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

Tutor	Constantino García Ulla – Javier Ulla Berdullas - Pablo Irimia Rega

Alumno

Alumno	Nicolás García Moreira

Datos do Proxecto

Título	*Autocare HUB*

CONTROL DE VERSIÓNS:

Versión	Data	Observacións
V0		
V1		
V2		

Índice:

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 alixeirar 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.

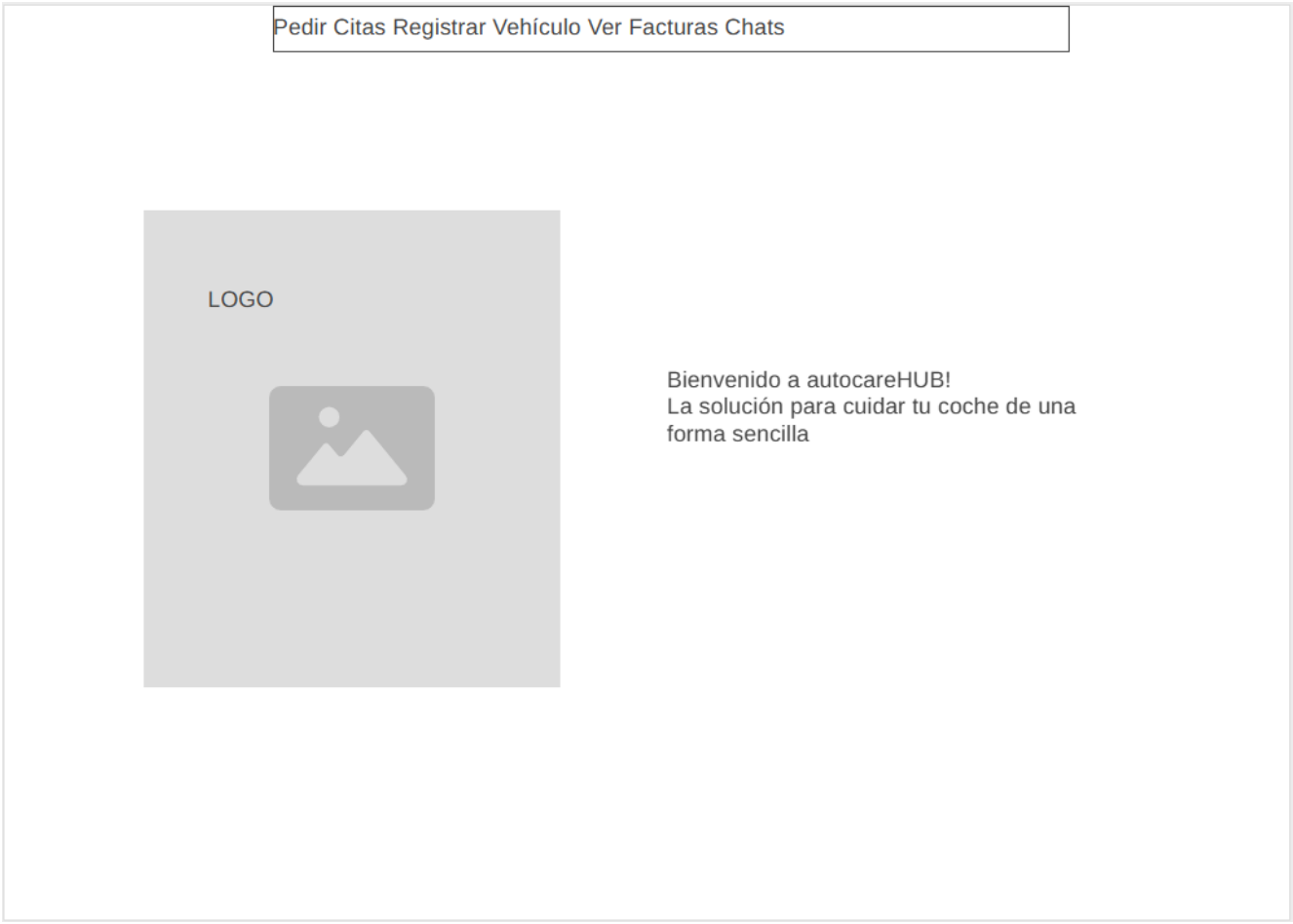
A motivación persoal que me levou a facer este proxecto, e que os talleres aos que habitúo levar o coche, non teñen un bo sistema de xestión, tanto para as facturas, como para a parte de levar o control das citas, e creo que sería bo para os talleres, ter un bo software para facer este tipo de cousas.

Uso de la aplicación

En caso de que un usuario no teña unha conta na aplicación, ten que acceder al formulario de rexistro da conta, para poder crear unha e acceder a aplicación

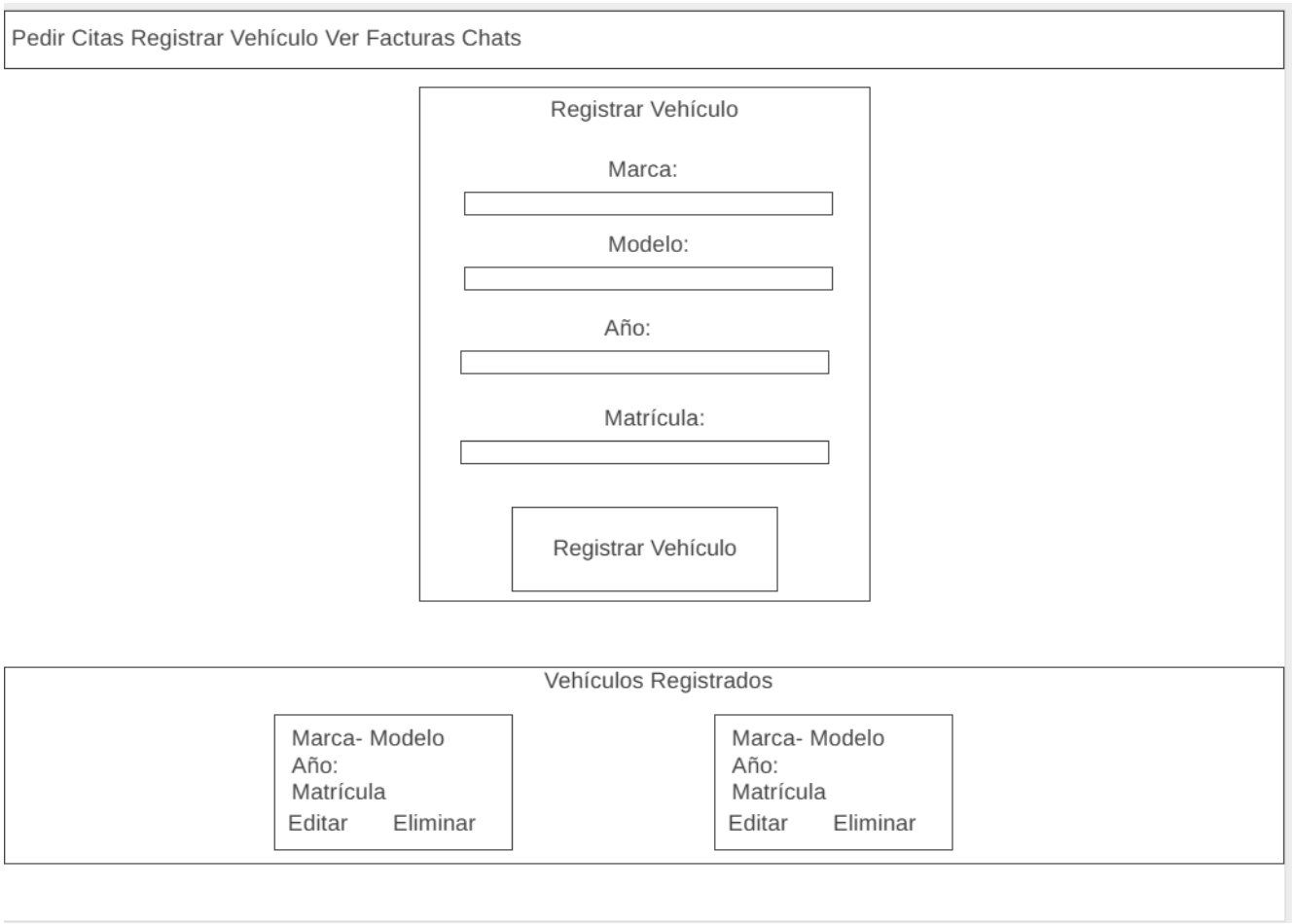
Unha vez temos unha conta creada, o próximo paso que temos que facer e iniciar sesión

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

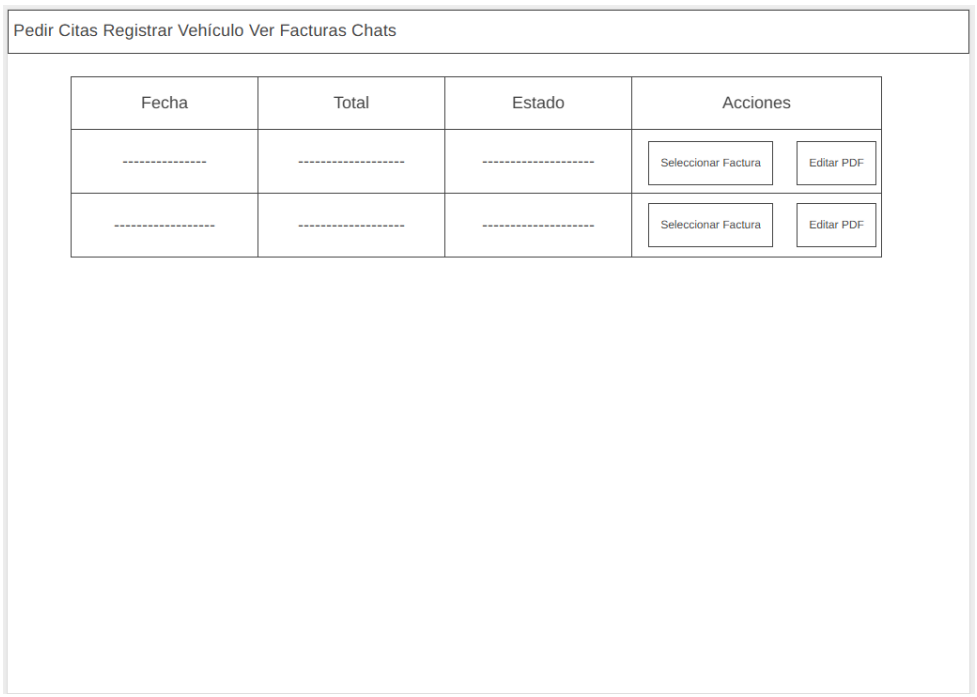


Unha vez pasemos o login, poderemos movernos libremente pola web, tanto para pedir citas, como para poder ver as nosas facturas, e tamen poder interactuar co taller.

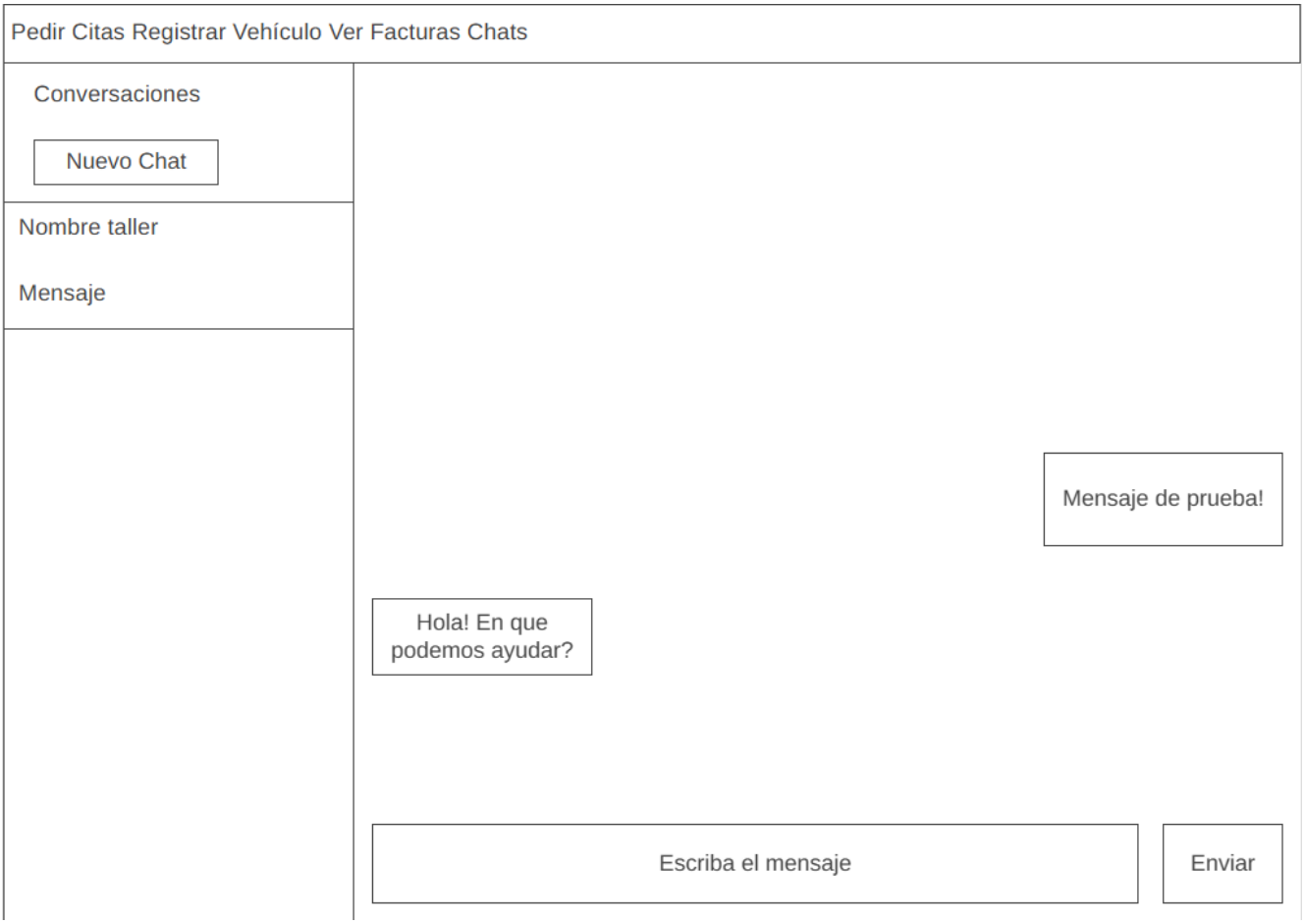
Para poder rexistrar un vehículo, teremos que ir a opción de “Registrar vehículo” e completar todos os campos



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

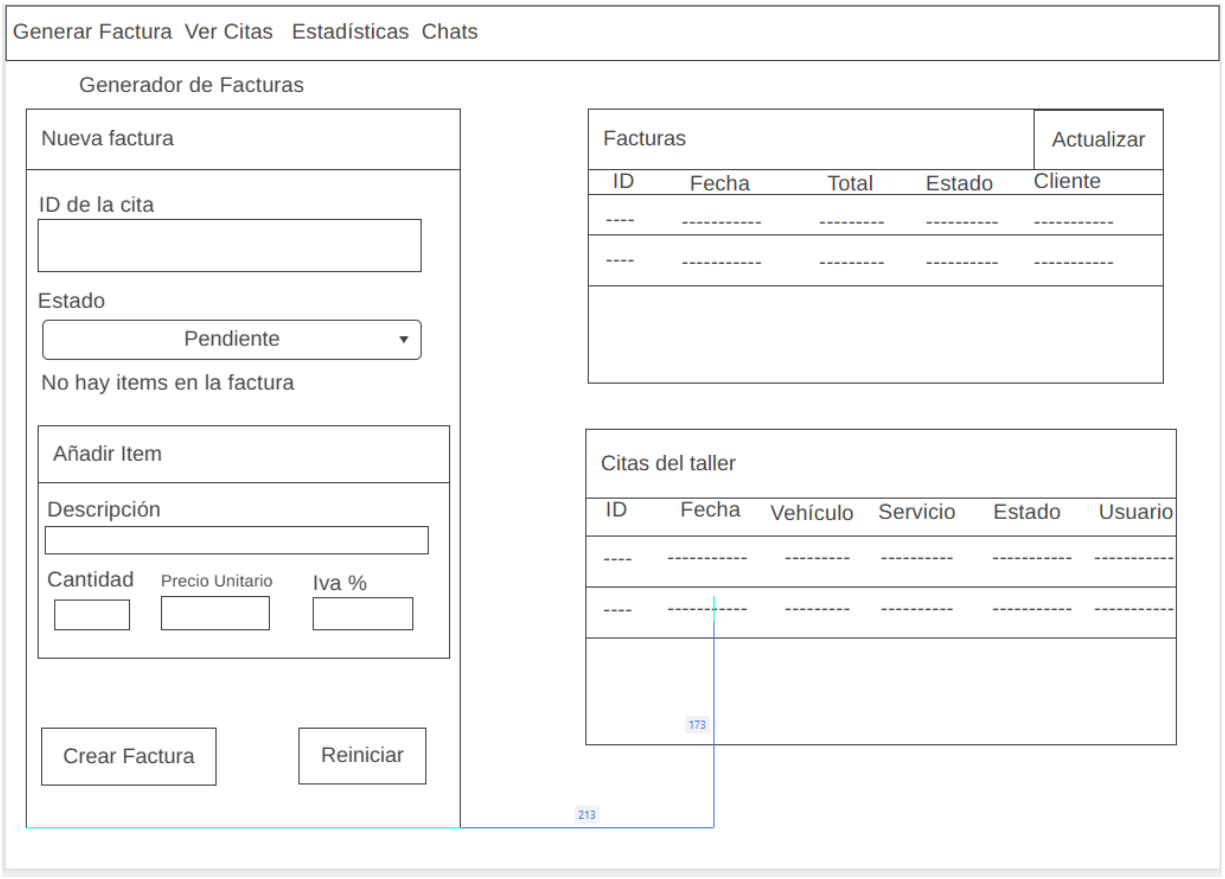


Tamén poderemos visitar o apartado de chats, donde poderemos falar cos talleres que hai na aplicación para facer as consultas que nos queiramos.

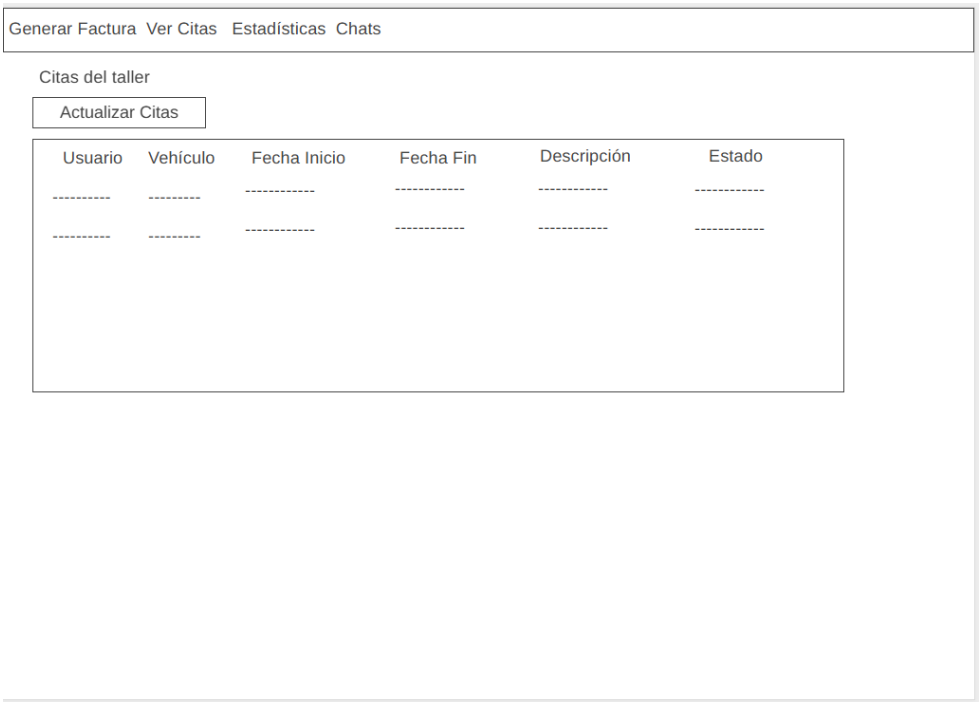


Esto eran as vistas para a parte do usuario, ahora veremos as vistas que temos dispoñibles para o taller

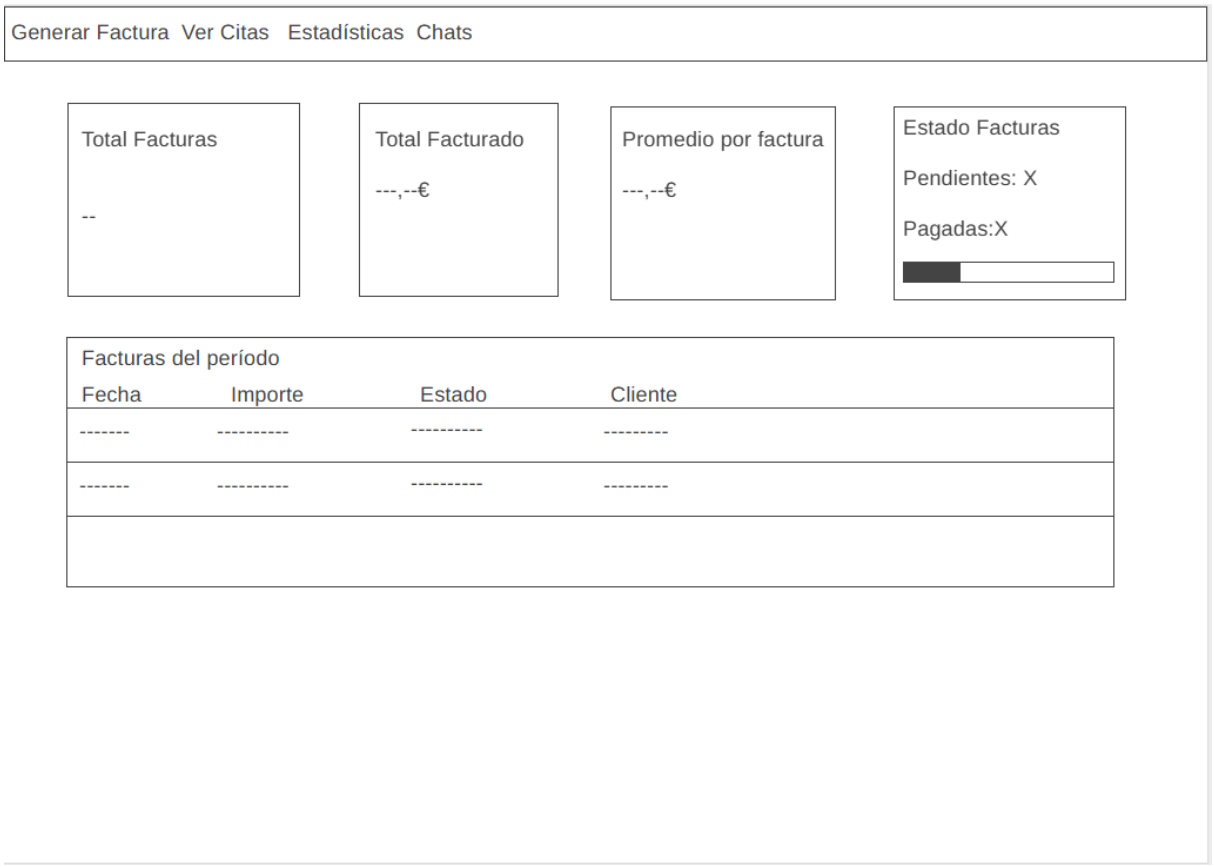
A primeira vista que temos e a de poder xerar as facturas para as citas que os usuarios solicitaron



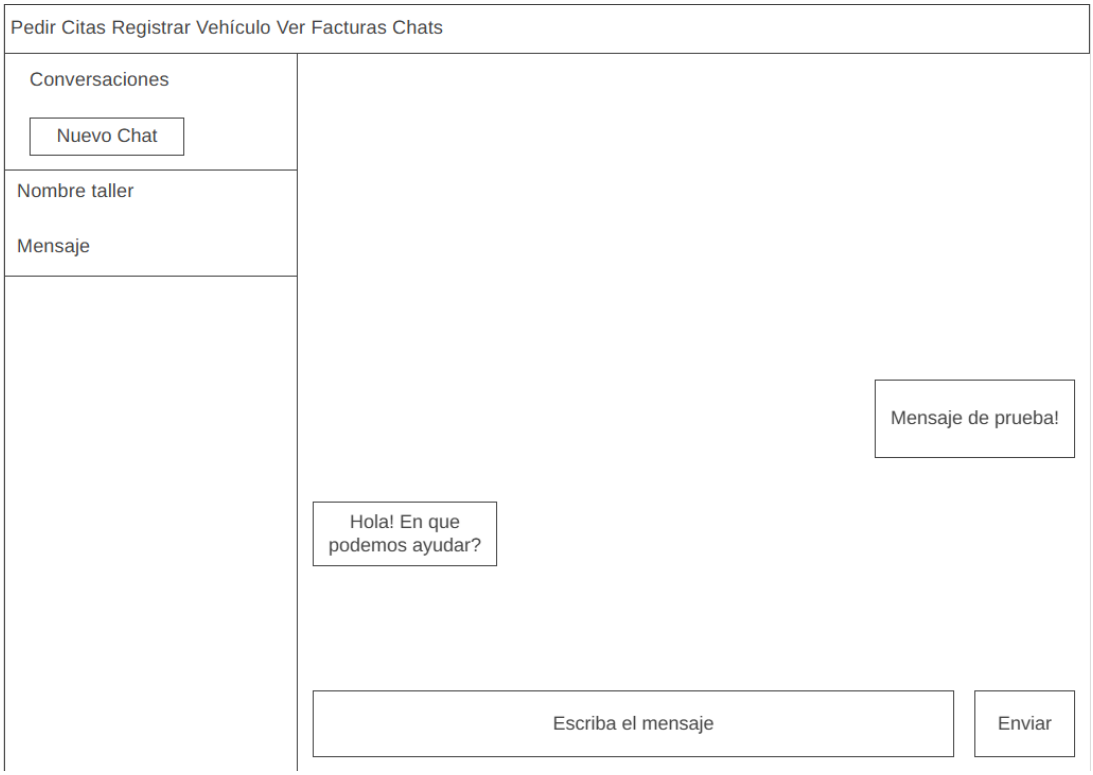
A segunda que temos dispoñible a de ver as propias citas que os usuarios solicitaron



Xa mais orientado a parte da vision estadistica 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...



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



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, sole acontecer 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 recurrentes, e dicir, que se podan programar citas cada 2 meses, cada 6 meses e así, para que os usuarios non teñan que estar pendientes 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 dia 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 desenvoler a aplicación

-> Plataforma de trello para a organización das tarefas a desenvolver.

-> Para o diseño da base de datos usarei a aplicación DIA

-> Para o diseño da interfaz web, usarase Wireframe.cc

-> Para probar as rutas e o funcionamento de PHP, usarase Postman

-> Para o diseño do logo usarase Inkskape 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. Presuposto

O presuposto do proxecto é:

Mano de obra

Duración do desarrollo	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

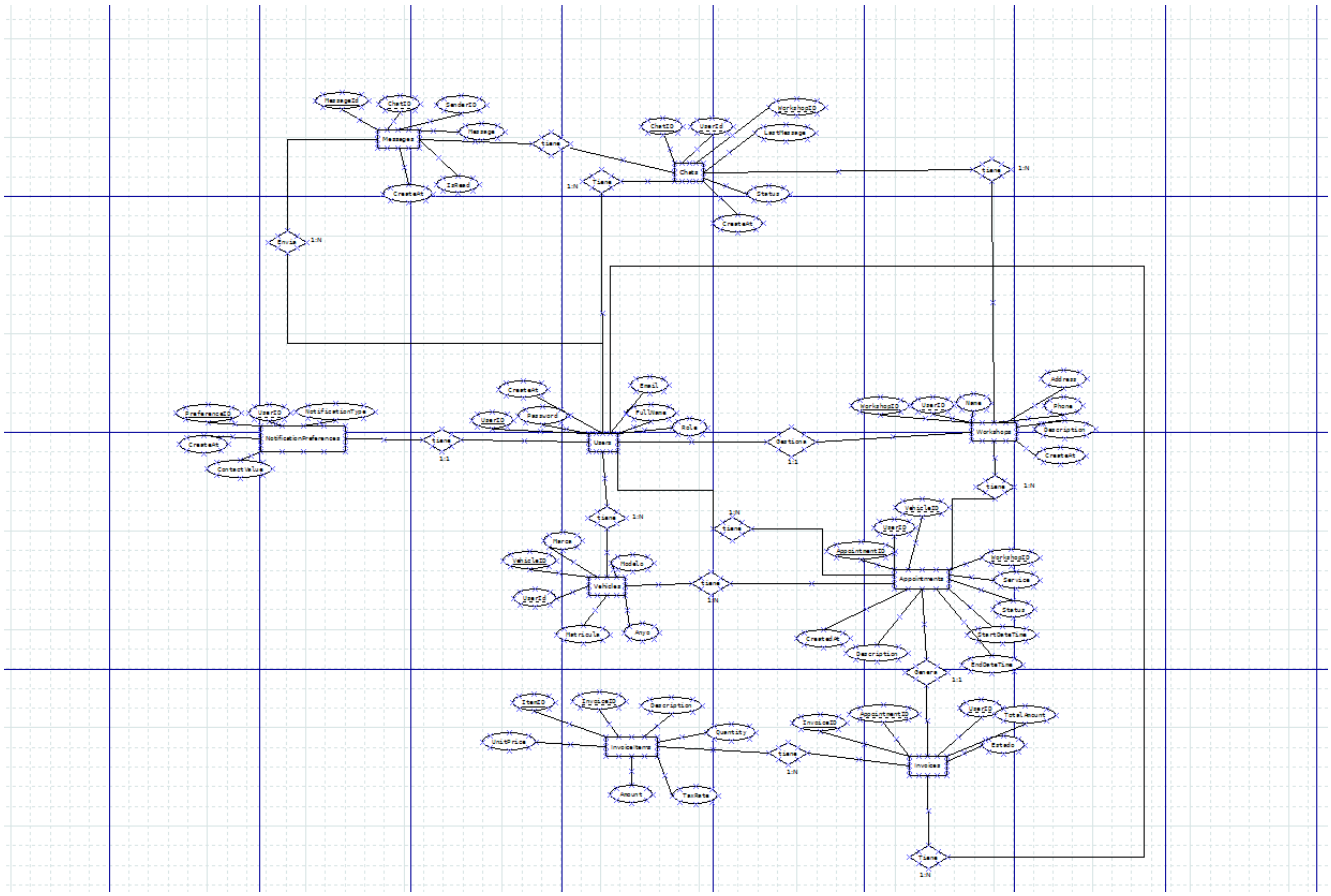
Autocare HUB

8. Execución

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

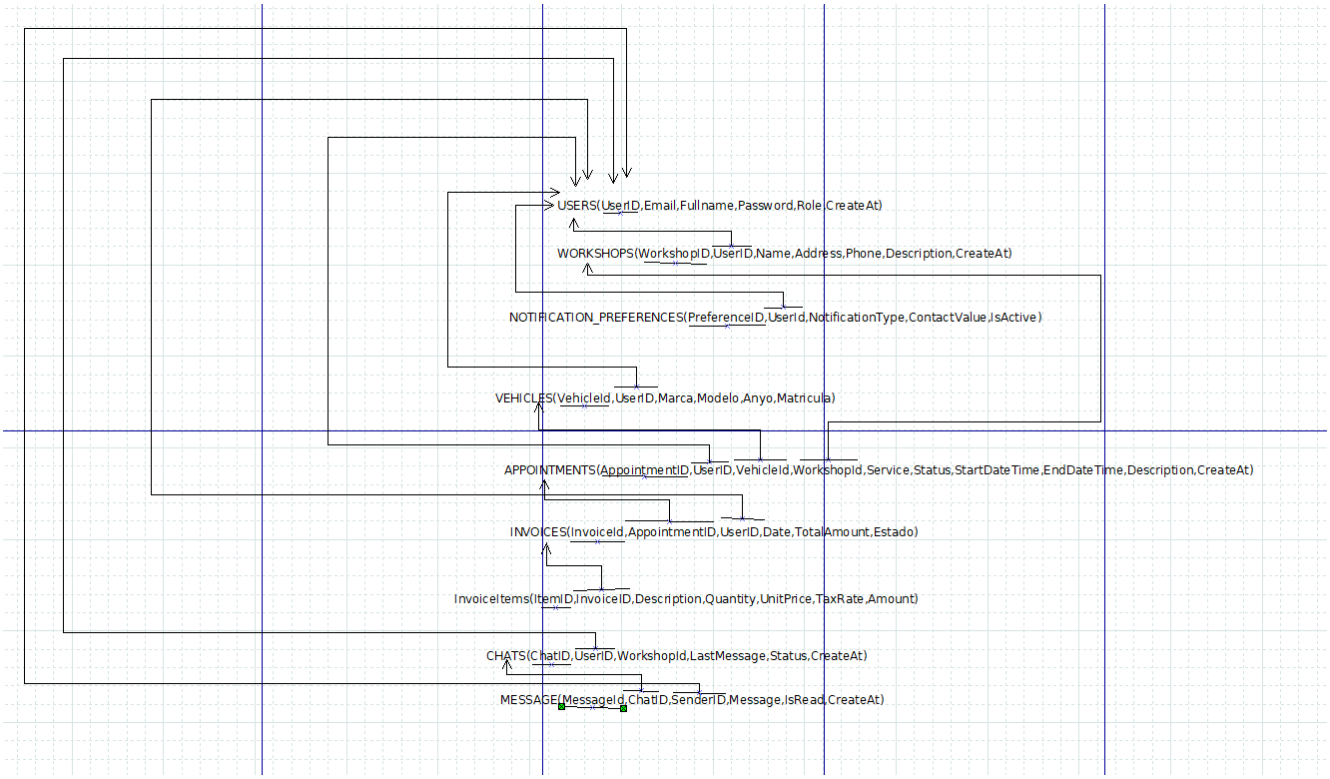
1.Diseño de base de datos

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:



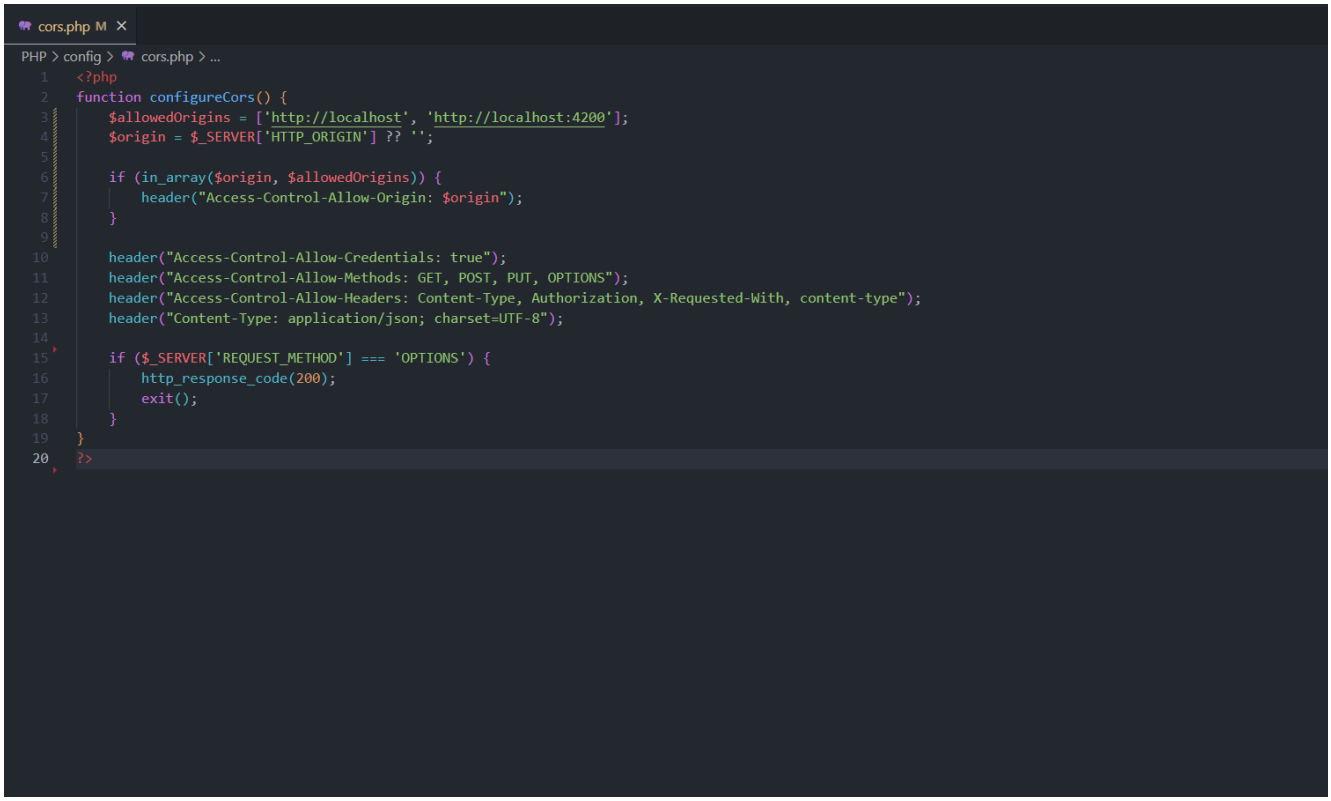
Detalle das restriccións da base de datos:

- Se un **usuario** é eliminado, elimínanse tamén todos os seus datos relacionados: os seus vehículos, citas, preferencias de notificación, facturas, conversas, 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 conversas 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 desarrollo do backend, utilizarase PHP, mais concretamente a versión 8.2.12.

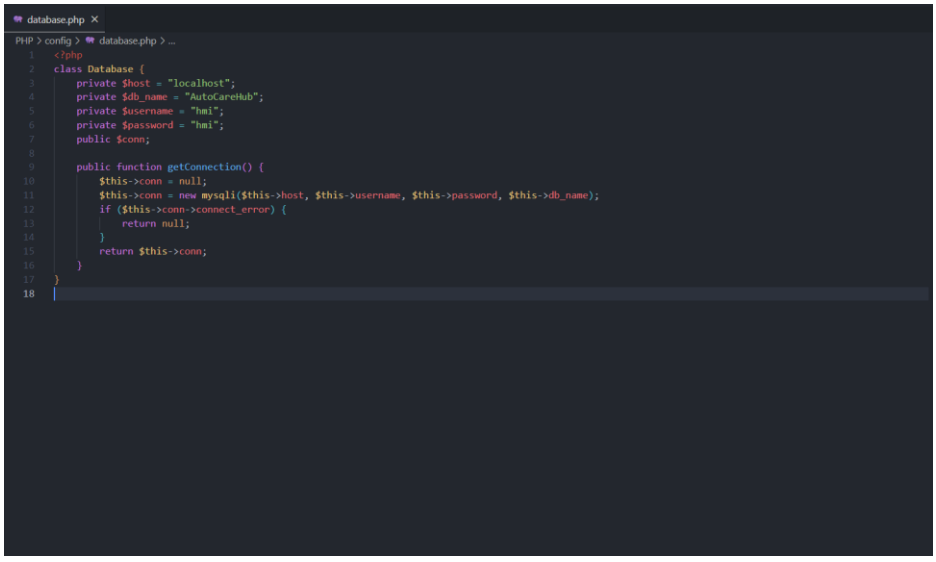
Para solucionar o problema de CORS,que e "Cross-Origin Resource Sharing", o "Intercambio de Recursos de Orixen Cruzado" en galego. Esto e un mecanismo de seguridade que teñen os navegadores, para controlar de forma mais sinxela a xestión de acceso a datos, imáxenes,e scripts.



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

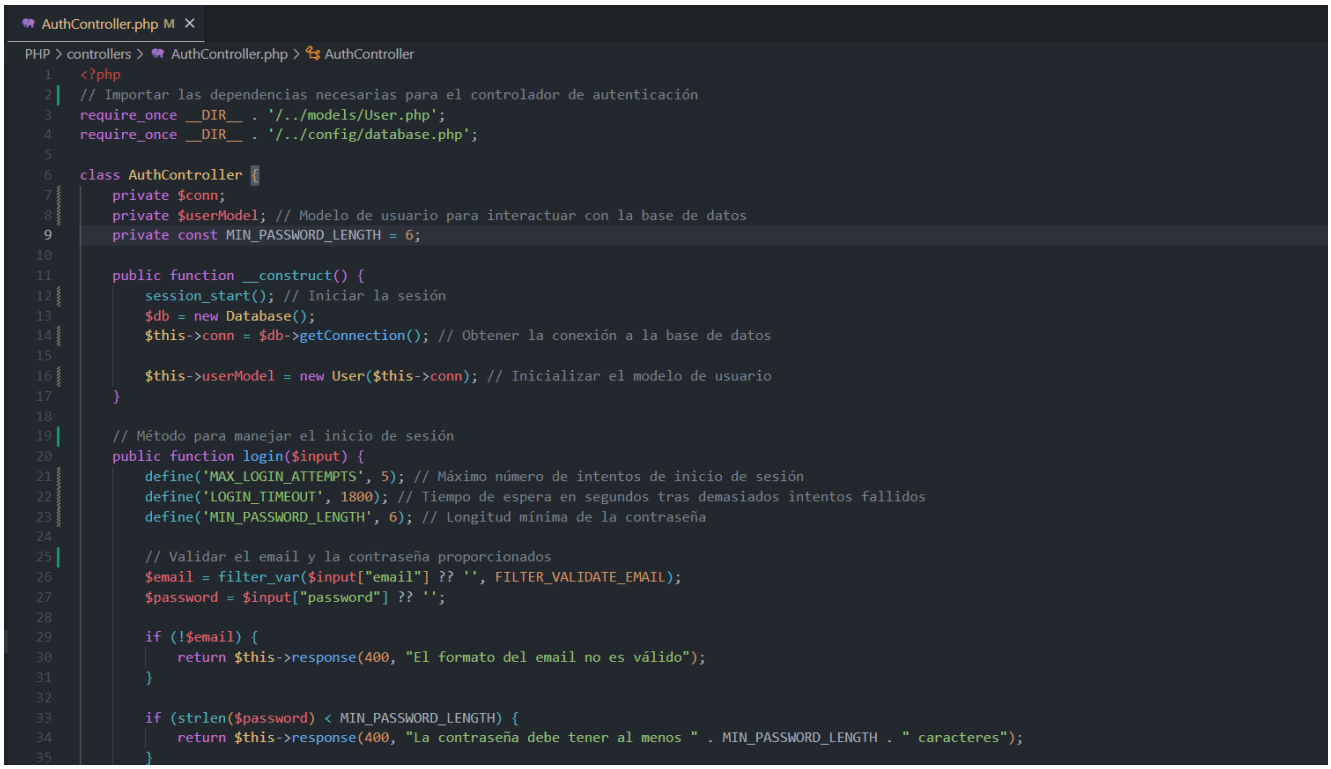
Unha vez estas cabeceiras estén na solicutud presentes, xa teremos o acceso aos datos.

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



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 a autenticación do usuario para o acceso a datos, estaremos usando un controlador



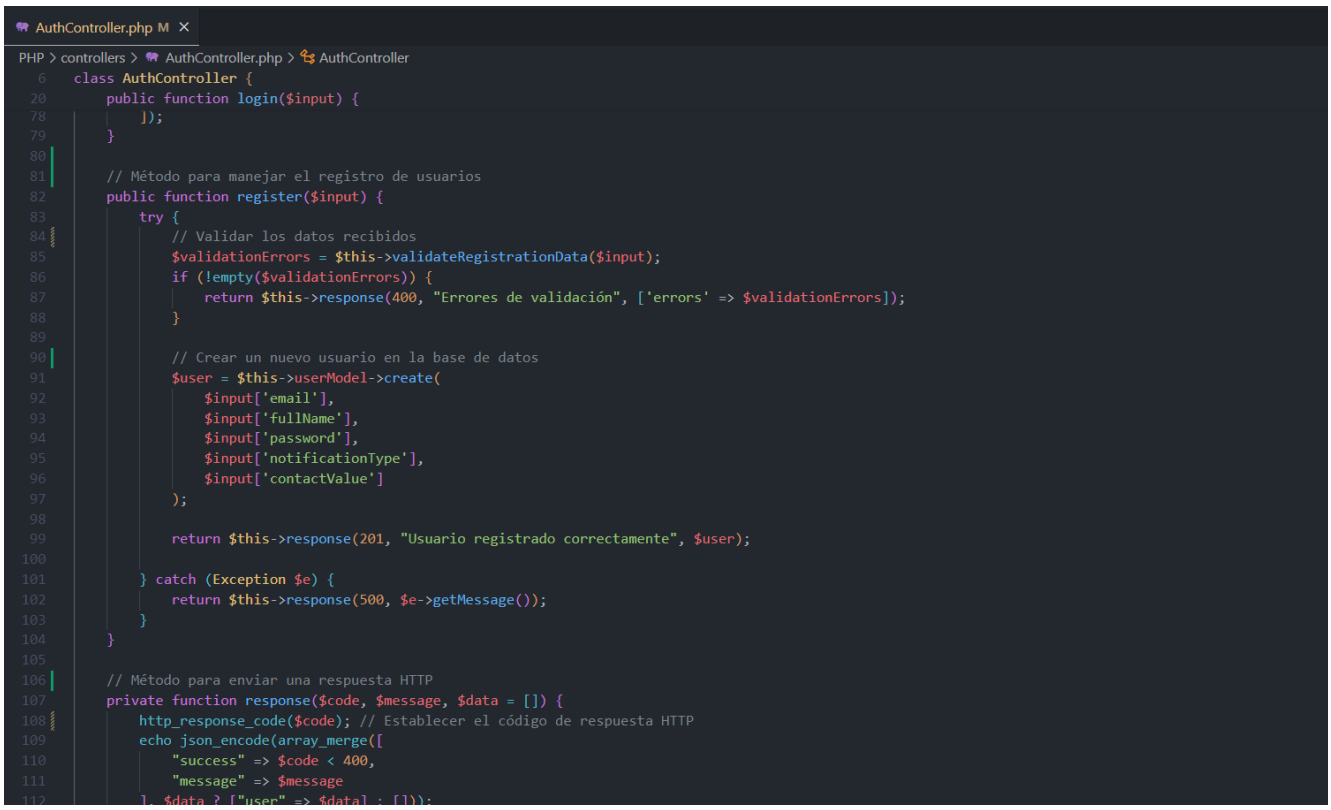
Aquí podemos ver que temos definida una constante para o mínimo de lonxitude de caracteres, estaremos tamén definindo o máximo de intentos máximos para poder iniciar sesión, e o tempo que tarda en volver a estar dispoñible a conta que se bloqueou, por logearse varias con unha contraseña incorrecta.

Tamén está a verificar que o email que se está a introducir e un correo que cumple os requisitos.

En caso de que o email non teña o formato correcto, devolvera un error, a parte de front.

Tamén en caso de que a lonxitude mínima non cumpra cos requisitos

Obteremos a conexión a base de datos, que a obtemos do fichero de Database.php



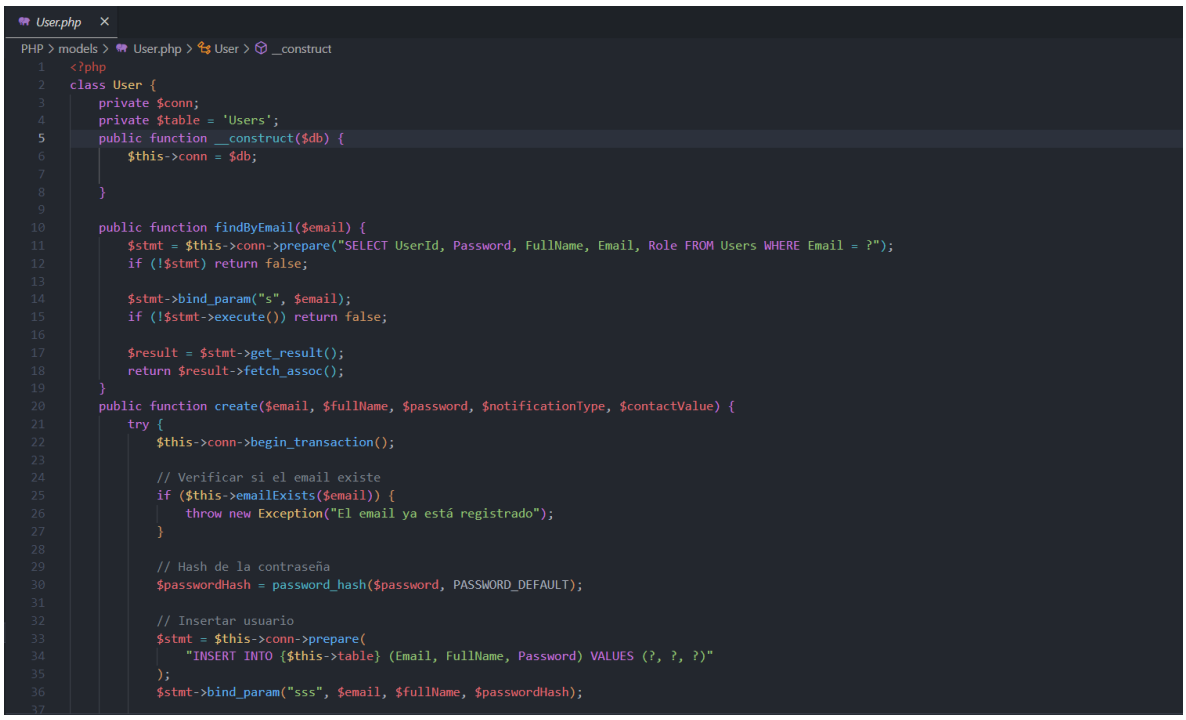
No controlador de autenticación tamen temos a xestión de registro.

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 nombre do usuario polo menos teña 3 caracteres, que a contraseña teña o numero 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, crearase o usuario na base de datos.

E despois mandarase unha resposta ao front.



```php
<?php
class User {
    private $conn;
    private $table = 'Users';
    public function __construct($db) {
        $this->conn = $db;

    }

    public function findByEmail($email) {
        $stmt = $this->conn->prepare("SELECT UserId, Password, FullName, Email, Role FROM Users WHERE Email = ?");
        if (!$stmt) return false;

        $stmt->bind_param("s", $email);
        if (!$stmt->execute()) return false;

        $result = $stmt->get_result();
        return $result->fetch_assoc();
    }
    public function create($email, $fullName, $password, $notificationType, $contactValue) {
        try {
            $this->conn->begin_transaction();

            // Verificar si el email existe
            if ($this->emailExists($email)) {
                throw new Exception("El email ya está registrado");
            }

            // Hash de la contraseña
            $passwordHash = password_hash($password, PASSWORD_DEFAULT);

            // Insertar usuario
            $stmt = $this->conn->prepare(
                "INSERT INTO {$this->table} (Email, FullName, Password) VALUES (?, ?, ?)"
            );
            $stmt->bind_param("sss", $email, $fullName, $passwordHash);
```

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

Temos as siguientes 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”.

```php
class User {
    public function findByEmail($email) {
        return $result->fetch_assoc();
    }
    public function create($email, $fullName, $password, $notificationType, $contactValue) {
        try {
            $this->conn->begin_transaction();

            // Verificar si el email existe
            if ($this->emailExists($email)) {
                throw new Exception("El email ya está registrado");
            }

            // Hash de la contraseña
            $passwordHash = password_hash($password, PASSWORD_DEFAULT);

            // Insertar usuario
            $stmt = $this->conn->prepare(
                "INSERT INTO {$this->table} (Email, FullName, Password) VALUES (?, ?, ?)"
            );
            $stmt->bind_param("sss", $email, $fullName, $passwordHash);

            if (!$stmt->execute()) {
                throw new Exception("Error al registrar el usuario");
            }

            $userId = $stmt->insert_id;

            // Insertar preferencia de notificación
            $stmtPref = $this->conn->prepare(
                "INSERT INTO NotificationPreferences (UserID, NotificationType, ContactValue, IsActive)
                 VALUES (?, ?, ?, true)"
            );
            $stmtPref->bind_param("iss", $userId, $notificationType, $contactValue);

            if (!$stmtPref->execute()) {
```

ç

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

PHP > routes > register.php > ...

```php
<?php
require_once __DIR__ . '/../config/cors.php';
require_once __DIR__ . '/../controllers/AuthController.php';

configureCors();

$controller = new AuthController();

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $input = json_decode(file_get_contents("php://input"), true);
    $controller->register($input);
} else {
    http_response_code(405);
    echo json_encode(["success" => false, "message" => "Método no permitido"]);
}
```

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

PHP > routes > login.php > ...

```php
<?php
require_once __DIR__ . '/../config/cors.php';
require_once __DIR__ . '/../controllers/AuthController.php';

configureCors();

$controller = new AuthController();

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

    // Establecer una cookie para recordar el rol del usuario
    if (isset($_SESSION['user']['role'])) {
        setcookie('user_role', $_SESSION['user']['role'], time() + (86400 * 30), "/"); // Expira en 30 días
    }
} else {
    http_response_code(405);
    echo json_encode(["success" => false, "message" => "Método no permitido"]);
}
```

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

PHP > models > Appointment.php > AppointmentModel

```php
<?php
class AppointmentModel {
    private $conn;
    private $table = 'Appointments';

    // Propiedades de la cita
    public $AppointmentID;
    public $UserID;
    public $VehicleID;
    public $WorkshopID;
    public $StartDateTime;
    public $EndDateTime;
    public $Descripcion;
    public $Status;

    // Constructor con conexión a la base de datos
    public function __construct($db) {
        $this->conn = $db;
    }

    // Método para obtener el taller asociado a un usuario
    public function getWorkshopByUserId($userId) {
        $query = "SELECT WorkshopID FROM Workshops WHERE UserID = ?";

        $stmt = $this->conn->prepare($query);
        $stmt->bind_param("i", $userId);
        $stmt->execute();

        return $stmt->get_result();
    }

    // Método para obtener slots semanales
    public function getWeeklySlots($workshopId, $startDate, $endDate) {
        $query = "SELECT StartDateTime
                  FROM " . $this->table . "
                  WHERE WorkshopID = ?
                  AND DATE(StartDateTime) BETWEEN ? AND ?
```

```php
    // Método para obtener slots diarios
    public function getDaySlots($workshopId, $date) {
        $query = "SELECT StartDateTime
                  FROM " . $this->table . "
                  WHERE WorkshopID = ?
                  AND DATE(StartDateTime) = ?";

        $stmt = $this->conn->prepare($query);
        $stmt->bind_param("is", $workshopId, $date);
        $stmt->execute();

        return $stmt->get_result();
    }
```

```php
    // Método para verificar si existe un taller
    public function checkWorkshopExists($workshopId) {
        $query = "SELECT WorkshopID FROM Workshops WHERE WorkshopID = ?";

        $stmt = $this->conn->prepare($query);
        $stmt->bind_param("i", $workshopId);
        $stmt->execute();

        return $stmt->get_result()->num_rows > 0;
    }
```

```php
    // Método para obtener slots semanales
    public function getWeeklySlots($workshopId, $startDate, $endDate) {
        $query = "SELECT StartDateTime
                  FROM " . $this->table . "
                  WHERE WorkshopID = ?
                  AND DATE(StartDateTime) BETWEEN ? AND ?
                  AND DAYOFWEEK(StartDateTime) BETWEEN 2 AND 6";

        $stmt = $this->conn->prepare($query);
        $stmt->bind_param("iss", $workshopId, $startDate, $endDate);
        $stmt->execute();

        return $stmt->get_result();
    }
```

```php
    // Método para crear una nueva cita
    public function create() {
        $query = "INSERT INTO " . $this->table . "
                  (UserID, VehicleID, WorkshopID, StartDateTime, EndDateTime, Description, Status)
                  VALUES (?, ?, ?, ?, ?, ?, ?)";

        $stmt = $this->conn->prepare($query);

        // Limpiar y sanitizar los datos
        $this->Descripcion = htmlspecialchars(strip_tags($this->Descripcion));
        $this->Status = htmlspecialchars(strip_tags($this->Status));

        // Vincular los parámetros
        $stmt->bind_param("iiissss",
            $this->UserID,
            $this->VehicleID,
            $this->WorkshopID,
            $this->StartDateTime,
            $this->EndDateTime,
            $this->Descripcion,
            $this->Status
        );

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

```php
    // Método para obtener todas las citas de un taller
    public function getWorkshopAppointments($workshopId) {
        $query = "SELECT
                    a.AppointmentID,
                    a.StartDateTime,
                    a.EndDateTime,
                    CONCAT(v.Marca, ' ', v.Modelo) AS Vehiculo,
                    a.Description,
                    a.Status,
                    u.FullName AS UserName
                  FROM " . $this->table . " a
                  JOIN Vehicles v ON a.VehicleID = v.VehicleID
                  JOIN Users u ON v.UserID = u.UserID
                  WHERE a.WorkshopID = ?";

        $stmt = $this->conn->prepare($query);
        $stmt->bind_param("i", $workshopId);
        $stmt->execute();

        return $stmt->get_result();
    }
```

```php
    // Método para verificar disponibilidad de slot
    public function checkSlotAvailability($workshopId, $startDateTime) {
        $query = "SELECT AppointmentID
                  FROM " . $this->table . "
                  WHERE WorkshopID = ?
                  AND StartDateTime = ?";

        $stmt = $this->conn->prepare($query);
        $stmt->bind_param("is", $workshopId, $startDateTime);
        $stmt->execute();

        return $stmt->get_result()->num_rows === 0;
    }

    // Método para obtener el nombre del taller
    public function getWorkshopName($workshopId) {
        $stmt = $this->conn->prepare("SELECT Name FROM Workshops WHERE WorkshopID = ?");
        $stmt->bind_param("i", $workshopId);
        $stmt->execute();
        $result = $stmt->get_result();

        if ($row = $result->fetch_assoc()) {
            return $row['Name'];
        }
        return "Taller no encontrado";
    }
```

A continuación un resumo 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

PHP > controllers > AppointmentController.php > AppointmentController > consultarSemana

```php
<?php
require_once __DIR__ . '/../models/Appointment.php';

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

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

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

            $workshopId = $data['WorkshopID'];

            // Verificar que el taller existe
            if (!$this->model->checkWorkshopExists($workshopId)) {
                return $this->sendResponse(404, false, "Taller no encontrado");
            }

            $fechaActual = new DateTime();
            $fechaInicio = $fechaActual->format('Y-m-d');
            $fechaFin = (clone $fechaActual)->modify('+30 days')->format('Y-m-d');

            $result = $this->model->getWeeklySlots($workshopId, $fechaInicio, $fechaFin);
            $slots = $this->procesarSlotsSemanales($result, $fechaInicio, $fechaFin);

            return $this->sendResponse(200, true, "", ["slotsSemana" => $slots]);
        } catch (Exception $e) {
            return $this->sendResponse(500, false, "Error al consultar disponibilidad: " . $e->getMessage());
        }
    }
```

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 incluindo o día actual.

```php
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, estableceéndoa na base de datos.

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

```php
    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
<?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.

```php
    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 siguiente

```php
<?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

```php
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 requiridos 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

```php
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.

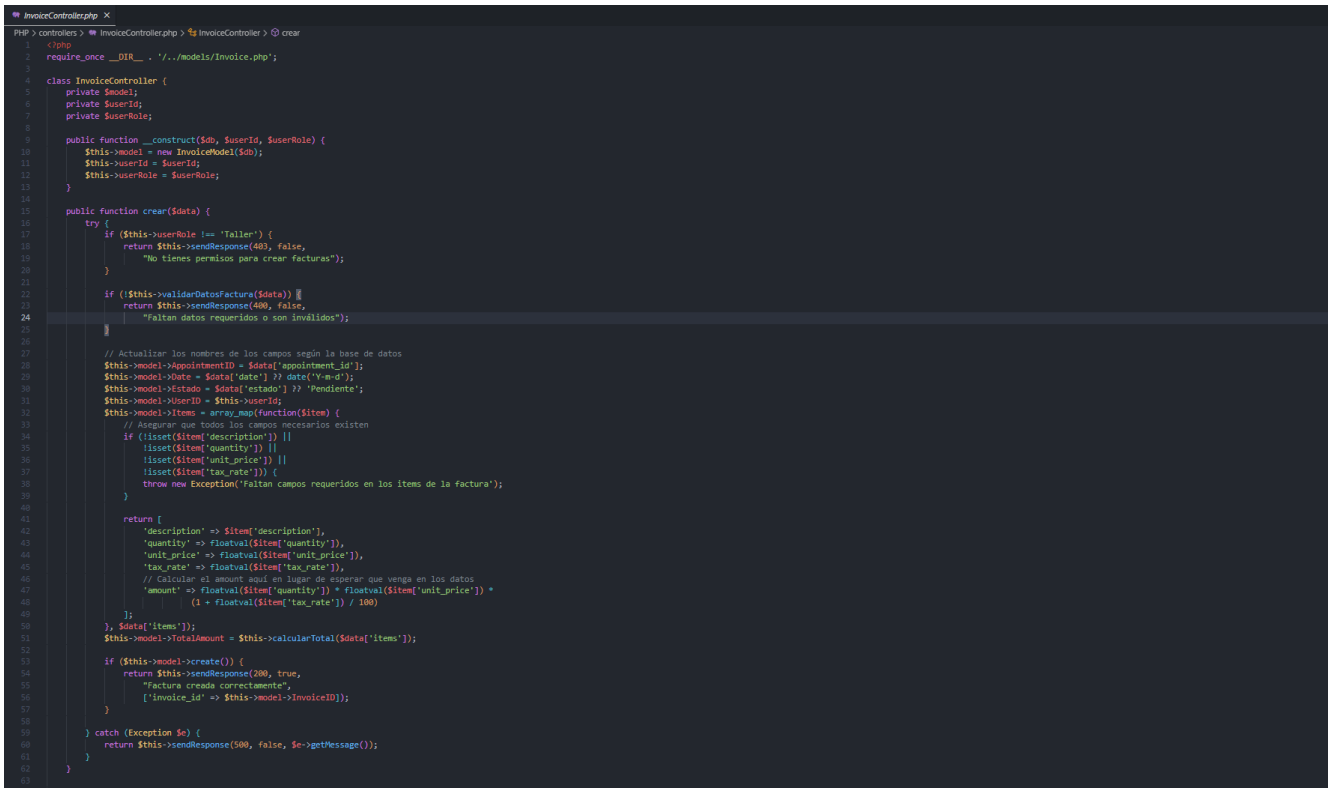
Despois mandaremos os chats todos na resposta.

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

```php
    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.



```php
<?php
require_once __DIR__ . '/../models/Invoice.php';

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

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

    public function crear($data) {
        try {
            if ($this->userRole !== 'Taller') {
                return $this->sendResponse(403, false,
                    "No tienes permisos para crear facturas");
            }

            if (!$this->validarDatosFactura($data)) {
                return $this->sendResponse(400, false,
                    "Faltan datos requeridos o son inválidos");
            }

            // Actualizar los nombres de los campos según la base de datos
            $this->model->AppointmentID = $data['appointment_id'];
            $this->model->Date = $data['date'] ?? date('Y-m-d');
            $this->model->Estado = $data['estado'] ?? 'Pendiente';
            $this->model->UserID = $this->userId;
            $this->model->Items = array_map(function($item) {
                // Asegurar que todos los campos necesarios existen
                if (!isset($item['description']) ||
                    !isset($item['quantity']) ||
                    !isset($item['unit_price']) ||
                    !isset($item['tax_rate'])) {
                    throw new Exception('Faltan campos requeridos en los items de la factura');
                }

                return [
                    'description' => $item['description'],
                    'quantity' => floatval($item['quantity']),
                    'unit_price' => floatval($item['unit_price']),
                    'tax_rate' => floatval($item['tax_rate']),
                    // Calcular el amount aquí en lugar de esperar que venga en los datos
                    'amount' => floatval($item['quantity']) * floatval($item['unit_price']) *
                                (1 + floatval($item['tax_rate']) / 100)
                ];
            }, $data['items']);
            $this->model->TotalAmount = $this->calcularTotal($data['items']);

            if ($this->model->create()) {
                return $this->sendResponse(200, true,
                    "Factura creada correctamente",
                    ['invoice_id' => $this->model->InvoiceID]);
            }

        } catch (Exception $e) {
            return $this->sendResponse(500, false, $e->getMessage());
        }
    }
```

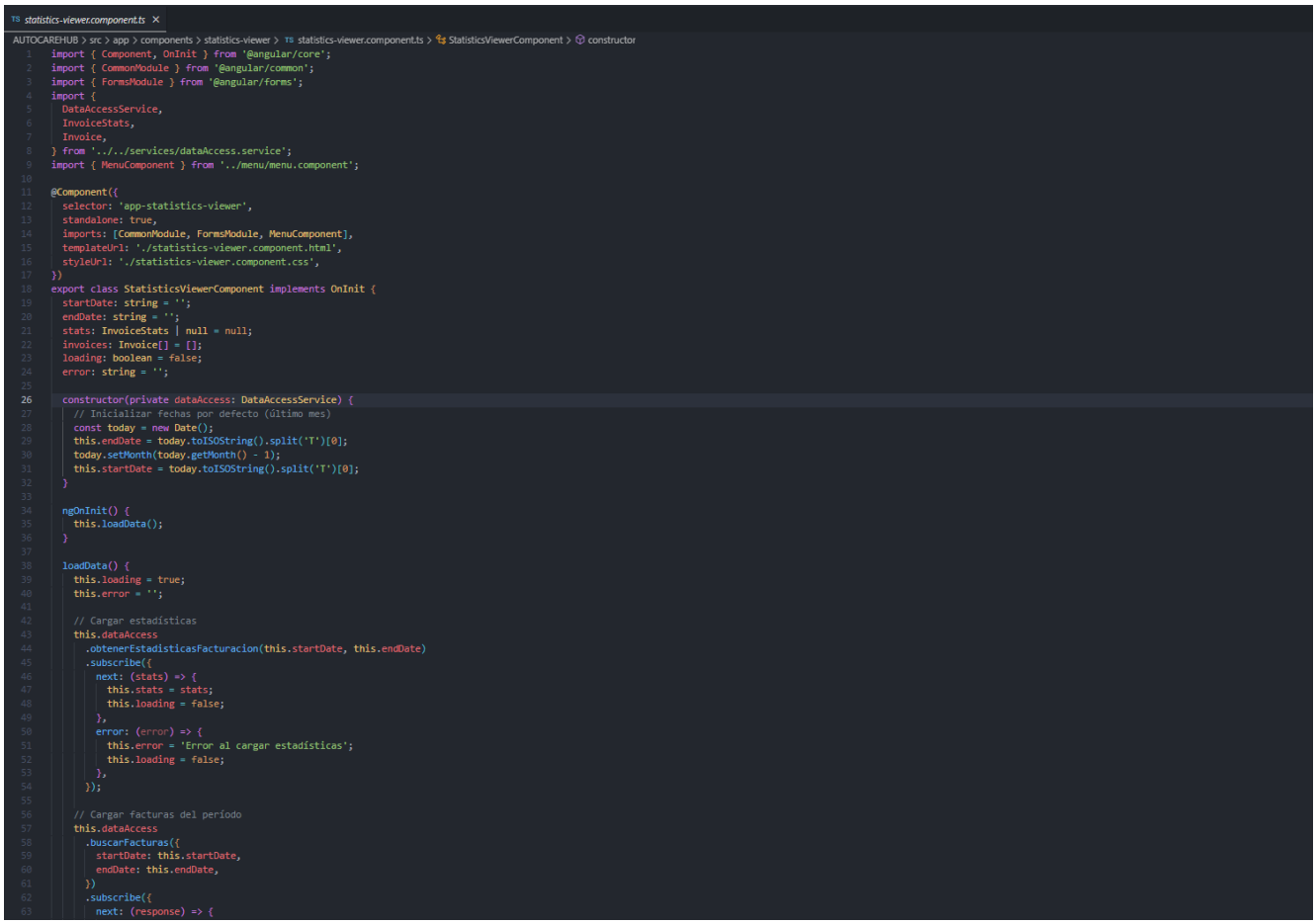
Aqui temos o método de crear unha factura, que primeiro e verificar que o rol do usuario e o correcto, despois verificamos que os datos son correctos,e os asignamos.

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



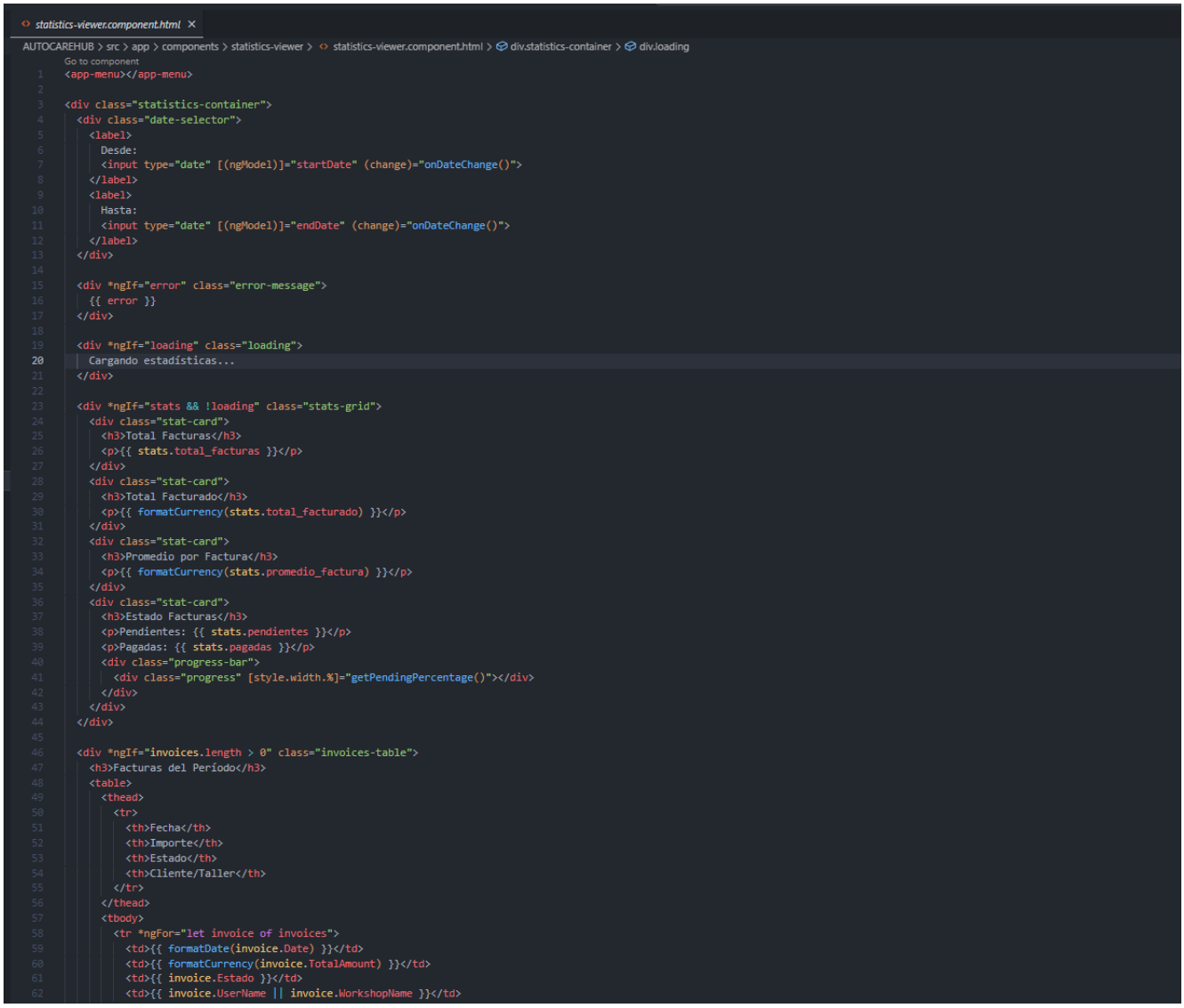
```
import { Component, OnInit } from '@angular/core';
import { CommonModule } from '@angular/common';
import { FormsModule } from '@angular/forms';
import {
  DataAccessService,
  InvoiceStats,
  Invoice,
} from '../../services/dataAccess.service';
import { MenuComponent } from '../menu/menu.component';

@Component({
  selector: 'app-statistics-viewer',
  standalone: true,
  imports: [CommonModule, FormsModule, MenuComponent],
  templateUrl: './statistics-viewer.component.html',
  styleUrl: './statistics-viewer.component.css',
})
export class StatisticsViewerComponent implements OnInit {
  startDate: string = '';
  endDate: string = '';
  stats: InvoiceStats | null = null;
  invoices: Invoice[] = [];
  loading: boolean = false;
  error: string = '';

  constructor(private dataAccess: DataAccessService) {
    // Inicializar fechas por defecto (último mes)
    const today = new Date();
    this.endDate = today.toISOString().split('T')[0];
    today.setMonth(today.getMonth() - 1);
    this.startDate = today.toISOString().split('T')[0];
  }

  ngOnInit() {
    this.loadData();
  }

  loadData() {
    this.loading = true;
    this.error = '';

    // Cargar estadísticas
    this.dataAccess
      .obtenerEstadisticasFacturacion(this.startDate, this.endDate)
      .subscribe({
        next: (stats) => {
          this.stats = stats;
          this.loading = false;
        },
        error: (error) => {
          this.error = 'Error al cargar estadísticas';
          this.loading = false;
        },
      });

    // Cargar facturas del período
    this.dataAccess
      .buscarFacturas({
        startDate: this.startDate,
        endDate: this.endDate,
      })
      .subscribe({
        next: (response) => {
```

```
    // Cargar facturas del período
    this.dataAccess
      .buscarFacturas({
        startDate: this.startDate,
        endDate: this.endDate,
      })
      .subscribe({
        next: (response) => {
          this.invoices = response.invoices;
        },
        error: (error) => {
          this.error = 'Error al cargar facturas';
        },
      });
  }

  onDateChange() {
    this.loadData();
  }

  formatCurrency(amount: number): string {
    return new Intl.NumberFormat('es-ES', {
      style: 'currency',
      currency: 'EUR',
    }).format(amount);
  }

  formatDate(date: string): string {
    return new Date(date).toLocaleDateString('es-ES');
  }

  getPendingPercentage(): number {
    if (!this.stats) return 0;
    return (this.stats.pendientes / this.stats.total_facturas) * 100;
  }
}
```

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.

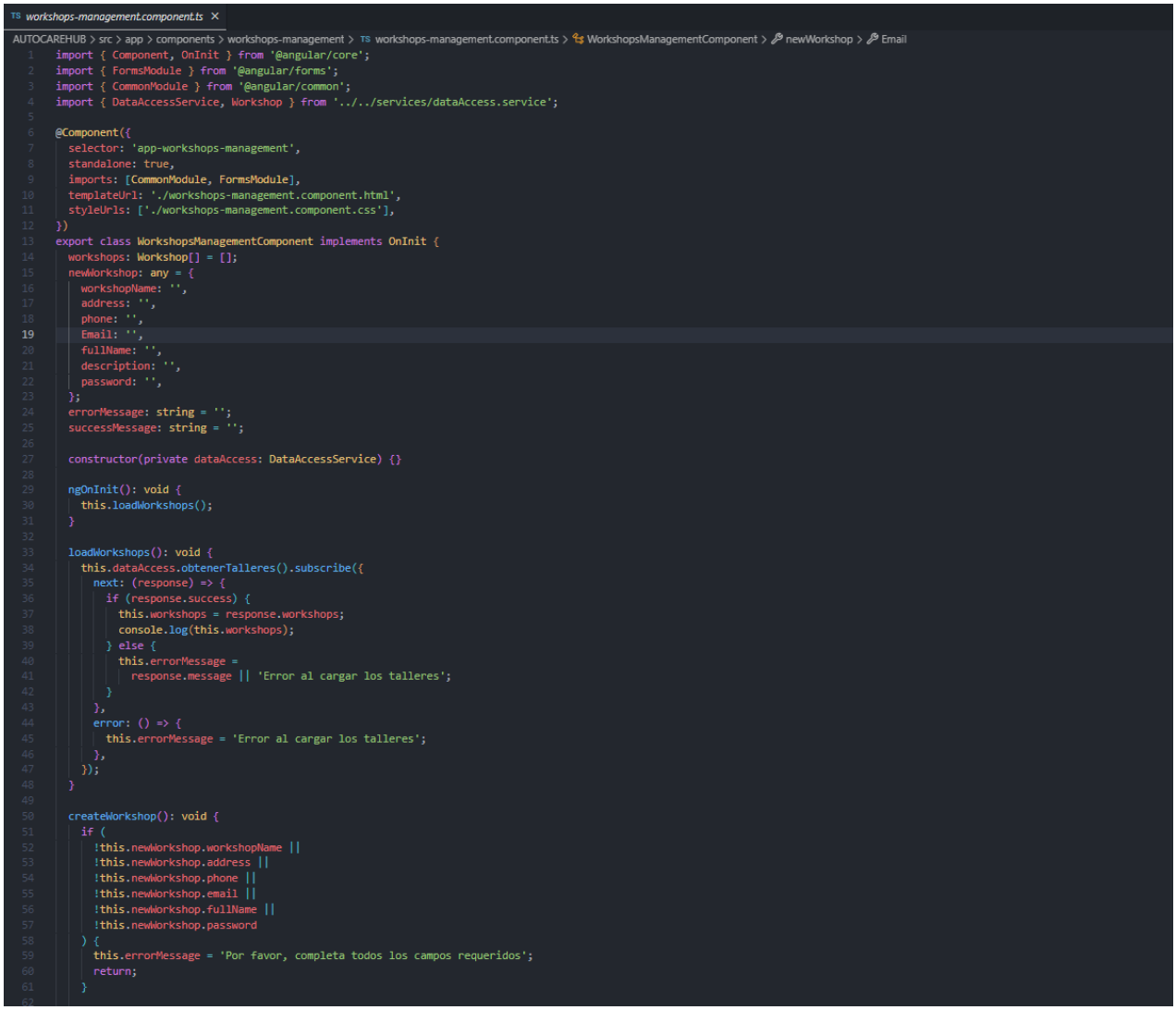
Tamén temos unha función para formatear os datos numéricos, e para formatear as datas.

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



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
import { Component, OnInit } from '@angular/core';
import { FormsModule } from '@angular/forms';
import { CommonModule } from '@angular/common';
import { DataAccessService, Workshop } from '../../services/dataAccess.service';

@Component({
  selector: 'app-workshops-management',
  standalone: true,
  imports: [CommonModule, FormsModule],
  templateUrl: './workshops-management.component.html',
  styleUrls: ['./workshops-management.component.css'],
})
export class WorkshopsManagementComponent implements OnInit {
  workshops: Workshop[] = [];
  newWorkshop: any = {
    workshopName: '',
    address: '',
    phone: '',
    Email: '',
    fullName: '',
    description: '',
    password: '',
  };
  errorMessage: string = '';
  successMessage: string = '';

  constructor(private dataAccess: DataAccessService) {}

  ngOnInit(): void {
    this.loadWorkshops();
  }

  loadWorkshops(): void {
    this.dataAccess.obtenerTalleres().subscribe({
      next: (response) => {
        if (response.success) {
          this.workshops = response.workshops;
          console.log(this.workshops);
        } else {
          this.errorMessage =
            response.message || 'Error al cargar los talleres';
        }
      },
      error: () => {
        this.errorMessage = 'Error al cargar los talleres';
      },
    });
  }

  createWorkshop(): void {
    if (
      !this.newWorkshop.workshopName ||
      !this.newWorkshop.address ||
      !this.newWorkshop.phone ||
      !this.newWorkshop.email ||
      !this.newWorkshop.fullName ||
      !this.newWorkshop.password
    ) {
      this.errorMessage = 'Por favor, completa todos los campos requeridos';
      return;
    }
```

```ts
    this.dataAccess.createWorkshop(this.newWorkshop).subscribe({
      next: (response) => {
        if (response.success) {
          this.successMessage = 'Taller creado exitosamente';
          this.loadWorkshops();
          this.resetForm();
        } else {
          this.errorMessage = response.message || 'Error al crear el taller';
        }
      },
      error: () => {
        this.errorMessage = 'Error al crear el taller';
      },
    });
  }

  private resetForm(): void {
    this.newWorkshop = {
      workshopName: '',
      address: '',
      phone: '',
      email: '',
      fullName: '',
      description: '',
      password: '',
    };
  }
}
```

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

```
<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>
```

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

Temos o compoñente tamen para poder pedir as citas

make-appointment.component.ts M

AUTOCAREHUB > src > app > components > make-appointment > make-appointment.component.ts > MakeAppointmentComponent > ngOnInit

```typescript
import { Component, OnInit } from '@angular/core';
import { DataAccessService } from '../../services/dataAccess.service';
import { CommonModule } from '@angular/common';
import { FormsModule, ReactiveFormsModule } from '@angular/forms';
import { MenuComponent } from '../menu/menu.component';

interface WeekSlots {
  [key: string]: {
    hora: string;
    estado: string;
  }[];
}

@Component({
  selector: 'app-make-appointment',
  standalone: true,
  imports: [CommonModule, FormsModule, ReactiveFormsModule, MenuComponent],
  templateUrl: './make-appointment.component.html',
  styleUrl: './make-appointment.component.css',
})
export class MakeAppointmentComponent implements OnInit {
  selectedLang: string = 'es';
  appointmentForm: any;
  errorMessage: string = '';
  vehiclesErrorMessage: string = '';
  monthSlots: WeekSlots = {};
  loading: boolean = false;
  loadingVehicles: boolean = false;
  loadingWorkshops: boolean = false;
  selectedSlots: { fecha: string; hora: string }[] = [];
  selectedVehicle: number = 0;
  selectedWorkshop: number = 0;
  motivo: string = '';
  vehicles: any[] = [];
  workshops: any[] = [];
  isPopupVisible = false;

  // Pagination properties
  visibleDates: string[] = [];
  currentPage: number = 1;
  datesPerPage: number = 5;
  totalPages: number = 1;

  constructor(private dataAccess: DataAccessService) {}

  /**
   * Inicializa el componente cargando los datos del mes actual, los vehículos y los talleres
   */
  ngOnInit(): void {
    this.cargarVehiculos();
    this.cargarTalleres();
  }
```

```typescript
  cargarVehiculos() {
    this.loadingVehicles = true;
    this.vehiclesErrorMessage = '';

    this.dataAccess.obtenerVehiculos().subscribe({
      next: (response) => {
        if (response && response.success) {
          this.vehicles = response.vehicles || [];
          if (this.vehicles.length > 0 && !this.selectedVehicle) {
            this.selectedVehicle = this.vehicles[0].VehicleID;
          }
        } else {
          this.vehiclesErrorMessage = 'No se pudieron cargar los vehículos';
        }
      },
      error: (error) => {
        this.vehiclesErrorMessage = 'Error de conexión al cargar vehículos';
      },
      complete: () => {
        this.loadingVehicles = false;
      },
    });
  }

  /**
   * Carga los talleres disponibles
   */
  cargarTalleres() {
    this.loadingWorkshops = true;
    this.dataAccess.obtenerTalleres().subscribe({
      next: (response) => {
        if (response && response.success) {
          this.workshops = response.workshops || [];
          if (this.workshops.length > 0 && !this.selectedWorkshop) {
            this.selectedWorkshop = this.workshops[0].WorkshopID;
            this.consultarMes(); // Ahora se llama aquí, tras seleccionar taller
          }
        } else {
          this.errorMessage = 'No se pudieron cargar los talleres';
        }
      },
      error: (error) => {
        this.errorMessage = 'Error de conexión al cargar talleres';
      },
      complete: () => {
        this.loadingWorkshops = false;
      },
    });
  }
```

```
  /**
   * Configura la paginación inicial basada en las fechas disponibles
   */
  setupPagination() {
    const allDates = Object.keys(this.monthSlots).sort();
    this.totalPages = Math.ceil(allDates.length / this.datesPerPage);
    this.goToPage(1);
  }

  /**
   * Navega a una página específica de la paginación
   * @param page Número de página a la que se desea navegar
   */
  goToPage(page: number) {
    if (page < 1 || page > this.totalPages) return;

    this.currentPage = page;
    const allDates = Object.keys(this.monthSlots).sort();
    const startIndex = (page - 1) * this.datesPerPage;
    this.visibleDates = allDates.slice(
      startIndex,
      startIndex + this.datesPerPage
    );
  }

  /**
   * Navega a la página anterior
   */
  prevPage() {
    this.goToPage(this.currentPage - 1);
  }

  /**
   * Navega a la página siguiente
   */
  nextPage() {
    this.goToPage(this.currentPage + 1);
  }

  /**
   * Alterna la selección de un horario específico
   * @param fecha Fecha del horario
   * @param hora Hora del horario
   */
  toggleSlotSelection(fecha: string, hora: string) {
    const index = this.selectedSlots.findIndex(
      (slot) => slot.fecha === fecha && slot.hora === hora
    );
    if (index > -1) {
      this.selectedSlots.splice(index, 1);
    } else {
      this.selectedSlots.push({ fecha, hora });
    }
  }
```

Aqui o que temos e tanto a parte de cargar os datos nada mais se acceda ao compoñente, como todas as propiedades e funcións, temos as funcións para establecer a paxinación para as citas, xa que non ten moito sentido, en canto a experiencia de usuario, que teña que deslizar para poder acceder a última cita, e mellor para o usuario ter a paxinación establecida

Este e o servicio para acceso a datos

dataAccess.service.ts

AUTOCAREHUB > src > app > services > dataAccess.service.ts > Workshop

```typescript
import { Injectable } from '@angular/core';
import { HttpClient, HttpHeaders } from '@angular/common/http';
import { Observable, throwError } from 'rxjs';
import { catchError, map } from 'rxjs/operators';

interface LoginResponse {
  success: boolean;
  message?: string;
  user?: {
    id: number;
    name: string;
    email: string;
    role: string;  // Añadir campo de rol para verificar si es Taller o Usuario
  };
}

interface RegisterResponse {
  success: boolean;
  message?: string;
  errors?: string[];
  user?: {
    id: number;
    name: string;
    email: string;
  };
}

export interface InvoiceItem {
  ItemID: number;
  Description: string;
  Quantity: number;
  UnitPrice: number;
  TaxRate: number;
  Amount: number;
}

export interface Invoice {
  InvoiceID: number;
  AppointmentID: number;
  Date: string;
  TotalAmount: number;
  Estado: string;
  UserName?: string; // Solo disponible para modo taller
  Marca?: string;
  Modelo?: string;
  Anyo: string;
  items: InvoiceItem[];
  WorkshopName?: string; // Solo disponible para modo usuario
  WorkshopAddress?: string; // Solo disponible para modo usuario
  WorkshopPhone?: string; // Solo disponible para modo usuario
}

export interface InvoiceStats {
  total_facturas: number;
  total_facturado: number;
  promedio_factura: number;
  pendientes: number;
  pagadas: number;
}

export interface Chat {
  ChatID: number;
  UserID: number;
  WorkshopID: number;
  LastMessage: string;
  Status: 'Active' | 'Archived';
  CreateAt: string;
  WorkshopName?: string;
  UserName?: string;
  unreadCount: number;
}

export interface Message {
  MessageID: number;
  ChatID: number;
  SenderID: number;
  Message: string;
  IsRead: boolean;
  CreateAt: string;
  SenderName: string;
}

export interface Workshop {
  WorkshopID?: number;
  UserID?: number;
  Name: string;
  Address: string;
  Phone: string;
  Description?: string;
  Email?: string;
  FullName?: string;
}
```

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

```
  };

  constructor(private http: HttpClient) {}

  /**
   * Autentica al usuario mediante sus credenciales
   * @param email - Correo electrónico del usuario
   * @param password - Contraseña del usuario
   * @returns Observable con la respuesta del login
   */
  checkUserAccount(email: string, password: string): Observable<LoginResponse> {
    return this.http.post<LoginResponse>(
      `${this.apiUrl}/login.php`,
      { email, password },
      {
        headers: new HttpHeaders({ 'Content-Type': 'application/json' }),
        withCredentials: true
      }
    ).pipe(
      map(response => {
        if (response.success) {
          localStorage.setItem('currentUser', JSON.stringify(response.user));
        }
        return response;
      }),
      catchError(error => {
        console.error('Error en login:', error);
        throw error;
      })
    );
  }


  /**
   * Registra un nuevo usuario en el sistema
   * @param email - Correo electrónico del nuevo usuario
   * @param fullName - Nombre completo
   * @param password - Contraseña
   * @param notificationType - Tipo de notificación preferida
   * @param contactValue - Valor de contacto según el tipo de notificación
   * @returns Observable con la respuesta del registro
   */
  registerUser(email: string, fullName: string, password: string,
    notificationType: string, contactValue: string): Observable<RegisterResponse> {
    return this.http.post<RegisterResponse>(
      `${this.apiUrl}/register.php`,
      { email, fullName, password, notificationType, contactValue },
      this.httpOptions
    ).pipe(
      map(response => {
        if (response.success) {
          localStorage.setItem('currentUser', JSON.stringify(response.user));
        }
        return response;
      }),
      catchError(error => {
        console.error('Error en registro:', error);
        throw error;
      })
    );
  }

  /**
   * Obtiene las facturas del usuario o taller actual
   * @returns Observable con la lista de facturas
   */
  obtenerFacturas(): Observable<InvoiceResponse> {
    const currentUser = this.getCurrentUser();
    if (!currentUser) {
      return throwError(() => new Error('Usuario no autenticado'));
    }

    return this.http.get<InvoiceResponse>(`${this.apiUrl}/Invoices.php`, this.httpOptions).pipe(
      map(response => response),
      catchError(error => {
        console.error('Error al obtener las facturas:', error);
        if (error.status === 401) {
          return throwError(() => new Error('Sesión expirada o no válida. Por favor, inicia sesión nuevamente.'));
        }
        return throwError(() => new Error('Error al obtener las facturas'));
      })
    );
  }
```

```typescript
  /**
   * Crea una nueva factura
   * @param appointmentId - ID de la cita asociada
   * @param items - Array de items de la factura
   * @param estado - Estado de la factura (por defecto 'Pendiente')
   * @returns Observable con la respuesta de la creación
   */
  crearFactura(appointmentId: number, items: any[], estado: string = 'Pendiente'): Observable<any> {
    const currentUser = this.getCurrentUser();
    if (!currentUser) {
      return throwError(() => new Error('Usuario no autenticado'));
    }

    const body = {
      appointment_id: appointmentId,
      estado: estado,
      items: items
    };

    return this.http.post<any>(`${this.apiUrl}/Invoices.php`, body, this.httpOptions).pipe(
      map(response => response),
      catchError(error => {
        console.error('Error al crear la factura:', error);
        return throwError(() => new Error('Error al crear la factura'));
      })
    );
  }

  /**
   * Consulta los horarios disponibles de un taller
   * @param workshopID - ID del taller
   * @param fechaInicio - Fecha de inicio de la consulta
   * @param fechaFin - Fecha fin de la consulta
   * @returns Observable con los horarios disponibles
   */
  consultarSemana(workshopID: number, fechaInicio: string, fechaFin: string): Observable<any> {
    const body = {
      accion: 'consultar_semana',
      WorkshopID: workshopID,
      FechaInicio: fechaInicio,
      FechaFin: fechaFin
    };

    return this.http.post<any>(`${this.apiUrl}/appointments.php`, body, {
      ...this.httpOptions,
      withCredentials: true
    }).pipe(
      map(response => response),
      catchError(error => {
        console.error('Error al consultar huecos de la semana:', error);
        throw error;
      })
    );
  }
```

```
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