../views/authAdmin/login.php';
48             return;
49         }
50
51         $_SESSION['admin'] = [
52             'id' => $admin->getId(),
53             'username' => $admin->getUsername(),
54         ];
55
56         header(header: 'Location: index.php?c=product&a=index');
57         exit;
58     }
59
60     /**
61      * Cierra la sesión.
62      */
63     1 reference | 0 overrides
64     public function logout(): void
65     {
66         ensureSession();
67         $_SESSION = [];
68         if (ini_get(option: 'session.use_cookies')) {
69             $params = session_get_cookie_params();

```

```

73     $params = session_get_cookie_params();
74     setcookie(
75         name: session_name(),
76         value: '',
77         expires_or_options: time() - 42000,
78         path: $params['path'],
79         domain: $params['domain'],
80         secure: $params['secure'],
81         httponly: $params['httponly']
82     );
83 }
84 session_destroy();
85 header(header: 'Location: index.php?c=authAdmin&a=login');
86 exit;
87 }

```

Tenemos también nuestro archivo index.php, este va a ser el punto de partida de nuestra web y además el enrutador que nos guíara a través de ella

```

<?php
declare(strict_types=1);

require_once __DIR__ . '/config.php';
require_once __DIR__ . '/models/Admin.php';
require_once __DIR__ . '/controllers/AuthAdminController.php';
require_once __DIR__ . '/controllers/ProductController.php';
require_once __DIR__ . '/controllers/BikesController.php';

//si no se ha creado la base de datos, la creamos
if (!file_exists(filename: __DIR__ . '/data/app.sqlite')) {
    require_once __DIR__ . '/create_db.php';
}

//iniciamos la sesión si no está iniciada
ensureSession();

$pdo = getPdo();

// Obtenemos el controlador y la acción desde la URL
$controllerName = $_GET['c'] ?? null;
$action = $_GET['a'] ?? null;

// Si no hay usuario autenticado, forzamos ir al login
//in_array($action, ['login', 'doLogin'], true sirve para permitir acceder a
// esas acciones sin estar autenticado
if (!isset($_SESSION['admin']) && !($controllerName === 'authAdmin' && in_array(needle: $action, haystack: ['login', 'doLogin'], strict: true))) {
    $controllerName = 'authAdmin';
    $action = 'login';
}

// Valores por defecto
// Si no hay sesión, vamos a auth/login
if ($controllerName === null) {
    $controllerName = isset($_SESSION['admin']) ? 'product' : 'auth';
}
if ($action === null) {
    $action = ($controllerName === 'authAdmin') ? 'login' : 'index';
}

//creamos las instancias de los controladores
$authAdminController = new AuthAdminController(pdo: $pdo);
$productController = new ProductController(pdo: $pdo);

switch ($controllerName) {
    case 'authAdmin':
        switch ($action) {
            case 'login':
                $authAdminController->login();
                break;
            case 'doLogin':
                $authAdminController->doLogin();
                break;
            case 'logout':
                $authAdminController->logout();
                break;
            default:
                http_response_code(response_code: 404);
                echo 'Acción de autenticación no encontrada';
        }
        break;
}

```

Resultado final:

[Login](#)

Bienvenido a PedalPoint. Inicie Sesión

Usuario Contraseña

Usuario por defecto: **adminMaria** — Contraseña: **adminMaria**
(Se ejecuta create_db.php de forma automatica al carga INDEX).

Tienda sobre bicicletas y mucho más de Maria

Como resultado final tenemos un inicio de sesión completamente funcional

Dia 2: 20/11/2025

Para el dia de hoy mi objetivo era hacer el crud completo del objeto bicicleta

Para ello me cree la base de datos de la bicicleta

```
// Crear tabla de bicicletas
$stmt => $pdo->exec(statement: "
    CREATE TABLE IF NOT EXISTS bikes (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        marca TEXT NOT NULL,
        modelo TEXT NOT NULL,
        tipo TEXT NOT NULL,
        color TEXT NOT NULL,
        price REAL NOT NULL
    )
");
```

Despues me cree el modelo de bicicleta

```
1  <?php
2  declare(strict_types=1);
3
4  5 references | 0 implementations
5  class Bikes
6  {
7      0 references
8      public int $id;
9      0 references
10     public string $marca;
11     0 references
12     public string $modelo;
13     0 references
14     public string $tipo;
15     0 references
16     public string $color;
17     0 references
18     public float $price;
19
20     /**
21      * Obtiene todas las bicicletas.
22      */
23     1 reference | 0 overrides
24     public static function all(PDO $pdo): array
25     {
26         $stmt = $pdo->query(query: 'SELECT * FROM bikes ORDER BY id DESC');
27         return $stmt->fetchAll(mode: PDO::FETCH_ASSOC);
28     }
29
30     /**
31      * Busca una bicicleta por ID.
32      */
33     1 reference | 0 overrides
34     public static function find(PDO $pdo, int $id): ?array
35     {
36         $stmt = $pdo->prepare(query: 'SELECT * FROM bikes WHERE id = :id');
37         $stmt->execute(params: [':id' => $id]);
38         $row = $stmt->fetch(mode: PDO::FETCH_ASSOC);
39
40         return $row ? : null;
41     }
42
43     /**
44      * Crea una nueva bicicleta.
45      */
46     1 reference | 0 overrides
47     public static function create(PDO $pdo, string $marca, string $modelo , string $tipo , string $color , float $price): void
48     {
49         $stmt = $pdo->prepare(query: '
50             INSERT INTO bikes (marca, modelo, tipo, color, price)
51             VALUES (:marca, :modelo, :tipo, :color, :price)
52         ');
53         $stmt->execute(params: [
54             ':marca' => $marca,
55             ':modelo' => $modelo,
56             ':tipo' => $tipo,
57             ':color' => $color,
58             ':price' => $price,
59         ]);
60     }
61
62     /**
63      * Actualiza una bicicleta existente.
64      */
65     1 reference | 0 overrides
66     public static function update(PDO $pdo, int $id, string $marca, string $modelo , string $tipo , string $color , float $price): void
67     {
68         $stmt = $pdo->prepare(query: '
69             UPDATE bikes
70             SET marca = :marca, modelo = :modelo, tipo = :tipo, color = :color, price = :price
71             WHERE id = :id
72         ');
73         $stmt->execute(params: [
74             ':marca' => $marca,
75             ':modelo' => $modelo,
76             ':tipo' => $tipo,
77             ':color' => $color,
78             ':price' => $price,
79             ':id' => $id,
80         ]);
81     }
82
83     /**
84      * Elimina una bicicleta
85      */
86     1 reference | 0 overrides
87     public static function delete(PDO $pdo, int $id): void
88     {
89         $stmt = $pdo->prepare(query: 'DELETE FROM bikes WHERE id = :id');
90         $stmt->execute(params: [':id' => $id]);
91     }
92 }
```


Despues me cree el controlador de bicicleta

Dwes_Maria > UD5HerramientasWeb > PedalPoint > controllers > BikesController.php > ...

```
1 <?php
2 declare(strict_types=1);
3 require_once __DIR__ . '/../models/Bikes.php';
4
5 1 reference | 0 implementations
6 class BikesController
7 {
8     6 references
9     private PDO $pdo;
10
11     1 reference | 0 overrides
12     public function __construct(PDO $pdo)
13     {
14         $this->pdo = $pdo;
15     }
16
17     6 references
18     private function requireLogin(): void
19     {
20         ensureSession();
21         if (!isset($_SESSION['admin'])) {
22             header('Location: index.php?c=authadmin&a=login');
23             exit;
24         }
25     }
26
27     1 reference | 0 overrides
28     public function index(): void
29     {
30         $this->requireLogin();
31         $bikes = Bikes::all($pdo: $this->pdo);
32         require __DIR__ . '/../views/bikes/index.php';
33     }
34
35     1 reference | 0 overrides
36     public function create(): void
37     {
38         $this->requireLogin();
39         $bike = ['id' => null, 'marca' => '', 'modelo' => '', 'tipo' => '', 'color' => '', 'price' => ''];
40         $action = 'store';
41         require __DIR__ . '/../views/bikes/form.php';
42     }
43
44     1 reference | 0 overrides
45     public function store(): void
46     {
47         $this->requireLogin();
48
49         $marca = trim(string: $_POST['marca'] ?? '');
50         $modelo = trim(string: $_POST['modelo'] ?? '');
51         $tipo = trim(string: $_POST['tipo'] ?? '');
52         $color = trim(string: $_POST['color'] ?? '');
53         $priceStr = trim(string: $_POST['price'] ?? '');
54
55         if ($marca === '' || $modelo === '' || $tipo === '' || $color === '' || $priceStr === '') {
56             $error = 'Todos los campos son obligatorios.';
57             $bikes = ['id' => null, 'marca' => $marca, 'modelo' => $modelo, 'tipo' => $tipo, 'color' => $color, 'price' => $priceStr];
58             $action = 'store';
59             require __DIR__ . '/../views/bikes/form.php';
60             return;
61         }
62
63         $price = (float)$priceStr;
64
65         Bikes::create($pdo: $this->pdo, marca: $marca, modelo: $modelo, tipo: $tipo, color: $color, price: $price);
66
67         header('Location: index.php?c=bikes&a=index');
68         exit;
69     }
70
71     1 reference | 0 overrides
72     public function edit(): void
73     {
74         $this->requireLogin();
75
76         $id = isset($_GET['id']) ? (int)$_GET['id'] : 0;
77         $bikes = Bikes::find($pdo: $this->pdo, id: $id);
78
79         if (!$bikes) {
80             http_response_code(response_code: 404);
81             echo 'Bicicleta no encontrada';
82             return;
83         }
84
85         $action = 'update';
86         require __DIR__ . '/../views/bikes/form.php';
87     }
88
89     1 reference | 0 overrides
90     public function update(): void
91     {
92         $this->requireLogin();
93
94         $id = isset($_POST['id']) ? (int)$_POST['id'] : 0;
95         $bike = ['id' => $id, 'marca' => trim(string: $_POST['marca'] ?? ''), 'modelo' => trim(string: $_POST['modelo'] ?? ''), 'tipo' => trim(string: $_POST['tipo'] ?? ''), 'color' => trim(string: $_POST['color'] ?? ''), 'price' => trim(string: $_POST['price'] ?? '')];
96         $action = 'update';
97         require __DIR__ . '/../views/bikes/form.php';
98     }
99 }
```

```

86     $marca = trim(string: $_POST['marca'] ?? '');
87     $modelo = trim(string: $_POST['modelo'] ?? '');
88     $tipo = trim(string: $_POST['tipo'] ?? '');
89     $color = trim(string: $_POST['color'] ?? '');
90     $priceStr = trim(string: $_POST['price'] ?? '');
91
92     if ($marca === '' || $modelo === '' || $tipo === '' || $color === '' || $priceStr === '') {
93         $error = 'Todos los campos son obligatorios.';
94         $bikes = ['id' => null, 'marca' => $marca, 'modelo' => $modelo, 'tipo' => $tipo, 'color' => $color, 'price' => $priceStr];
95         $action = 'update';
96         require __DIR__ . '/../views/bikes/form.php';
97         return;
98     }
99
100     $price = (float)$priceStr;
101
102     Bikes::update(pdo: $this->pdo, id: $id, marca: $marca, modelo: $modelo, tipo: $tipo, color: $color, price: $price);
103
104     header(header: 'Location: index.php?c=bikes&a=index');
105     exit;
106 }
107
108 1 reference | 0 overrides
109 public function delete(): void
110 {
111     $this->requireLogin();
112
113     $id = isset($_GET['id']) ? (int)$_GET['id'] : 0;
114     if ($id > 0) {
115         Bikes::delete(pdo: $this->pdo, id: $id);
116     }
117
118     header(header: 'Location: index.php?c=bikes&a=index');
119     exit;
120 }

```

Las vistas de la bicicleta tanto el index como el formulario de creacion

```

Dwes_Maria > UD5HerramientasWeb > PedalPoint > views > bikes > index.php
1  <?php require __DIR__ . '/../layout/header.php'; ?>
2
3  <h1>Listado de bicicletas</h1>
4
5  <p>
6      <a href="index.php?c=bikes&a=create" class="button">Nueva bicicleta</a>
7  </p>
8
9  <?php if (empty($bikes)): ?>
10     <p>No hay bicicletas todavía.</p>
11 <?php else: ?>
12     <table class="table">
13         <thead>
14             <tr>
15                 <th>ID</th>
16                 <th>Marca</th>
17                 <th>Modelo</th>
18                 <th>Tipo</th>
19                 <th>Color</th>
20                 <th>Precio (€)</th>
21             </tr>
22         </thead>
23         <tbody>
24             <?php foreach ($bikes as $b): ?>
25                 <tr>
26                     <td><?= htmlspecialchars(string: $b['id']) ?></td>
27                     <td><?= htmlspecialchars(string: $b['marca']) ?></td>
28                     <td><?= htmlspecialchars(string: $b['modelo']) ?></td>
29                     <td><?= htmlspecialchars(string: $b['tipo']) ?></td>
30                     <td><?= htmlspecialchars(string: $b['color']) ?></td>
31                     <td><?= number_format(num: (float)$b['price'], decimals: 2, decimal_separator: ',', thousands_sep: '.') ?></td>
32                     <td>
33                         <a href="index.php?c=bikes&a=edit&id=<?= urlencode(string: $b['id']) ?>">Editar</a>
34                         <a href="index.php?c=bikes&a=delete&id=<?= urlencode(string: $b['id']) ?>"
35                             onclick="return confirm('¿Seguro que quieres eliminar este producto?');">
36                             Eliminar
37                         </a>
38                     </td>
39                 </tr>
40             <?php endforeach; ?>
41         </tbody>
42     </table>
43 <?php endif; ?>
44
45 <?php require __DIR__ . '/../layout/footer.php'; ?>

```

Dwes_Maria > UD5HerramientasWeb > PedalPoint > views > bikes > form.php

```
1  <?php require __DIR__ . '/../layout/header.php'; ?>
2
3  <h1><?= $action === 'store' ? 'Nueva bicicleta' : 'Editar bicicleta' ?></h1>
4
5  <?php if (!empty($error)): ?>
6      <div class="alert alert-error">
7          <?= htmlspecialchars(string: $error) ?>
8      </div>
9  <?php endif; ?>
10
11 <form method="post" action="index.php?c=bikes&a=<?= htmlspecialchars(string: $action) ?>" class="form">
12     <?php if (!empty($bikes['id'])): ?>
13         <input type="hidden" name="id" value="<?= htmlspecialchars(string: $bikes['id']) ?>">
14     <?php endif; ?>
15
16     <label>
17         Marca
18         <select name="marca" required>
19             <option value="">Selecciona una marca</option>
20             <option value="Trek" <?= ($bikes['marca'] ?? '') === 'Trek' ? 'selected' : '' ?>>Trek</option>
21             <option value="Specialized" <?= ($bikes['marca'] ?? '') === 'Specialized' ? 'selected' : '' ?>>Specialized
22             </option>
23             <option value="Giant" <?= ($bikes['marca'] ?? '') === 'Giant' ? 'selected' : '' ?>>Giant</option>
24             <option value="Cannondale" <?= ($bikes['marca'] ?? '') === 'Cannondale' ? 'selected' : '' ?>>Cannondale
25             </option>
26             <option value="Orbea" <?= ($bikes['marca'] ?? '') === 'Orbea' ? 'selected' : '' ?>>Orbea</option>
27         </select>
28     </label>
29
30     <label>
31         Modelo
32         <select name="modelo" required>
33             <option value="">Selecciona un modelo</option>
34             <option value="Marlin 5" <?= ($bikes['modelo'] ?? '') === 'Marlin 5' ? 'selected' : '' ?>>Marlin 5</option>
35             <option value="Rockhopper" <?= ($bikes['modelo'] ?? '') === 'Rockhopper' ? 'selected' : '' ?>>Rockhopper
36             </option>
37             <option value="Escape 3" <?= ($bikes['modelo'] ?? '') === 'Escape 3' ? 'selected' : '' ?>>Escape 3</option>
38             <option value="Bad Boy 3" <?= ($bikes['modelo'] ?? '') === 'Bad Boy 3' ? 'selected' : '' ?>>Bad Boy 3</option>
39             <option value="Orbea Alma" <?= ($bikes['modelo'] ?? '') === 'Orbea Alma' ? 'selected' : '' ?>>Orbea Alma
40             </option>
41         </select>
42     </label>
43
44     <label>
45         Tipo
46         <select name="tipo" required>
47             <option value="">Selecciona un tipo</option>
48             <option value="Montaña" <?= ($bikes['tipo'] ?? '') === 'Montaña' ? 'selected' : '' ?>>Montaña</option>
49             <option value="Carretera" <?= ($bikes['tipo'] ?? '') === 'Carretera' ? 'selected' : '' ?>>Carretera</option>
50             <option value="Híbrida" <?= ($bikes['tipo'] ?? '') === 'Híbrida' ? 'selected' : '' ?>>Híbrida</option>
51             <option value="Eléctrica" <?= ($bikes['tipo'] ?? '') === 'Eléctrica' ? 'selected' : '' ?>>Eléctrica</option>
52             <option value="BMX" <?= ($bikes['tipo'] ?? '') === 'BMX' ? 'selected' : '' ?>>BMX</option>
53         </select>
54     </label>
55
56     <label>
57         Color
58         <input type="text" name="color" value="<?= htmlspecialchars(string: (string) ($bikes['color'] ?? '')) ?>" required>
59     </label>
60
61     <label>
62         Precio (€)
63         <input type="number" step="0.01" min="0" name="price" value="<?= htmlspecialchars(string: (string) $product['price']) ?>"
64             required>
65     </label>
66
67     <button type="submit">
68         <?= $action === 'store' ? 'Crear' : 'Actualizar' ?>
69     </button>
70
71     <a href="index.php?c=product&a=index" class="button button-secondary">Volver</a>
72 </form>
73
74 <?php require __DIR__ . '/../layout/footer.php'; ?>
```

Y actualicé el index.php con el case producto que es la vista de bienvenida, a través del header el admin podrá elegir a donde quiere ir y bicicleta

```
case 'product':
    switch ($action) {
        case 'index':
            $productController->index();
            break;
    }
    break;

case 'bikes':
    $bikesController = new BikesController(pdo: $pdo);
    switch ($action) {
        case 'index':
            $bikesController->index();
            break;
        case 'create':
            $bikesController->create();
            break;
        case 'store':
            $bikesController->store();
            break;
        case 'edit':
            $bikesController->edit();
            break;
        case 'update':
            $bikesController->update();
            break;
        case 'delete':
            $bikesController->delete();
            break;
        default:
            http_response_code(response_code: 404);
            echo 'Acción de bicicletas no encontrada';
    }
    break;

default:
    http_response_code(response_code: 404);
    echo 'Controlador no encontrado';
```

Hola, adminMaria [Bicicletas](#) [Cerrar sesión](#)

Bienvenido a PedalPoint !!!

Seleccione en el menú hacia donde quiere ir

Listado de bicicletas

[Nueva bicicleta](#)

No hay bicicletas todavía.

Tienda sobre bicicletas y mucho más de Maria