

**Centro educativo**

Código	Centro	Concello	Ano académico
15005397	I.E.S. Fernando Wirtz Suárez	A Coruña	2024-2025

**Ciclo formativo**

Código da familia profesional	Familia profesional	Código do ciclo formativo	Ciclo formativo	Grao	Réxime
IFC	Informática e comunicacións	SIFC01	Desenvolvemento de Aplicacións Web	Superior	Ordinario

**Módulo profesional e unidades formativas de menor duración (\*)**

Código MP/UF	Nome
MP0374	Proxecto de Desenvolvemento de Aplicacións Web Equivalencia en créditos ECTS: 5.

**Profesorado responsable**

<b>Tutor</b>	Constantino García Ulla – Javier Ulla Berdullas - Pablo Irimia Rega
--------------	---

**Alumno**

<b>Alumno</b>	Nicolás García Moreira
---------------	------------------------

**Datos do Proxecto**

<b>Título</b>	<i>Autocare HUB</i>
---------------	---------------------

**CONTROL DE VERSIONS:**

Versión	Data	Observacións
V0		
V1		
V2		

**Índice:**

<b>1.Objetivo .....</b>	<b>3</b>
<b>2.Descripción.....</b>	<b>3</b>
<b>3.Alcance.....</b>	<b>3</b>
<b>4.Planificación.....</b>	<b>4</b>
<b>5.Medios a emplegar .....</b>	<b>4</b>
<b>6.Presuposto .....</b>	<b>4</b>
<b>7.Título.....</b>	<b>5</b>
<b>8.Execución .....</b>	<b>5</b>

## 1. Obxectivo

A idea de desenvolver esta aplicación, e debido a que os talleres a de onde levo eu o meu coche, son poucos eficientes en canto a organización de citas, e ao envío de facturas, esta inspirada en outros programas de xestión de albaráns e de facturas, como pode ser stel order entre outras.

Esta aplicación persigue dous obxectivos:

- Ser unha ferramenta intuitiva de usar, tanto por un usuario de taller, como tamén os usuarios solicitantes de citas e facturas, para que poda solicitar y visualizar de forma sinxela las citas y las facturas.
- Tamén pretende axilizar y alixear la carga de traballo, para todos los talleres que sexan demandantes de esta aplicación, por experiencia propia, tanto nas prácticas como en outros traballos que tiven, ter que estar atendendo correos e chamadas, acabas perdendo moito do tempo da xornada laboral e podendo dedicarlle pouco tempo ao que e mais importante.

## 2. Descripción

AutocareHUB, es una aplicación que consta de 3 partes:

- A primeira mais enfocada a parte de un usuario, que ten o apartado de poder xestionar as citas, poder xestionar as facturas, e tamén poder ter un chat de comunicación entre o usuario e o taller.
- A segunda, e a mais enfocada a xestión do taller, donde ten o visor das citas, a xestión das facturas, e tamén o chat de comunicación cos clientes que ten o taller.
- A terceira e última, e a mais enfocada a xestión da aplicación, para poder dar de alta os talleres, e tamén completar os detalles do mesmo.

A motivación persoal que me levou a facer este proxecto, e que os talleres aos que habitúo levar o coche, non teñen un bo sistema de xestión, tanto para as facturas, como para a parte de levar o control das citas, e creo que sería bo para os talleres, ter un bo software para facer este tipo de cousas.

## Uso de la aplicación

En caso de que un usuario no tenga una cuenta en la aplicación, tendrá que acceder al formulario de registro de la cuenta, para poder crear una y acceder a la aplicación

LOGO  
  
50px x 50px

Español ▼

### REGISTRO

Correo Electrónico

Nombre completo

Contraseña

Confirmar contraseña

Método de notificación

Contacto

Registrarse

¿Ya tienes una cuenta? [Inicia sesión](#)

Una vez tenemos una cuenta creada, el próximo paso que tenemos que hacer es iniciar sesión

LOGO  
  
50px x 50px

Español ▼

### Iniciar Sesión

Correo Electrónico

Contraseña

Iniciar Sesión

¿No tienes una cuenta? [Crea una aquí](#)

Unha vez completado o inicio de sesión, xa estaremos visualizando a aplicación de AutocareHUB



Unha vez pasemos o login, poderemos movernos libremente pola web, tanto para pedir citas, como para poder ver as nosas facturas, e tamen poder interactuar co taller.

Para poder rexistrar un vehículo, teremos que ir a opción de “Registrar vehículo” e completar todos os campos

Pedir CitasRegistrar VehículoVer FacturasChats

Registrar Vehículo

Marca:

Modelo:

Año:

Matrícula:

Registrar Vehículo

Vehículos Registrados

Marca- Modelo  
Año:  
Matrícula  
EditarEliminar

Marca- Modelo  
Año:  
Matrícula  
EditarEliminar

Despois, poderemos ver acceder as facturas dende a opción de “Ver facturas”, e poderemos seleccionalas e descargalas en PDF

Pedir CitasRegistrar VehículoVer FacturasChats

Fecha	Total	Estado	Acciones	
-----	-----	-----	Seleccionar Factura	Editar PDF
-----	-----	-----	Seleccionar Factura	Editar PDF

Tamén poderemos visitar o apartado de chats, donde poderemos falar cos talleres que hai na aplicación para facer as consultas que nos queiramos.

Pedir Citas Registrar Vehículo Ver Facturas Chats

Conversaciones

Nuevo Chat

Nombre taller

Mensaje

Hola! En que podemos ayudar?

Mensaje de prueba!

Escriba el mensaje

Enviar

Esto eran as vistas para a parte do usuario, agora veremos as vistas que temos dispoñibles para o taller

A primeira vista que temos é a de poder xerar as facturas para as citas que os usuarios solicitaron

[Generar Factura](#)
[Ver Citas](#)
[Estadísticas](#)
[Chats](#)

---

### Generador de Facturas

Nueva factura

ID de la cita

Estado

No hay items en la factura

Añadir Item

Descripción	Cantidad	Precio Unitario	Iva %
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>

Crear Factura
Reiniciar

Facturas				Actualizar
ID	Fecha	Total	Estado	Cliente
----	-----	-----	-----	-----
----	-----	-----	-----	-----

Citas del taller					
ID	Fecha	Vehículo	Servicio	Estado	Usuario
----	-----	-----	-----	-----	-----
----	-----	-----	-----	-----	-----

A segunda que temos dispoñible a de ver as propias citas que os usuarios solicitaron

Generar Factura Ver Citas Estadísticas Chats					
Citas del taller					
Actualizar Citas					
Usuario	Vehículo	Fecha Inicio	Fecha Fin	Descripción	Estado
-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----



Xa mais orientado a parte da vision estadística do taller, e dicir, cantas facturas se emitiron, canto foi o total facturado, canto e o media das facturas, cantas temos cobradas e cantas temos pendientes...

Generar Factura Ver Citas Estadísticas Chats

Total Facturas  
--

Total Facturado  
---,--€

Promedio por factura  
---,--€

Estado Facturas  
Pendientes: X  
Pagadas:X

Facturas del período			
Fecha	Importe	Estado	Cliente
-----	-----	-----	-----
-----	-----	-----	-----

Despois, como xa sucedía cos usuarios, temos a parte do chat, para poder comunicarse cos usuarios que nos están a solicitar información sobre os servizos que ten o taller

Pedir Citas Registrar Vehículo Ver Facturas Chats

Conversaciones  
Nuevo Chat

Nombre taller  
Mensaje

Hola! En que podemos ayudar?

Mensaje de prueba!

Escriba el mensaje

Enviar

### 3. Alcance

O alcance da aplicación que estou a desenvolver, sería implementar algunha funcionalidade mais, como podería ser a de poder enviar un mensaxe de texto no momento no que se pide a cita para que o usuario teña mais presente que ten unha cita, e así evitar un pouco o absentismo nas citas, xa que pola información que teño, só acontece bastante esta condición.

Outras das funcionalidades que podería implementar neste proxecto, e o de poder opinar sobre as reparacións que fixeron no taller, e ter un ranking dos mellores talleres, para que así os usuarios podan seleccionar os mellores talleres para levar o seu coche, e tamén así os talleres se poñerían as pilas, xa que se están na parte baixa da clasificación, a xente non iría aos seus talleres tendo que pechalo, xa que non o poderían manter.

Tamén se podería desenvolver a funcionalidade de poder programar citas recorrentes, e dicir, que se podan programar citas cada 2 meses, cada 6 meses e así, para que os usuarios non teñan que estar pendentes de andar pedindo citas.

Tamen se pode implementar, a funcionalidade de que se un usuario fixo o cambio de aceite nun taller da aplicación, que se estime un tempo para que volva a ter que cambiar o aceite, e que antes de que ese tempo transcorra, que se lle mande unha mensaxe ao usuario avisando de eso.

A funcionalidade que eu considero mais importante a desenvolver e un sistema de notificacións ao usuario, de todos os pasos que vai transcorrendo en toda o vehículo, cando se esta a realizar a reparación, de cando está listo o vehículo, e tamén para no momento no que se lle xenere a factura da reparación, lle chegue unha notificación, para que non teña que estar tan pendiente de entrar na aplicación para poder visualizar a factura.

### 4. Planificación

Para a organización de este proxecto, voume axudar de la plataforma TRELLO.

Trello é unha ferramenta visual de xestión de proxectos baseada no sistema Kanban, que facilita a organización, planificación e seguimento de tarefas dun xeito intuitivo e colaborativo. É amplamente utilizada en equipos de desenvolvemento de software e outros ámbitos que precisan dunha xestión áxil, xa que permite dividir o traballo en tarxetas facilmente manexables e visualizables en taboleiros.

Unha das grandes vantaxes de Trello é a súa flexibilidade para adaptarse a diferentes metodoloxías, incluíndo SCRUM e Kanban. Cada proxecto represéntase mediante un taboleiro, que pode conter varias listas que representan fases do fluxo de traballo, tales como "Por facer", "En progreso" e "Finalizado". Dentro de cada lista, as tarxetas representan tarefas individuais ou ítems de traballo.

Trello permite personalizar cada tarxeta con descricións detalladas, listas de verificación (checklists), datas de vencemento, etiquetas de cores, arquivos adxuntos e comentarios, o que facilita unha comunicación clara e centralizada entre os membros do equipo. As tarxetas poden asignarse a diferentes usuarios, o que axuda a definir responsabilidades e distribuír a carga de traballo de forma transparente.

Aínda que Trello non ten por defecto conceptos como "epics" ou "sprints" como Jira, é posible estruturar estes elementos mediante listas e etiquetas. Por exemplo, un sprint pode representarse como unha lista específica no taboleiro, onde se agrupan todas as tarefas planificadas para ese período. Tamén se poden empregar etiquetas para categorizar tarefas por funcionalidade, prioridade ou estado.

Ademais, Trello conta con Power-Ups (extensións e plugins) que amplían as súas capacidades, permitindo integrar calendarios, diagramas de Gantt, informes de produtividade, automatizacións (con Butler), e conexión con ferramentas externas como Slack, GitHub ou Google Drive,

facilitando un ecosistema colaborativo moi completo.

Unha funcionalidade clave é a xestión dinámica do fluxo de traballo, xa que as tarxetas poden moverse facilmente entre listas para reflectir o progreso real, como "Pendente", "En desenvolvemento", "En revisión" ou "Bloqueada". Isto ofrece unha visión clara e actualizada do estado do proxecto para todos os membros, favorecendo a transparencia e a rápida identificación de cuellos de botella.

Na planificación de sprints con Trello, téñense en conta a capacidade do equipo e as prioridades do proxecto, asignando tarefas de xeito coherente para que o sprint teña obxectivos claros e alcanzables. Os comentarios e a asignación de responsables permiten un seguimento constante e axustes áxiles conforme avanza o traballo.

Para este proxecto, vouno organizar por sprints.

No 1º sprint, que vai durar desde o día 24/03/2025 hasta o 20/04/2025, vou comezar a facer a memoria do proxecto, a maiores de facer a guía de estilos, e o deseño e creación da base de datos.

No 2º sprint, que terá unha duración estimada de 15 días, e dicir, dende o día 20/04/2025 hasta o 4/05/2025, voume dedicar a facer toda a parte de front-end, que veñen sendo todos os formularios, toda a xestión de facturas, toda a xestión de citas, e o chat de comunicación entre os usuarios de taller e os clientes.

No 3º sprint, que ten unha duración de 20 días, que ven sendo dende o día 05/05/2025, hasta o 25/05/2025, voume dedicar a facer toda a parte de xestión de datos dende o backend, onde se engloba toda a tanto a inserción de datos, como a recuperación, como a actualización de datos.

Nos días que faltan hasta realizar a entrega final do proxecto, dedicareime a comprobar que todas as funcionalidades estean correctamente implementadas.

Tamén vou a retocar os estilos de cousas que ao mellor non están da mellor forma posible, e en xeral, afinando o que sería a aplicación en xeral como se fose unha aplicación real de unha empresa.

## 5. Medios a empregar

Os medios que utilizarei para este proxecto serán os seguintes

- > Ordenador para o desenvolver a aplicación
- > Plataforma de trello para a organización das tarefas a desenvolver.
- > Para o deseño da base de datos usarei a aplicación DIA
- > Para o deseño da interfaz web, usarase Wireframe.cc
- > Para probar as rutas e o funcionamento de PHP, usarase Postman
- > Para o deseño do logo usarase Inkscape e GIMP
- > Para el desarrollo de código se usará VS Code, con alguna extensión para formatear el código.

Linguaxes:

Para o lado do navegador, usarase Angular, que engloba as seguintes linguaxes de programación para os seus compoñentes: HTML, CSS e TypeScript. Estes compoñentes son reutilizables, polo que poden empregarse onde o programador o desexe. Empregarase Angular 18.2.15, sen ningún paquete externo adicional. Para o lado do servidor, usarase PHP.

O persoal que vai a estar dedicado a este proxecto vai a ser de 1 persona, de unha duración de 2 meses e medio.

## 6. Presupuesto

O presuposto do proxecto é:

Mano de obra		
Duración do desenvolrollo	2 meses y medio(10 semanas)	
Horas Semanales	20	
Total de horas adicadas	200	
Precio Hora	12,5€	
Total	2500€	
Software utilizado		
Herramienta	Finalidad de la misma	Precio
Dia	Desarrollo de la base de datos, diseño lógico y conceptual	0
Wireframe CC	Desarrollo de la interfaz	0
VS Code	Editor de código para el desarrollo de la aplicación	0
Total		0€
Planificación y elaboración de documentación		
Tarea	Horas	Coste
-Planificación de sprints -Planificación e creación da guía de estilos -Creación dos apartados da memoria -Creación de documento de como despregar a web	15	500
TOTAL PRESUPUESTO		3000€

## 7. Título

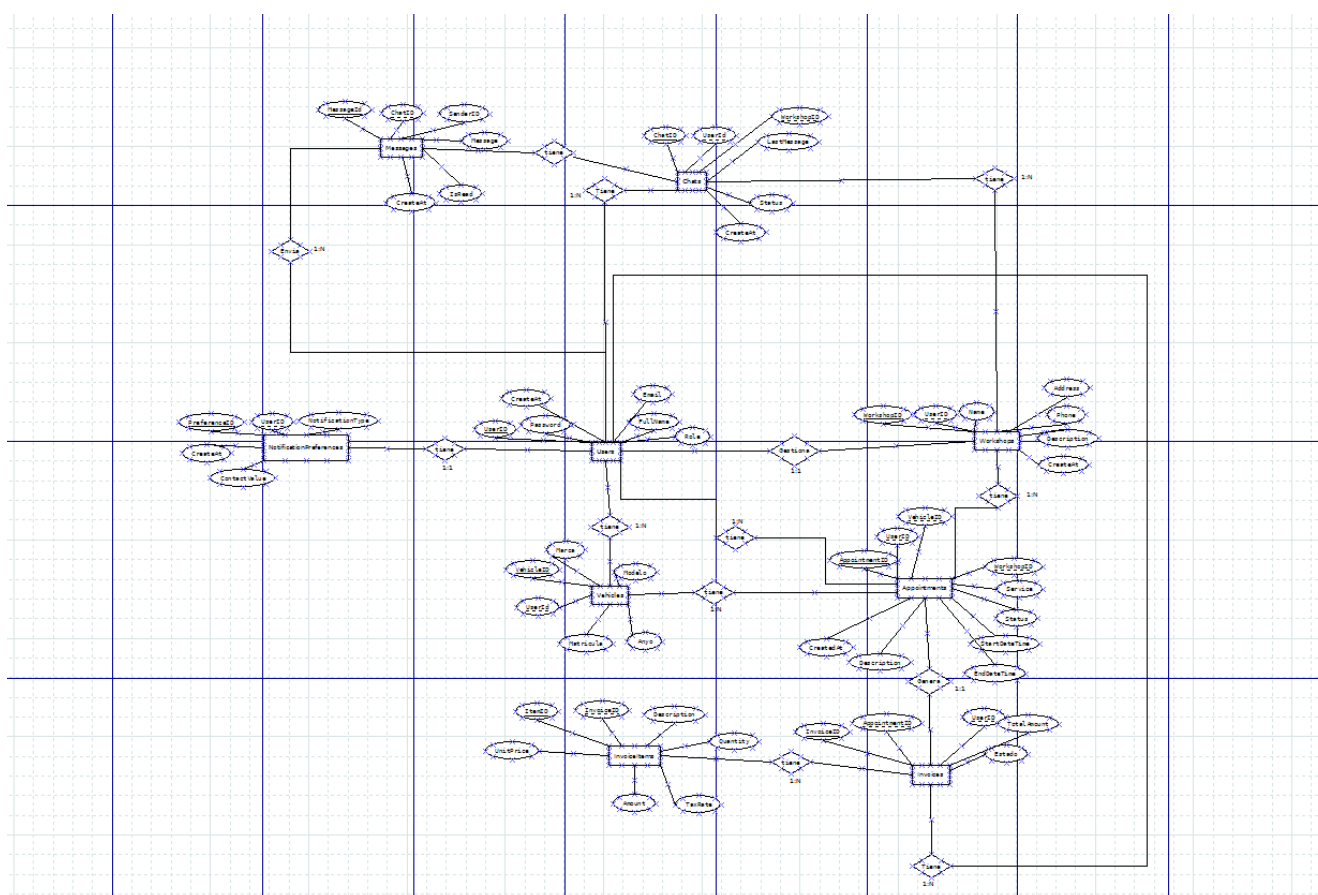
Autocare HUB

## 8. Execución

Neste apartado, vou a profundizar un pouco mais no que e o funcionamento da aplicación

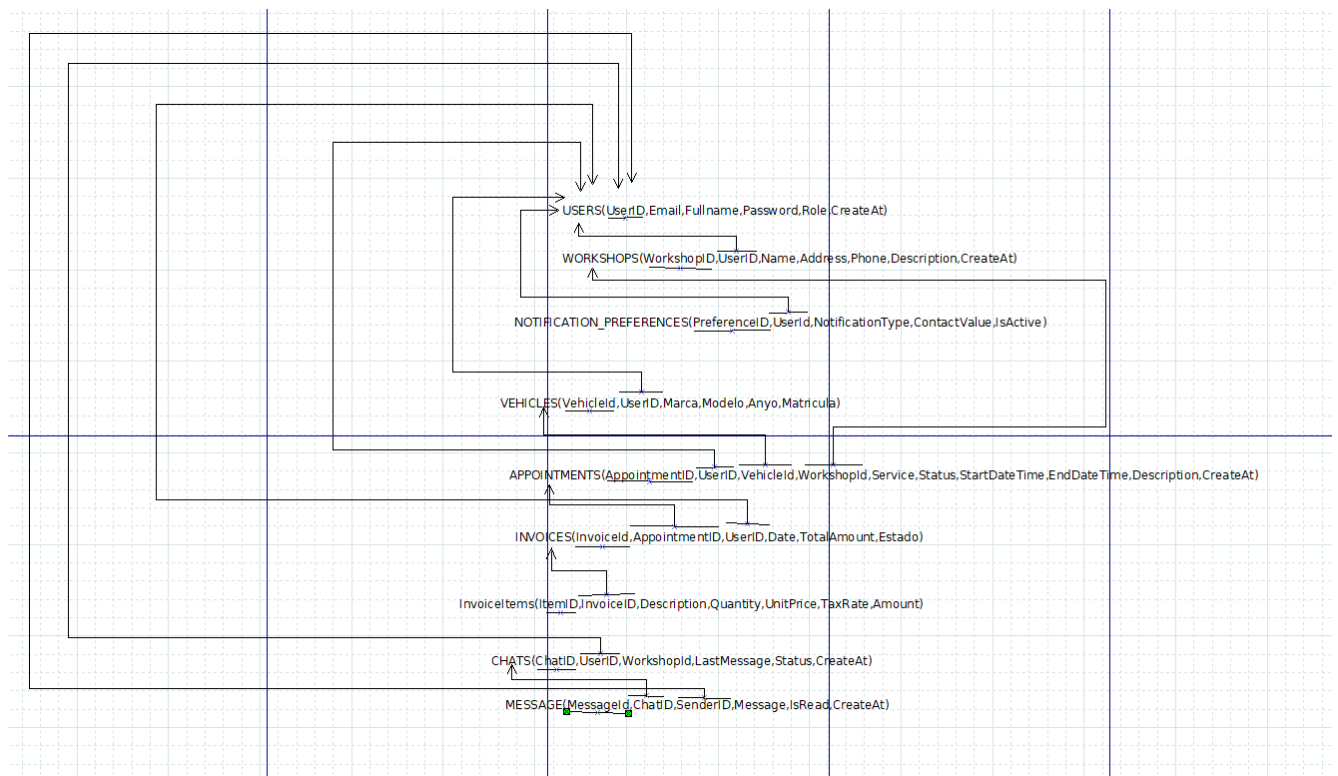
### 1.Diseño de base de datos

Esquema Conceptual:



\*NOTA: o esquema conceptual e lóxico da base de datos está na carpeta do repositorio, xa que aquí non se ve ben, esta na carpeta de (ProyectoFinCiclo/DiseñoBaseDatos)

## Esquema lóxico:



## Detalle das restricións da base de datos:

- Se un **usuario** é eliminado, elimínanse tamén todos os seus datos relacionados: os seus vehículos, citas, preferencias de notificación, facturas, conversacións, mensaxes e, no caso de ser taller, o taller correspondente. O identificador do usuario (UserID) é único e non se pode modificar.
- Se se elimina un **vehículo**, elimínanse todas as citas asociadas a ese vehículo. O identificador do vehículo (VehicleID) é único e non se pode actualizar.
- Se se elimina unha **cita**, elimínanse tamén todas as facturas relacionadas. O código da cita (AppointmentID) é único e inmutable.
- Se se elimina unha **factura**, elimínanse automaticamente todos os ítems asociados. O identificador da factura (InvoiceID) é único e non se pode modificar.
- Se un **taller** é eliminado, tamén se eliminan todas as citas e conversacións nas que participa. O identificador do taller (WorkshopID) é único e non se pode actualizar.
- Se se elimina unha **preferencia de notificación**, pérdese o medio de contacto correspondente. O identificador da preferencia (PreferenceID) e o UserID asociado son únicos e non se poden modificar.
- Se se elimina un **chat**, elimínanse tamén todas as mensaxes contidas nese chat. O código do chat (ChatID) e o identificador da mensaxe (MessageID) son únicos e non se poden actualizar.

## 2.BACKEND:

Para o desenvolvemento do backend, utilizarase PHP, mais concretamente a versión 8.2.12.

Para solucionar o problema de CORS, que é "Cross-Origin Resource Sharing", o "Intercambio de Recursos de Oríxenes Cruzados" en galego. Isto é un mecanismo de seguridade que teñen os navegadores, para controlar de forma máis sinxela a xestión de acceso a datos, imáxenes, e scripts.

```
cors.php M X
PHP > config > cors.php > ...
1  <?php
2  function configureCors() {
3      $allowedOrigins = ['http://localhost', 'http://localhost:4200'];
4      $origin = $_SERVER['HTTP_ORIGIN'] ?? '';
5
6      if (in_array($origin, $allowedOrigins)) {
7          header("Access-Control-Allow-Origin: $origin");
8      }
9
10     header("Access-Control-Allow-Credentials: true");
11     header("Access-Control-Allow-Methods: GET, POST, PUT, OPTIONS");
12     header("Access-Control-Allow-Headers: Content-Type, Authorization, X-Requested-With, content-type");
13     header("Content-Type: application/json; charset=UTF-8");
14
15     if ($_SERVER['REQUEST_METHOD'] === 'OPTIONS') {
16         http_response_code(200);
17         exit();
18     }
19 }
20 ?>
```

En este ficheiro, o único que estamos a realizar é añadirlle unhas cabeceiras, a todas as solicitudes, para solucionar o problema anterior mencionado.

Unha vez estas cabeceiras estén na solicitude presentes, xa teremos o acceso aos datos.

Para a conexión da base de datos, farase mediante un arquivo que se chama Database.php

```
database.php X
PHP > config > database.php > ...
1  <?php
2  class Database {
3      private $host = "localhost";
4      private $db_name = "AutoCareHub";
5      private $username = "hmi";
6      private $password = "hmi";
7      public $conn;
8
9      public function getConnection() {
10         $this->conn = null;
11         $this->conn = new mysqli($this->host, $this->username, $this->password, $this->db_name);
12         if ($this->conn->connect_error) {
13             return null;
14         }
15         return $this->conn;
16     }
17 }
18
```

En caso de que queiramos modificar a conexión tanto ao servidor a BD, como o usuario e a contraseña, teremos que modificalo en este ficheiro

Para controlar la autenticación del usuario para el acceso a datos, estaremos usando un controlador

```
AuthController.php M X
PHP > controllers > AuthController.php > AuthController
1 <?php
2 // Importar las dependencias necesarias para el controlador de autenticación
3 require_once __DIR__ . '/../models/User.php';
4 require_once __DIR__ . '/../config/database.php';
5
6 class AuthController {
7     private $conn;
8     private $userModel; // Modelo de usuario para interactuar con la base de datos
9     private const MIN_PASSWORD_LENGTH = 6;
10
11     public function __construct() {
12         session_start(); // Iniciar la sesión
13         $db = new Database();
14         $this->conn = $db->getConnection(); // Obtener la conexión a la base de datos
15
16         $this->userModel = new User($this->conn); // Inicializar el modelo de usuario
17     }
18
19     // Método para manejar el inicio de sesión
20     public function login($input) {
21         define('MAX_LOGIN_ATTEMPTS', 5); // Máximo número de intentos de inicio de sesión
22         define('LOGIN_TIMEOUT', 1800); // Tiempo de espera en segundos tras demasiados intentos fallidos
23         define('MIN_PASSWORD_LENGTH', 6); // Longitud mínima de la contraseña
24
25         // Validar el email y la contraseña proporcionados
26         $email = filter_var($input["email"] ?? '', FILTER_VALIDATE_EMAIL);
27         $password = $input["password"] ?? '';
28
29         if (!$email) {
30             return $this->response(400, "El formato del email no es válido");
31         }
32
33         if (strlen($password) < MIN_PASSWORD_LENGTH) {
34             return $this->response(400, "La contraseña debe tener al menos " . MIN_PASSWORD_LENGTH . " caracteres");
35         }
36     }
37 }
```

Aquí podemos ver que tenemos definida una constante para el mínimo de longitud de caracteres, estaremos también definiendo el máximo de intentos máximos para poder iniciar sesión, y el tiempo que tarda en volver a estar disponible la cuenta que se bloqueó, por logearse varias veces con una contraseña incorrecta.

También está a verificar que el email que se está a introducir es un correo que cumple los requisitos.

En caso de que el email no tenga el formato correcto, devolverá un error, a parte de front.

También en caso de que la longitud mínima no cumpla los requisitos

Obtendremos la conexión a base de datos, que la obtenemos del fichero de Database.php



```

AuthController.php M X
PHP > controllers > AuthController.php > AuthController
6   class AuthController {
20   public function login($input) {
78       };
79   }
80
81   // Método para manejar el registro de usuarios
82   public function register($input) {
83       try {
84           // Validar los datos recibidos
85           $validationErrors = $this->validateRegistrationData($input);
86           if (!empty($validationErrors)) {
87               return $this->response(400, "Errores de validación", ['errors' => $validationErrors]);
88           }
89
90           // Crear un nuevo usuario en la base de datos
91           $user = $this->userModel->create(
92               $input['email'],
93               $input['fullName'],
94               $input['password'],
95               $input['notificationType'],
96               $input['contactValue']
97           );
98
99           return $this->response(201, "Usuario registrado correctamente", $user);
100
101       } catch (Exception $e) {
102           return $this->response(500, $e->getMessage());
103       }
104   }
105
106   // Método para enviar una respuesta HTTP
107   private function response($code, $message, $data = []) {
108       http_response_code($code); // Establecer el código de respuesta HTTP
109       echo json_encode(array_merge([
110           "success" => $code < 400,
111           "message" => $message
112       ], $data ? ["user" => $data] : []));
113   }

```

No controlador de autenticación tamén temos a xestión de registro.

Temos para a validación dos datos a función de `validateRegistrationData`, na que se está a validar os datos para o rexistro.

Esta validando a función tanto o email, como que o nome do usuario polo menos teña 3 caracteres, que a contraseña teña o número mínimo de caracteres, e tamén valida o tipo de notificación, e que o valor do contacto non esté vacío.

Unha vez que a verificación foi correcta, créase o usuario na base de datos.

E despois mandárase unha resposta ao front.

```

User.php
PHP > models > User.php > User > _construct
1 <?php
2 class User {
3     private $conn;
4     private $table = 'Users';
5     public function __construct($db) {
6         $this->conn = $db;
7     }
8
9
10    public function findByEmail($email) {
11        $stmt = $this->conn->prepare("SELECT UserId, Password, FullName, Email, Role FROM Users WHERE Email = ?");
12        if (!$stmt) return false;
13
14        $stmt->bind_param("s", $email);
15        if (!$stmt->execute()) return false;
16
17        $result = $stmt->get_result();
18        return $result->fetch_assoc();
19    }
20    public function create($email, $fullName, $password, $notificationType, $contactValue) {
21        try {
22            $this->conn->begin_transaction();
23
24            // Verificar si el email existe
25            if ($this->emailExists($email)) {
26                throw new Exception("El email ya está registrado");
27            }
28
29            // Hash de la contraseña
30            $passwordHash = password_hash($password, PASSWORD_DEFAULT);
31
32            // Insertar usuario
33            $stmt = $this->conn->prepare(
34                "INSERT INTO {$this->table} (Email, FullName, Password) VALUES (?, ?, ?)"
35            );
36            $stmt->bind_param("sss", $email, $fullName, $passwordHash);
37

```

En este modelo, o que temos definidos son os métodos para o acceso a base de datos

Temos as seguintes funcións, a primeira que temos definida e para buscar un usuario mediante o email, en caso de que o usuario non esté na base de datos, devolverá "false".

```

User.php
PHP > models > User.php > User > create
2 class User {
10    public function findByEmail($email) {
11        ...
12        return $result->fetch_assoc();
13    }
14
15    public function create($email, $fullName, $password, $notificationType, $contactValue) {
16        try {
17            $this->conn->begin_transaction();
18
19            // Verificar si el email existe
20            if ($this->emailExists($email)) {
21                throw new Exception("El email ya está registrado");
22            }
23
24            // Hash de la contraseña
25            $passwordHash = password_hash($password, PASSWORD_DEFAULT);
26
27            // Insertar usuario
28            $stmt = $this->conn->prepare(
29                "INSERT INTO {$this->table} (Email, FullName, Password) VALUES (?, ?, ?)"
30            );
31            $stmt->bind_param("sss", $email, $fullName, $passwordHash);
32
33            if (!$stmt->execute()) {
34                throw new Exception("Error al registrar el usuario");
35            }
36
37            $userId = $stmt->insert_id;
38
39            // Insertar preferencia de notificación
40            $stmtPref = $this->conn->prepare(
41                "INSERT INTO NotificationPreferences (UserID, NotificationType, ContactValue, IsActive)
42                VALUES (?, ?, ?, true)"
43            );
44            $stmtPref->bind_param("iss", $userId, $notificationType, $contactValue);
45
46            if (!$stmtPref->execute()) {
47                ...
48            }
49

```

Aqui temos o código para a inserción do usuario dentro da base de datos, temos tamen a verificación de si o usuario existe dentro da base de datos, para que non se poda volver a insertar, tamen temos o de password\_hash(), que o que fai e que a contraseña como e un campo seguro, lle faga un hash: e un algoritmo matemático, que convirte a o texto de entrada nunha cadea alfanumérica de lonxitude fixa.

```
register.php x
PHP > routes > register.php > ...
1 <?php
2 require_once __DIR__ . '/../config/cors.php';
3 require_once __DIR__ . '/../controllers/AuthController.php';
4
5 configureCors();
6
7 $controller = new AuthController();
8
9 if ($_SERVER['REQUEST_METHOD'] === 'POST') {
10     $input = json_decode(file_get_contents("php://input"), true);
11     $controller->register($input);
12 } else {
13     http_response_code(405);
14     echo json_encode(["success" => false, "message" => "Método no permitido"]);
15 }
```

No fichero da ruta de rexistro, o que temos e que simplemente a configuración de CORS, e a verificación de que o método co que se está facendo a petición e correcta, e despois de esa verificación o que se fai e rexistrar o usuario.

```
login.php x
PHP > routes > login.php > ...
1 <?php
2 require_once __DIR__ . '/../config/cors.php';
3 require_once __DIR__ . '/../controllers/AuthController.php';
4
5 configureCors();
6
7 $controller = new AuthController();
8
9 if ($_SERVER['REQUEST_METHOD'] === 'POST') {
10     $input = json_decode(file_get_contents("php://input"), true);
11     $controller->login($input);
12
13     // Establecer una cookie para recordar el rol del usuario
14     if (isset($_SESSION['user']['role'])) {
15         setcookie('user_role', $_SESSION['user']['role'], time() + (86400 * 30), "/"); // Expira en 30 días
16     }
17 } else {
18     http_response_code(405);
19     echo json_encode(["success" => false, "message" => "Método no permitido"]);
20 }
21
```

Na ruta de login, o que esta a verificarse e que o método co que se esta facendo a petición e correcta, e tamén se está a realizar o login propiamente dito.

Tamén esta a realizarse o establecemento de unha cookie, para almacenar o rol do usuario que se está a loguear.

En este caso non se devolve a resposta de se o inicio de sesion foi correcto dende aquí, xa que podería fallar por varias cousas, polo que se esta a facer dende o controlador.

Appointment.php X

PHP > models > Appointment.php > AppointmentModel

```
1  <?php
2  class AppointmentModel {
3      private $conn;
4      private $table = 'Appointments';
5
6      // Propiedades de la cita
7      public $AppointmentID;
8      public $UserID;
9      public $VehicleID;
10     public $WorkshopID;
11     public $StartDateTime;
12     public $EndDateTime;
13     public $Descripcion;
14     public $Status;
15
16     // Constructor con conexión a la base de datos
17     public function __construct($db) {
18         $this->conn = $db;
19     }
20
21     // Método para obtener el taller asociado a un usuario
22     public function getWorkshopByUserId($userId) {
23         $query = "SELECT WorkshopID FROM Workshops WHERE UserID = ?";
24
25         $stmt = $this->conn->prepare($query);
26         $stmt->bind_param("i", $userId);
27         $stmt->execute();
28
29         return $stmt->get_result();
30     }
31
32     // Método para obtener slots semanales
33     public function getWeeklySlots($workshopId, $startDate, $endDate) {
34         $query = "SELECT StartDateTime
35                 FROM " . $this->table . "
36                 WHERE WorkshopID = ?
37                 AND DATE(StartDateTime) BETWEEN ? AND ?";
```

```
46
47     // Método para obtener slots diarios
48     public function getDaySlots($workshopId, $date) {
49         $query = "SELECT StartDateTime
50                 FROM " . $this->table . "
51                 WHERE WorkshopID = ?
52                 AND DATE(StartDateTime) = ?";
53
54         $stmt = $this->conn->prepare($query);
55         $stmt->bind_param("is", $workshopId, $date);
56         $stmt->execute();
57
58         return $stmt->get_result();
59     }
60
```

```
61     // Método para verificar si existe un taller
62     public function checkWorkshopExists($workshopId) {
63         $query = "SELECT WorkshopID FROM Workshops WHERE WorkshopID = ?";
64
65         $stmt = $this->conn->prepare($query);
66         $stmt->bind_param("i", $workshopId);
67         $stmt->execute();
68
69         return $stmt->get_result()->num_rows > 0;
70     }

```

```
32     // Método para obtener slots semanales
33     public function getWeeklySlots($workshopId, $startDate, $endDate) {
34         $query = "SELECT StartDateTime
35                 FROM " . $this->table . "
36                 WHERE WorkshopID = ?
37                 AND DATE(StartDateTime) BETWEEN ? AND ?
38                 AND DAYOFWEEK(StartDateTime) BETWEEN 2 AND 6";
39
40         $stmt = $this->conn->prepare($query);
41         $stmt->bind_param("iss", $workshopId, $startDate, $endDate);
42         $stmt->execute();
43
44         return $stmt->get_result();
45     }
46
```

```

72 // Método para crear una nueva cita
73 public function create() {
74     $query = "INSERT INTO " . $this->table . "
75         (UserID, VehicleID, WorkshopID, StartDateTime, EndDateTime, Description, Status)
76         VALUES (?, ?, ?, ?, ?, ?, ?)";
77
78     $stmt = $this->conn->prepare($query);
79
80     // Limpiar y sanitizar los datos
81     $this->Descripcion = htmlspecialchars(strip_tags($this->Descripcion));
82     $this->Status = htmlspecialchars(strip_tags($this->Status));
83
84     // Vincular los parámetros
85     $stmt->bind_param("iiissss",
86         $this->UserID,
87         $this->VehicleID,
88         $this->WorkshopID,
89         $this->StartDateTime,
90         $this->EndDateTime,
91         $this->Descripcion,
92         $this->Status
93     );
94
95     return $stmt->execute();
96 }
97

```

```

98 // Método para obtener todas las citas de un taller
99 public function getWorkshopAppointments($workshopId) {
100     $query = "SELECT
101         a.AppointmentID,
102         a.StartDateTime,
103         a.EndDateTime,
104         CONCAT(v.Marca, ' ', v.Modelo) AS Vehiculo,
105         a.Description,
106         a.Status,
107         u.FullName AS UserName
108     FROM " . $this->table . " a
109     JOIN Vehicles v ON a.VehicleID = v.VehicleID
110     JOIN Users u ON v.UserID = u.UserID
111     WHERE a.WorkshopID = ?";
112
113     $stmt = $this->conn->prepare($query);
114     $stmt->bind_param("i", $workshopId);
115     $stmt->execute();
116
117     return $stmt->get_result();
118 }
119

```

```

120 // Método para verificar disponibilidad de slot
121 public function checkSlotAvailability($workshopId, $startDateTime) {
122     $query = "SELECT AppointmentID
123     FROM " . $this->table . "
124     WHERE WorkshopID = ?
125     AND StartDateTime = ?";
126
127     $stmt = $this->conn->prepare($query);
128     $stmt->bind_param("is", $workshopId, $startDateTime);
129     $stmt->execute();
130
131     return $stmt->get_result()->num_rows === 0;
132 }
133
134 // Método para obtener el nombre del taller
135 public function getWorkshopName($workshopId) {
136     $stmt = $this->conn->prepare("SELECT Name FROM Workshops WHERE WorkshopID = ?");
137     $stmt->bind_param("i", $workshopId);
138     $stmt->execute();
139     $result = $stmt->get_result();
140
141     if ($row = $result->fetch_assoc()) {
142         return $row['Name'];
143     }
144     return "Taller no encontrado";
145 }
146

```

A continuación un resumen dos métodos que hai:

1. **getWorkshopByUserId(\$userId)**  
Obtén o identificador (WorkshopID) do taller asociado a un usuario específico (UserID).
2. **getWeeklySlots(\$workshopId, \$startDate, \$endDate)**  
Recupera os slots dispoñibles dun taller (WorkshopID) durante unha semana específica, considerando só os días laborais (de luns a venres).
3. **getDaySlots(\$workshopId, \$date)**  
Obtén os slots dispoñibles dun taller (WorkshopID) nun día específico (date).
4. **checkWorkshopExists(\$workshopId)**  
Verifica se un taller existe na base de datos, devolvendo true ou false.
5. **create()**  
Crea unha nova cita na táboa Appointments coa información proporcionada (usuario, vehículo, taller, datas, descrición e estado).
6. **getWorkshopAppointments(\$workshopId)**  
Obtén todas as citas dun taller (WorkshopID), incluíndo detalles como vehículo, usuario, datas e estado.
7. **checkSlotAvailability(\$workshopId, \$startDateTime)**  
Verifica se un slot específico (StartDateTime) está dispoñible nun taller (WorkshopID).
8. **getWorkshopName(\$workshopId)**  
Obtén o nome dun taller (WorkshopID). Se non existe, devolve "Taller no encontrado".

A continuación verase o controlador

```
appointments.php AppointmentController.php X
PHP > controllers > AppointmentController.php > AppointmentController > consultarSemana
1  <?php
2  require_once __DIR__ . '/../models/Appointment.php';
3
4  class AppointmentController {
5      private $model;
6      private $userId;
7      private $userRole;
8
9      public function __construct($db, $userId, $userRole) {
10         $this->model = new AppointmentModel($db);
11         $this->userId = $userId;
12         $this->userRole = $userRole;
13     }
14
15     public function consultarSemana($data) {
16         try {
17             if (!isset($data['WorkshopID'])) {
18                 return $this->sendResponse(400, false, "Falta el ID del taller");
19             }
20
21             $workshopId = $data['WorkshopID'];
22
23             // Verificar que el taller existe
24             if (!$this->model->checkWorkshopExists($workshopId)) {
25                 return $this->sendResponse(404, false, "Taller no encontrado");
26             }
27
28             $fechaActual = new DateTime();
29             $fechaInicio = $fechaActual->format('Y-m-d');
30             $fechaFin = (clone $fechaActual)->modify('+30 days')->format('Y-m-d');
31
32             $result = $this->model->getWeeklySlots($workshopId, $fechaInicio, $fechaFin);
33             $slots = $this->procesarSlotsSemanales($result, $fechaInicio, $fechaFin);
34
35             return $this->sendResponse(200, true, "", ["slotsSemana" => $slots]);
36         } catch (Exception $e) {
37             return $this->sendResponse(500, false, "Error al consultar disponibilidad: " . $e->getMessage());
38         }
39     }
40 }
```

Temos unha función para a xestión de consultar as citas que están tanto dispoñibles como non, e temos tamén a comprobación de si o taller existe, este metodo retorna un mes, tanto de citas dispoñibles como non dispoñibles, devolve os 30 días seguintes incluíndo o día actual.

```

public function crear($data) {
    try {
        // Validar campos requeridos
        $camposRequeridos = ['Fecha', 'Hora', 'VehicleID', 'WorkshopID'];
        foreach ($camposRequeridos as $campo) {
            if (!isset($data[$campo])) {
                return $this->sendResponse(400, false, "Falta el campo: $campo");
            }
        }

        // Establecer descripción por defecto si no se proporciona
        $data['Descripcion'] = $data['Descripcion'] ?? '';

        // Validar fecha y hora
        if (!$this->validarFechaHora($data['Fecha'], $data['Hora'])) {
            return $this->sendResponse(400, false, "Formato de fecha u hora inválido");
        }

        $startDateTime = $data['Fecha'] . ' ' . $data['Hora'] . ':00';
        $endDateTime = date('Y-m-d H:i:s', strtotime($startDateTime . ' +1 hour'));

        // Verificar disponibilidad
        if (!$this->model->checkSlotAvailability($data['WorkshopID'], $startDateTime)) {
            return $this->sendResponse(409, false, "Horario no disponible");
        }

        // Crear la cita
        $this->model->UserID = $this->userId;
        $this->model->VehicleID = $data['VehicleID'];
        $this->model->WorkshopID = $data['WorkshopID'];
        $this->model->StartDateTime = $startDateTime;
        $this->model->EndDateTime = $endDateTime;
        $this->model->Descripcion = $data['Descripcion'];
        $this->model->Status = $data['Status'] ?? 'Pendiente';

        if ($this->model->create()) {
            return $this->sendResponse(200, true, "Cita creada correctamente");
        } else {
            return $this->sendResponse(500, false, "Error al crear la cita");
        }
    } catch (Exception $e) {
        return $this->sendResponse(500, false, "Error: " . $e->getMessage());
    }
}

```

Temos o método para poder crear unha cita, que valida que todos os campos imprescindibles están cubertos, a descripción en caso de que a descripción non estea correctamente definida, está posto para que o estabrezca en branco.

Tamén se valida a data e a hora da cita, validase concretamente o formato, despois de esto, farase a comprobación de si o slot que se esta a solicitar está vacío ou xa ten outro usuario establecido.

A continuación de esto, o que se fai e crear a cita como tal, establecéndoa na base de datos.

Por último o que se fai e o envío de se se creou a cita correctamente ou no ao front.



```

public function verCitasTaller() {
    try {
        if ($this->userRole !== 'Taller') {
            return $this->sendResponse(403, false, "No autorizado");
        }

        $workshop = $this->model->getWorkshopByUserId($this->userId)->fetch_assoc();
        if (!$workshop) {
            return $this->sendResponse(404, false, "No se encontró el taller asociado");
        }

        $result = $this->model->getWorkshopAppointments($workshop['WorkshopID']);
        $citas = [];
        while ($row = $result->fetch_assoc()) {
            $citas[] = $row;
        }

        return $this->sendResponse(200, true, "", ["citas" => $citas]);
    } catch (Exception $e) {
        return $this->sendResponse(500, false, "Error: " . $e->getMessage());
    }
}

```

Esto o que fai e comprobar primeiramente, comprobar que o usuario autenticado e un taller, Despois estase a recuperar o taller asociado ao id do usuario.

Cando esto esta todo correcto, o que fai e que se recupera da base de datos as citas de este usuario de taller.

Por último o que se fai e o envío de datos.

Ruta de appointments:

```

<?php
require_once __DIR__ . '/../config/cors.php';
require_once __DIR__ . '/../controllers/AppointmentController.php';
require_once __DIR__ . '/../config/database.php';

configureCors();

session_start();

// Verificar autenticación
if (!isset($_SESSION['user']['id']) || !isset($_SESSION['user']['role'])) {
    http_response_code(401);
    echo json_encode(['success' => false, 'message' => 'No autorizado']);
    exit();
}

// Inicializar la base de datos y el controlador
$db = new Database();
$conn = $db->getConnection();

if (!$conn) {
    http_response_code(500);
    echo json_encode(['success' => false, 'message' => 'Error de conexión a la base de datos']);
    exit();
}

```

O que se fai e como en todas as rutas de PHP, o que se fai e configurar as cabeceiras de CORS

Despois iniciase a sesión, e faise a verificación de que o usuario ten tanto o id de usuario como o rol, están establecidos.

En caso de que a conexión a base de datos falle por calquera motivo, mandaremos unha resposta de error na conexión.

```
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $input = json_decode(file_get_contents("php://input"), true);

    if (!isset($input['accion'])) {
        throw new Exception("Falta el parámetro 'accion'");
    }

    switch ($input['accion']) {
        case 'consultar_semana':
            $controller->consultarSemana($input);
            break;
        case 'crear':
            $controller->crear($input);
            break;
        case 'ver_citas_taller':
            $controller->verCitasTaller();
            break;
        default:
            throw new Exception("Acción no válida");
    }
} else {
    throw new Exception("Método no permitido");
}
} catch (Exception $e) {
    http_response_code(400);
    echo json_encode([
        'success' => false,
        'message' => $e->getMessage()
    ]);
}
```

Aqui o que faremos e en función da acción que queremos realizar, chamaremos aos métodos pertinentes.

En caso de que a acción que estemos a mandar, non estea contemplada no switch, o que se fará e devolver unha excepción.

En caso de que se esté facendo a petición con un método que non está permitido, devolveremos unha excepción.

Para o controlador do chat temos o seguinte

```

<?php
require_once __DIR__ . '/../models/Chat.php';

class ChatController {
    private $model;
    private $userId;
    private $userRole;

    public function __construct($db, $userId, $userRole) {
        $this->model = new ChatModel($db);
        $this->userId = $userId;
        $this->userRole = $userRole;
    }

    public function iniciarChat($data) {
        try {
            if (!isset($data['workshop_id'])) {
                return $this->sendResponse(400, false, "Falta el ID del taller");
            }

            // Verificar si ya existe un chat activo
            $existingChatId = $this->model->checkExistingChat($this->userId, $data['workshop_id']);
            if ($existingChatId) {
                return $this->sendResponse(200, true, "Chat existente recuperado",
                    ['chat_id' => $existingChatId]);
            }

            $chatId = $this->model->createChat($this->userId, $data['workshop_id']);
            if ($chatId) {
                // Enviar mensaje automático de bienvenida
                $this->model->sendMessage($chatId, $this->userId, "¡Hola! He iniciado un nuevo chat.");
                return $this->sendResponse(200, true, "Chat iniciado correctamente",
                    ['chat_id' => $chatId]);
            }

            return $this->sendResponse(500, false, "Error al iniciar el chat");
        } catch (Exception $e) {
            return $this->sendResponse(500, false, $e->getMessage());
        }
    }
}

```

Primeiro o que facemos e comprobar que todos os campos necesarios para poder crear un chat, estan establecidos.

Despois o que facemos e realizar a comprobación de si un chat existente existe, en caso de que esto exista, o que facemos e recuperar o chat existente.

En caso de que non exista un chat existente, o que facemos e crear un novo chat, e mandaremos un mensaxe para ver que todo foi correcto.

En caso de que algo de esto falle, devolveremos un error

```

public function enviarMensaje($data) {
    try {
        if (!isset($data['chat_id']) || !isset($data['message'])) {
            return $this->sendResponse(400, false, "Faltan datos requeridos");
        }

        if ($this->model->sendMessage($data['chat_id'], $this->userId, $data['message'])) {
            return $this->sendResponse(200, true, "Mensaje enviado correctamente");
        }

        return $this->sendResponse(500, false, "Error al enviar el mensaje");
    } catch (Exception $e) {
        return $this->sendResponse(500, false, $e->getMessage());
    }
}

```

En este método o que estamos a facer e comprobar primeiramente que os datos requeridos están establecidos.

Despois intentamos mandar o mensaxe como tal, e se este foi correcto, mandaremos unha resposta de que todo foi correcto.

Se algo falla, o que faremos e mandar unha resposta de que houbo un error

```

public function obtenerChats() {
    try {
        $chats = $this->userRole === 'Taller' ?
            $this->model->getWorkshopChats($this->getWorkshopId()) :
            $this->model->getUserChats($this->userId);

        $result = [];
        while ($chat = $chats->fetch_assoc()) {
            $chat['unreadCount'] = $this->model->getUnreadCount($chat['ChatID'], $this->userId);
            $result[] = $chat;
        }

        return $this->sendResponse(200, true, "", ['chats' => $result]);
    } catch (Exception $e) {
        return $this->sendResponse(500, false, $e->getMessage());
    }
}

public function obtenerMensajes($chatId) {
    try {
        $messages = $this->model->getMessages($chatId);
        $this->model->markAsRead($chatId, $this->userId);

        return $this->sendResponse(200, true, "",
            ['messages' => $messages->fetch_all(MYSQLI_ASSOC)]);
    } catch (Exception $e) {
        return $this->sendResponse(500, false, $e->getMessage());
    }
}

```

O que se fai e primeiro comprobar que rol ten o usuario, xa que en función de eso, devolveremos uns chats ou outros.

Despois mandaremos os chats todos na resposta.

Temos despois para obter os mensaxes asociados a un chat específico e marcaos como lidos.

```

private function getWorkshopId() {
    $query = "SELECT WorkshopID FROM Workshops WHERE UserID = ?";
    $stmt = $this->model->getConnection()->prepare($query);
    $stmt->bind_param("i", $this->userId);
    $stmt->execute();
    $result = $stmt->get_result();
    return $result->fetch_assoc()['WorkshopID'];
}

private function sendResponse($code, $success, $message, $data = []) {
    http_response_code($code);
    echo json_encode(array_merge(
        ["success" => $success, "message" => $message],
        $data
    ));
    return true;
}

```

Aqui temos 2 funcións:

O método `getWorkshopId` ten como propósito obter o identificador (WorkshopID) do taller asociado ao usuario actual (`$this->userId`).

A outra función o que queda, o que fai e mandar unha resposta HTTP con un código, e datos adicionais.

```

InvoiceController.php X
PHP > controllers > InvoiceController.php > InvoiceController > crear
1 <?php
2 require_once __DIR__ . '/../models/Invoice.php';
3
4 class InvoiceController {
5     private $model;
6     private $userId;
7     private $userRole;
8
9     public function __construct($db, $userId, $userRole) {
10         $this->model = new InvoiceModel($db);
11         $this->userId = $userId;
12         $this->userRole = $userRole;
13     }
14
15     public function crear($data) {
16         try {
17             if ($this->userRole != 'Taller') {
18                 return $this->sendResponse(403, false,
19                     "No tienes permisos para crear facturas");
20             }
21
22             if (!$this->validarDatosFactura($data)) {
23                 return $this->sendResponse(400, false,
24                     "Faltan datos requeridos o son inválidos");
25             }
26
27             // Actualizar los nombres de los campos según la base de datos
28             $this->model->AppointmentID = $data['appointment_id'];
29             $this->model->date = $data['date']; date('Y-m-d');
30             $this->model->Estado = $data['estado']; 'Pendiente';
31             $this->model->UserID = $this->userId;
32             $this->model->Items = array_map(function($item) {
33                 // Asegurar que todos los campos necesarios existen
34                 if (!isset($item['description'])) {
35                     throw new Exception("Faltan campos requeridos en los items de la factura");
36                 }
37                 if (!isset($item['quantity'])) {
38                     throw new Exception("Faltan campos requeridos en los items de la factura");
39                 }
40                 if (!isset($item['unit_price'])) {
41                     throw new Exception("Faltan campos requeridos en los items de la factura");
42                 }
43                 if (!isset($item['tax_rate'])) {
44                     throw new Exception("Faltan campos requeridos en los items de la factura");
45                 }
46                 // Calcular el amount aquí en lugar de esperar que venga en los datos
47                 $amount = floatval($item['quantity']) * floatval($item['unit_price']) *
48                     (1 + floatval($item['tax_rate']) / 100);
49             }, $data['items']);
50             $this->model->TotalAmount = $this->calcularTotal($data['items']);
51
52             if ($this->model->create()) {
53                 return $this->sendResponse(200, true,
54                     "Factura creada correctamente",
55                     ['invoice_id' => $this->model->invoiceID]);
56             }
57
58         } catch (Exception $e) {
59             return $this->sendResponse(500, false, $e->getMessage());
60         }
61     }
62 }

```

Aqui temos o método de crear unha factura, que primeiro e verificar que o rol do usuario e o correcto, despois verificamos que os datos son correctos, e os asignamos.

Calculamos o total e creamos a factura

Despois temos o envío de que todo foi correcto, en caso de que unha excepción ocorrese, devolvería un código 500 co detalle do erro.

## Código da parte de Angular

No componente de estadísticas, o que temos primeiramente a importación de todas as cousas necesarias, despois temos a declaración do componente, e despois todos os métodos declarados

```
TS statistics-viewer.component.ts X
AUTOCAREHUB > src > app > components > statistics-viewer > TS statistics-viewer.component.ts > StatisticsViewerComponent > constructor
1 import { Component, OnInit } from '@angular/core';
2 import { CommonModule } from '@angular/common';
3 import { FormsModule } from '@angular/forms';
4 import {
5   DataAccessService,
6   InvoiceStats,
7   Invoice,
8 } from '../services/dataAccess.service';
9 import { MenuComponent } from '../menu/menu.component';
10
11 @Component({
12   selector: 'app-statistics-viewer',
13   standalone: true,
14   imports: [CommonModule, FormsModule, MenuComponent],
15   templateUrl: './statistics-viewer.component.html',
16   styleUrls: ['./statistics-viewer.component.css'],
17 })
18 export class StatisticsViewerComponent implements OnInit {
19   startDate: string = '';
20   endDate: string = '';
21   stats: InvoiceStats | null = null;
22   invoices: Invoice[] = [];
23   loading: boolean = false;
24   error: string = '';
25
26   constructor(private dataAccess: DataAccessService) {
27     // Inicializar fechas por defecto (último mes)
28     const today = new Date();
29     this.endDate = today.toISOString().split('T')[0];
30     today.setMonth(today.getMonth() - 1);
31     this.startDate = today.toISOString().split('T')[0];
32   }
33
34   ngOnInit() {
35     this.loadData();
36   }
37
38   loadData() {
39     this.loading = true;
40     this.error = '';
41
42     // Cargar estadísticas
43     this.dataAccess
44       .obtenerEstadisticasFacturacion(this.startDate, this.endDate)
45       .subscribe({
46         next: (stats) => {
47           this.stats = stats;
48           this.loading = false;
49         },
50         error: (error) => {
51           this.error = 'Error al cargar estadísticas';
52           this.loading = false;
53         },
54       });
55
56     // Cargar facturas del periodo
57     this.dataAccess
58       .buscarFacturas({
59         startDate: this.startDate,
60         endDate: this.endDate,
61       })
62       .subscribe({
63         next: (response) => {
64
65           // Cargar facturas del periodo
66           this.dataAccess
67             .buscarFacturas({
68               startDate: this.startDate,
69               endDate: this.endDate,
70             })
71             .subscribe({
72               next: (response) => {
73                 this.invoices = response.invoices;
74               },
75               error: (error) => {
76                 this.error = 'Error al cargar facturas';
77               },
78             });
79         },
80       });
81
82   }
83
84   onChangeDate() {
85     this.loadData();
86   }
87
88   formatCurrency(amount: number): string {
89     return new Intl.NumberFormat('es-ES', {
90       style: 'currency',
91       currency: 'EUR',
92     }).format(amount);
93   }
94
95   formatDate(date: string): string {
96     return new Date(date).toLocaleDateString('es-ES');
97   }
98
99   getPendingPercentage(): number {
100     if (!this.stats) return 0;
101     return (this.stats.pendientes / this.stats.total_facturas) * 100;
102   }
103 }
```

Aquí o que vemos, e a recuperación de varios datos, como pode ser tanto as estatísticas de facturación, e tamén o que facemos e obter as facturas en función de unhas fechas que estamos a establecer no compoñente de angular.

Tamén temos unha función para formatear os datos numéricos, e para formatear as datas.

A última función o que fai e que calcula o porcentaxe de facturas pendentes que ten o usuario.

```
statistics-viewer.component.html X
AUTOCAREHUB > src > app > components > statistics-viewer > statistics-viewer.component.html > div.statistics-container > div.loading
Go to component
1 <app-menu></app-menu>
2
3 <div class="statistics-container">
4   <div class="date-selector">
5     <label>
6       Desde:
7       <input type="date" [(ngModel)]="startDate" (change)="onDateChange()">
8     </label>
9     <label>
10      Hasta:
11      <input type="date" [(ngModel)]="endDate" (change)="onDateChange()">
12    </label>
13  </div>
14  <div *ngIf="error" class="error-message">
15    {{ error }}
16  </div>
17  <div *ngIf="loading" class="loading">
18    Cargando estadísticas...
19  </div>
20
21  <div *ngIf="stats && !loading" class="stats-grid">
22    <div class="stat-card">
23      <h3>Total Facturas</h3>
24      <p>{{ stats.total_facturas }}</p>
25    </div>
26    <div class="stat-card">
27      <h3>Total Facturado</h3>
28      <p>{{ formatCurrency(stats.total_facturado) }}</p>
29    </div>
30    <div class="stat-card">
31      <h3>Promedio por Factura</h3>
32      <p>{{ formatCurrency(stats.promedio_factura) }}</p>
33    </div>
34    <div class="stat-card">
35      <h3>Estado Facturas</h3>
36      <p>Pendientes: {{ stats.pendientes }}</p>
37      <p>Pagadas: {{ stats.pagadas }}</p>
38      <div class="progress-bar">
39        <div class="progress" [style.width.%]="getPendingPercentage()"></div>
40      </div>
41    </div>
42  </div>
43
44  <div *ngIf="invoices.length > 0" class="invoices-table">
45    <h3>Facturas del Período</h3>
46    <table>
47      <thead>
48        <tr>
49          <th>Fecha</th>
50          <th>Importe</th>
51          <th>Estado</th>
52          <th>Cliente/Taller</th>
53        </tr>
54      </thead>
55      <tbody>
56        <tr *ngFor="let invoice of invoices">
57          <td>{{ formatDate(invoice.Date) }}</td>
58          <td>{{ formatCurrency(invoice.TotalAmount) }}</td>
59          <td>{{ invoice.Estado }}</td>
60          <td>{{ invoice.UserName || invoice.WorkshopName }}</td>
61        </tr>
62      </tbody>
63    </table>
64  </div>
65</div>
```

Este e o html asociado ao compoñente asociado, no que podemos ver os datos.

Temos tamén un compoñente para poder dar de alta uns talleres

```
TS workshops-management.component.ts X
AUTOCAREHUB > src > app > components > workshops-management > TS workshops-management.component.ts > WorkshopsManagementComponent > newWorkshop > Email
1 import { Component, OnInit } from '@angular/core';
2 import { FormsModule } from '@angular/forms';
3 import { CommonModule } from '@angular/common';
4 import { DataAccessService, Workshop } from '../../services/dataAccess.service';
5
6 @Component({
7   selector: 'app-workshops-management',
8   standalone: true,
9   imports: [CommonModule, FormsModule],
10  templateUrl: './workshops-management.component.html',
11  styleUrls: ['./workshops-management.component.css'],
12 })
13 export class WorkshopsManagementComponent implements OnInit {
14   workshops: Workshop[] = [];
15   newWorkshop: any = {
16     workshopName: '',
17     address: '',
18     phone: '',
19     Email: '',
20     fullName: '',
21     description: '',
22     password: '',
23   };
24   errorMessage: string = '';
25   successMessage: string = '';
26
27   constructor(private dataAccess: DataAccessService) {}
28
29   ngOnInit(): void {
30     this.loadWorkshops();
31   }
32
33   loadWorkshops(): void {
34     this.dataAccess.obtenerTalleres().subscribe({
35       next: (response) => {
36         if (response.success) {
37           this.workshops = response.workshops;
38           console.log(this.workshops);
39         } else {
40           this.errorMessage =
41             response.message || 'Error al cargar los talleres';
42         }
43       },
44       error: () => {
45         this.errorMessage = 'Error al cargar los talleres';
46       },
47     });
48   }
49
50   createWorkshop(): void {
51     if (
52       !this.newWorkshop.workshopName ||
53       !this.newWorkshop.address ||
54       !this.newWorkshop.phone ||
55       !this.newWorkshop.email ||
56       !this.newWorkshop.fullName ||
57       !this.newWorkshop.password
58     ) {
59       this.errorMessage = 'Por favor, completa todos los campos requeridos';
60       return;
61     }
62
63     this.dataAccess.createWorkshop(this.newWorkshop).subscribe({
64       next: (response) => {
65         if (response.success) {
66           this.successMessage = 'Taller creado exitosamente';
67           this.loadWorkshops();
68           this.resetForm();
69         } else {
70           this.errorMessage = response.message || 'Error al crear el taller';
71         }
72       },
73       error: () => {
74         this.errorMessage = 'Error al crear el taller';
75       },
76     });
77   }
78
79   private resetForm(): void {
80     this.newWorkshop = {
81       workshopName: '',
82       address: '',
83       phone: '',
84       email: '',
85       fullName: '',
86       description: '',
87       password: '',
88     };
89   }
90 }
```

```
81     this.dataAccess.createWorkshop(this.newWorkshop).subscribe({
82       next: (response) => {
83         if (response.success) {
84           this.successMessage = 'Taller creado exitosamente';
85           this.loadWorkshops();
86           this.resetForm();
87         } else {
88           this.errorMessage = response.message || 'Error al crear el taller';
89         }
90       },
91       error: () => {
92         this.errorMessage = 'Error al crear el taller';
93       },
94     });
95   }
96
97   private resetForm(): void {
98     this.newWorkshop = {
99       workshopName: '',
100      address: '',
101      phone: '',
102      email: '',
103      fullName: '',
104      description: '',
105      password: '',
106    };
107  }
108 }
```



Esto e un componente para poder dar de alta un novo taller, que esto solo o poderá facer un usuario con rol administrador

```
do to component
<div class="container">
  <h2>Gestión de Talleres</h2>

  <!-- Mensajes de error y éxito -->
  <div *ngIf="errorMessage" class="alert alert-danger">
    {{ errorMessage }}
  </div>
  <div *ngIf="successMessage" class="alert alert-success">
    {{ successMessage }}
  </div>

  <!-- Formulario de creación -->
  <div class="workshop-form">
    <h3>Crear Nuevo Taller</h3>
    <div class="form-group">
      <label>Nombre</label>
      <input
        type="text"
        class="form-control"
        [(ngModel)]="newWorkshop.workshopName"
      />
    </div>

    <div class="form-group">
      <label>Dirección</label>
      <input
        type="text"
        class="form-control"
        [(ngModel)]="newWorkshop.address"
      />
    </div>

    <div class="form-group">
      <label>Teléfono</label>
      <input type="tel" class="form-control" [(ngModel)]="newWorkshop.phone" />
    </div>

    <div class="form-group">
      <label>Correo Electrónico</label>
      <input
        type="email"
        class="form-control"
        [(ngModel)]="newWorkshop.email"
      />
    </div>

    <div class="form-group">
      <label>Nombre Completo</label>
      <input
        type="text"
        class="form-control"
        [(ngModel)]="newWorkshop.fullName"
      />
    </div>

    <div class="form-group">
      <label>Descripción</label>
      <textarea
        class="form-control"
        [(ngModel)]="newWorkshop.description"
      ></textarea>
    </div>
  </div>
</div>
```

Este e o html asociado ao TS, temos un formulario para dar de alta os talleres e tamen visualizar os que xa temos

Temos o compoñente tamén para poder pedir as citas

```
make-appointment.component.ts M X
AUTOCAREHUB > src > app > components > make-appointment > ts make-appointment.component.ts > MakeAppointmentComponent > ngOnInit
1 import { Component, OnInit } from '@angular/core';
2 import { DataAccessService } from '../../services/dataAccess.service';
3 import { CommonModule } from '@angular/common';
4 import { FormsModule, ReactiveFormsModule } from '@angular/forms';
5 import { MenuComponent } from '../../menu/menu.component';
6
7 interface WeekSlots {
8   [key: string]: {
9     hora: string;
10    estado: string;
11  };
12 }
13
14 @Component({
15   selector: 'app-make-appointment',
16   standalone: true,
17   imports: [CommonModule, FormsModule, ReactiveFormsModule, MenuComponent],
18   templateUrl: './make-appointment.component.html',
19   styleUrls: ['./make-appointment.component.css'],
20 })
21 export class MakeAppointmentComponent implements OnInit {
22   selectedLang: string = 'es';
23   appointmentForm: any;
24   errorMessage: string = '';
25   vehiclesErrorMessage: string = '';
26   monthSlots: WeekSlots = {};
27   loading: boolean = false;
28   loadingVehicles: boolean = false;
29   loadingWorkshops: boolean = false;
30   selectedSlots: { fecha: string; hora: string }[] = [];
31   selectedVehicle: number = 0;
32   selectedWorkshop: number = 0;
33   motivo: string = '';
34   vehicles: any[] = [];
35   workshops: any[] = [];
36   isPopupVisible = false;
37
38   // Pagination properties
39   visibleDates: string[] = [];
40   currentPage: number = 1;
41   datesPerPage: number = 5;
42   totalPages: number = 1;
43
44   constructor(private dataAccess: DataAccessService) {}
45
46   /**
47    * Inicializa el componente cargando los datos del mes actual, los vehículos y los talleres
48    */
49   ngOnInit(): void {
50     this.cargarVehiculos();
51     this.cargarTalleres();
52   }
53 }
```

```
57 cargarVehiculos() {
58   this.loadingVehicles = true;
59   this.vehiclesErrorMessage = '';
60
61   this.dataAccess.obtenerVehiculos().subscribe({
62     next: (response) => {
63       if (response && response.success) {
64         this.vehicles = response.vehicles || [];
65         if (this.vehicles.length > 0 && !this.selectedVehicle) {
66           this.selectedVehicle = this.vehicles[0].VehicleID;
67         }
68       } else {
69         this.vehiclesErrorMessage = 'No se pudieron cargar los vehículos';
70       }
71     },
72     error: (error) => {
73       this.vehiclesErrorMessage = 'Error de conexión al cargar vehículos';
74     },
75     complete: () => {
76       this.loadingVehicles = false;
77     },
78   });
79 }
80
81 /**
82  * Carga los talleres disponibles
83  */
84 cargarTalleres() {
85   this.loadingWorkshops = true;
86   this.dataAccess.obtenerTalleres().subscribe({
87     next: (response) => {
88       if (response && response.success) {
89         this.workshops = response.workshops || [];
90         if (this.workshops.length > 0 && !this.selectedWorkshop) {
91           this.selectedWorkshop = this.workshops[0].WorkshopID;
92           this.consultarMes(); // Ahora se llama aquí, tras seleccionar taller
93         }
94       } else {
95         this.errorMessage = 'No se pudieron cargar los talleres';
96       }
97     },
98     error: (error) => {
99       this.errorMessage = 'Error de conexión al cargar talleres';
100     },
101     complete: () => {
102       this.loadingWorkshops = false;
103     },
104   });
105 }
106 }
```

```

142  /**
143   * Configura la paginación inicial basada en las fechas disponibles
144   */
145  setupPagination() {
146    const allDates = Object.keys(this.monthSlots).sort();
147    this.totalPages = Math.ceil(allDates.length / this.datesPerPage);
148    this.goToPage(1);
149  }
150
151  /**
152   * Navega a una página específica de la paginación
153   * @param page Número de página a la que se desea navegar
154   */
155  goToPage(page: number) {
156    if (page < 1 || page > this.totalPages) return;
157
158    this.currentPage = page;
159    const allDates = Object.keys(this.monthSlots).sort();
160    const startIndex = (page - 1) * this.datesPerPage;
161    this.visibleDates = allDates.slice(
162      startIndex,
163      startIndex + this.datesPerPage
164    );
165  }
166
167  /**
168   * Navega a la página anterior
169   */
170  prevPage() {
171    this.goToPage(this.currentPage - 1);
172  }
173
174  /**
175   * Navega a la página siguiente
176   */
177  nextPage() {
178    this.goToPage(this.currentPage + 1);
179  }
180
181  /**
182   * Alterna la selección de un horario específico
183   * @param fecha Fecha del horario
184   * @param hora Hora del horario
185   */
186  toggleSlotSelection(fecha: string, hora: string) {
187    const index = this.selectedSlots.findIndex(
188      (slot) => slot.fecha === fecha && slot.hora === hora
189    );
190    if (index > -1) {
191      this.selectedSlots.splice(index, 1);
192    } else {
193      this.selectedSlots.push({ fecha, hora });
194    }
195  }

```

Aquí lo que tenemos es tanto a parte de cargar los datos nada más se acceda al componente, como todas las propiedades y funciones, tenemos las funciones para establecer la paginación para las citas, ya que no tiene mucho sentido, en cuanto a experiencia de usuario, que tenga que deslizar para poder acceder a la última cita, es mejor para el usuario tener la paginación establecida

## Este é o serviço para acesso a dados

```
TS dataAccess.services.ts X
AUTOCAREHUB > src > app > services > TS dataAccess.services.ts > *O Workshop
1 import { Injectable } from '@angular/core';
2 import { HttpClient, HttpHeaders } from '@angular/common/http';
3 import { Observable, throwError } from 'rxjs';
4 import { catchError, map } from 'rxjs/operators';
5
6 interface LoginResponse {
7   success: boolean;
8   message?: string;
9   user?: {
10     id: number;
11     name: string;
12     email: string;
13     role: string; // Añadir campo de rol para verificar si es Taller o Usuario
14   };
15 }
16
17 interface RegisterResponse {
18   success: boolean;
19   message?: string;
20   errors?: string[];
21   user?: {
22     id: number;
23     name: string;
24     email: string;
25   };
26 }
27
28 export interface InvoiceItem {
29   ItemID: number;
30   Description: string;
31   Quantity: number;
32   UnitPrice: number;
33   TaxRate: number;
34   Amount: number;
35 }
36
37 export interface Invoice {
38   InvoiceID: number;
39   AppointmentID: number;
40   Date: string;
41   TotalAmount: number;
42   Estado: string;
43   UserName?: string; // Solo disponible para modo taller
44   Marca?: string;
45   Modelo?: string;
46   Anyo: string;
47   items: InvoiceItem[];
48   WorkshopName?: string; // Solo disponible para modo usuario
49   WorkshopAddress?: string; // Solo disponible para modo usuario
50   WorkshopPhone?: string; // Solo disponible para modo usuario
51 }
52
53 export interface InvoiceStats {
54   total_facturas: number;
55   total_facturado: number;
56   promedio_factura: number;
57   pendientes: number;
58   pagadas: number;
59 }
60
61
62 export interface Chat {
63   ChatID: number;
64   UserID: number;
65   WorkshopID: number;
66   LastMessage: string;
67   Status: 'Active' | 'Archived';
68   CreateAt: string;
69   WorkshopName?: string;
70   UserName?: string;
71   unreadCount: number;
72 }
73
74 export interface Message {
75   MessageID: number;
76   ChatID: number;
77   SenderID: number;
78   Message: string;
79   IsRead: boolean;
80   CreateAt: string;
81   SenderName: string;
82 }
83
84 export interface Workshop {
85   WorkshopID?: number;
86   UserID?: number;
87   Name: string;
88   Address: string;
89   Phone: string;
90   Description?: string;
91   Email?: string;
92   Fullname?: string;
93 }
```

Aquí temos establecidos as interfaces da resposta dende os ficheiros de PHP.

```

115 });
116
117 constructor(private http: HttpClient) {}
118
119 /**
120  * Autentica al usuario mediante sus credenciales
121  * @param email - Correo electrónico del usuario
122  * @param password - Contraseña del usuario
123  * @returns Observable con la respuesta del login
124  */
125 checkUserAccount(email: string, password: string): Observable<LoginResponse> {
126     return this.http.post<LoginResponse>({
127         `${this.apiUrl}/login.php`,
128         { email, password },
129         {
130             headers: new HttpHeaders({ 'Content-Type': 'application/json' }),
131             withCredentials: true
132         }
133     }).pipe(
134         map(response => {
135             if (response.success) {
136                 localStorage.setItem('currentUser', JSON.stringify(response.user));
137             }
138             return response;
139         }),
140         catchError(error => {
141             console.error('Error en login:', error);
142             throw error;
143         })
144     );
145 }
146
147 /**
148  * Registra un nuevo usuario en el sistema
149  * @param email - Correo electrónico del nuevo usuario
150  * @param fullName - Nombre completo
151  * @param password - Contraseña
152  * @param notificationType - Tipo de notificación preferida
153  * @param contactValue - Valor de contacto según el tipo de notificación
154  * @returns Observable con la respuesta del registro
155  */
156 registerUser(email: string, fullName: string, password: string,
157 notificationType: string, contactValue: string): Observable<RegisterResponse> {
158     return this.http.post<RegisterResponse>({
159         `${this.apiUrl}/register.php`,
160         { email, fullName, password, notificationType, contactValue },
161         this.httpOptions
162     }).pipe(
163         map(response => {
164             if (response.success) {
165                 localStorage.setItem('currentUser', JSON.stringify(response.user));
166             }
167             return response;
168         }),
169         catchError(error => {
170             console.error('Error en registro:', error);
171             throw error;
172         })
173     );
174 }
175
176 /**
177  * Obtiene las facturas del usuario o taller actual
178  * @returns Observable con la lista de facturas
179  */
180 obtenerFacturas(): Observable<InvoiceResponse> {
181     const currentUser = this.getCurrentUser();
182     if (!currentUser) {
183         return throwError(() => new Error('Usuario no autenticado'));
184     }
185
186     return this.http.get<InvoiceResponse>(`${this.apiUrl}/Invoices.php`, this.httpOptions).pipe(
187         map(response => response),
188         catchError(error => {
189             console.error('Error al obtener las facturas:', error);
190             if (error.status === 401) {
191                 return throwError(() => new Error('Sesión expirada o no válida. Por favor, inicia sesión nuevamente.'));
192             }
193             return throwError(() => new Error('Error al obtener las facturas'));
194         })
195     );
196 }

```

```

198 /**
199  * Crea una nueva factura
200  * @param appointmentId - ID de la cita asociada
201  * @param items - Array de items de la factura
202  * @param estado - Estado de la factura (por defecto 'Pendiente')
203  * @returns Observable con la respuesta de la creación
204  */
205 crearFactura(appointmentId: number, items: any[], estado: string = 'Pendiente'): Observable<any> {
206   const currentUser = this.getCurrentUser();
207   if (!currentUser) {
208     return throwError(() => new Error('Usuario no autenticado'));
209   }
210
211   const body = {
212     appointment_id: appointmentId,
213     estado: estado,
214     items: items
215   };
216
217   return this.http.post<any>(`${this.apiUrl}/Invoices.php`, body, this.httpOptions).pipe(
218     map(response => response),
219     catchError(error => {
220       console.error('Error al crear la factura:', error);
221       return throwError(() => new Error('Error al crear la factura'));
222     })
223   );
224 }
225
226 /**
227  * Consulta los horarios disponibles de un taller
228  * @param workshopID - ID del taller
229  * @param fechaInicio - Fecha de inicio de la consulta
230  * @param fechaFin - Fecha fin de la consulta
231  * @returns Observable con los horarios disponibles
232  */
233 consultarSemana(workshopID: number, fechaInicio: string, fechaFin: string): Observable<any> {
234   const body = {
235     accion: 'consultar_semana',
236     WorkshopID: workshopID,
237     FechaInicio: fechaInicio,
238     FechaFin: fechaFin
239   };
240
241   return this.http.post<any>(`${this.apiUrl}/appointments.php`, body, {
242     ...this.httpOptions,
243     withCredentials: true
244   }).pipe(
245     map(response => response),
246     catchError(error => {
247       console.error('Error al consultar huecos de la semana:', error);
248       throw error;
249     })
250   );
251 }
252

```

```

crearCita(cita: { Fecha: string, HoraInicio: string, VehicleID: number,
WorkshopID: number, Motivo: string }): Observable<any> {
  const body = {
    accion: 'crear',
    Fecha: cita.Fecha,
    Hora: cita.HoraInicio,
    VehicleID: cita.VehicleID,
    WorkshopID: cita.WorkshopID,
    Descripcion: cita.Motivo,
    Estado: 'Pendiente'
  };

  console.log('Enviando datos de cita:', body);

  return this.http.post<any>(`${this.apiUrl}/appointments.php`, body, {
    headers: new HttpHeaders({ 'Content-Type': 'application/json' }),
    withCredentials: true
  }).pipe(
    map(response => {
      console.log('Respuesta del servidor:', response);
      if (!response.success) {
        console.warn('No se pudo crear la cita:', response.message);
      }
      return response;
    }),
    catchError(error => {
      console.error('Error al crear la cita:', error);
      throw error;
    })
  );
}

/**
 * Crea un nuevo vehículo para el usuario actual
 * @param vehiculo - Objeto con los datos del vehículo
 * @returns Observable con la respuesta de la creación
 */
crearVehiculo(vehiculo: { marca: string, modelo: string, anyo: string,
matricula: string }): Observable<any> {
  const body = {
    accion: 'crear',
    marca: vehiculo.marca,
    modelo: vehiculo.modelo,
    anyo: vehiculo.anyo,
    matricula: vehiculo.matricula
  };

  return this.http.post<any>(`${this.apiUrl}/Vehicles.php`, body, {
    headers: new HttpHeaders({ 'Content-Type': 'application/json' }),
    withCredentials: true
  }).pipe(
    map(response => {
      if (!response.success) {
        console.warn('No se pudo crear el vehículo:', response.message);
      }
      return response;
    }),
    catchError(error => {
      console.error('Error al crear el vehículo:', error);
      throw error;
    })
  );
}

```

Aquí temos definidos os métodos todos de acceso a datos dos ficheiros de PHP