

**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	20/04/2025	
V1	25/05/2025	
V2	12/06/2025	

**Índice:**

<b>1.Objectivo.....</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 aos que levo eu o meu coche, son moi pouco eficientes en canto a organización de citas, no envío e xeración 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 poida solicitar e visualizar de forma sinxela as citas e as facturas.
- Tamén pretende axilizar y alixeirar a carga de traballo, para todos os talleres que sexan demandantes de esta aplicación. Por experiencia propia, tanto nas prácticas como noutros traballos que tiven, teño comprobado que atender constantemente correos e chamadas fai perder moito tempo da xornada laboral, restando dedicación ao que realmente é máis importante.

## 2. Descrición

AutocareHUB, es una aplicación que consta de 3 partes:

- A primeira máis enfocada a parte de un usuario, que ten o apartado de poder xestionar as citas, poder xestionar as facturas, e tamén dispoñer dun chat de comunicación entre o usuario e o taller.
- A segunda está destinada á xestión interna dos talleres. Nela, o persoal do taller poderá visualizar e organizar as citas, emitir e consultar facturas, así como comunicarse cos seus clientes mediante o chat.
- 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 da aplicación

Se un usuario non dispón dunha conta na aplicación, deberá acceder ao formulario de rexistro para crear unha e, posteriormente, iniciar sesión para poder utilizar os servizos dispoñibles.

LOGO  
  
50px x 50px

Español ▼

### REGISTRO

Correo Electrónico

Nombre completo

Contraseña

Confirmar contraseña

Método de notificación

Contacto

Registrarse

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

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

LOGO  
  
50px x 50px

Español ▼

### Iniciar Sesión

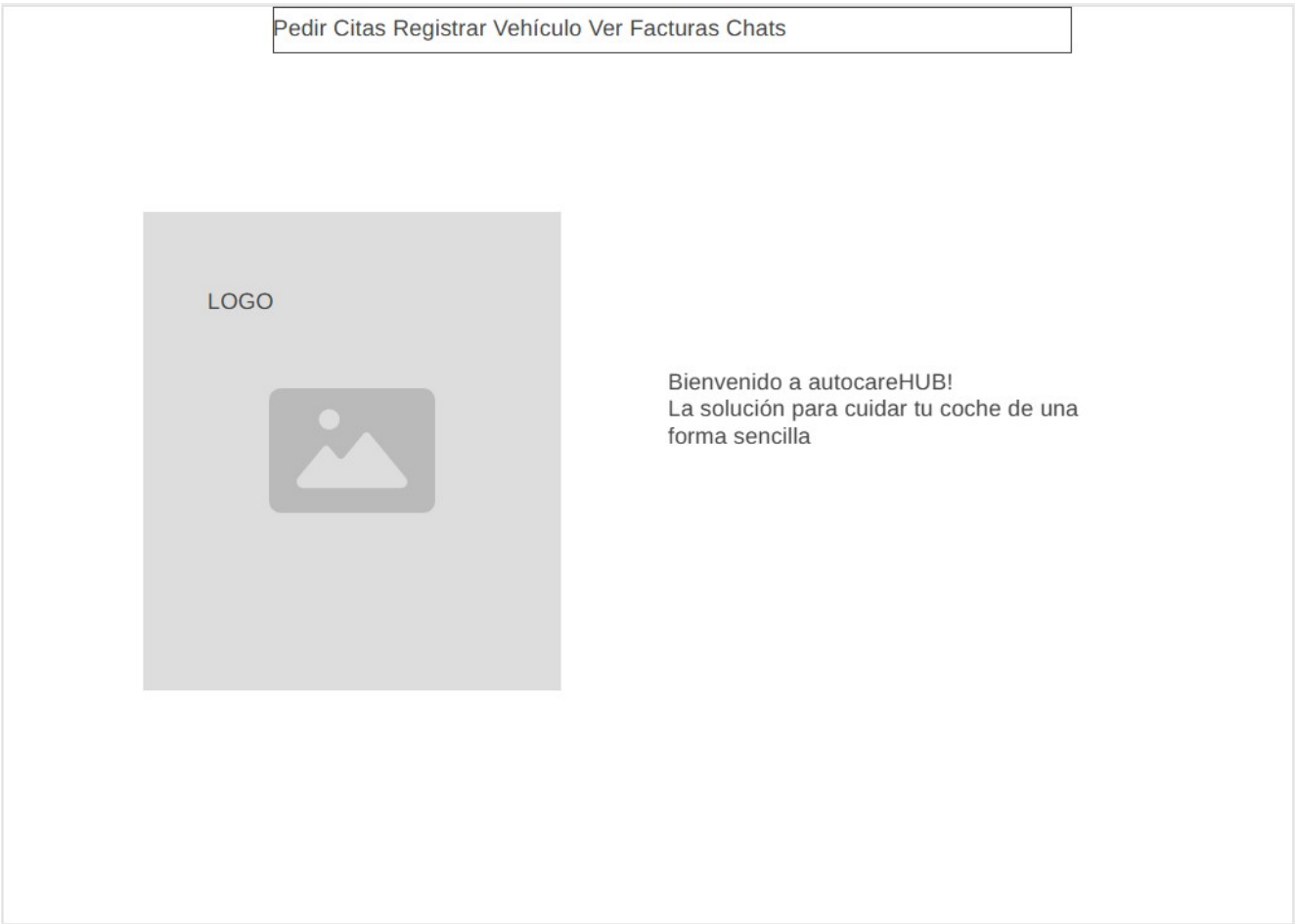
Correo Electrónico

Contraseña

Iniciar Sesión

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

Unha vez completado o proceso de inicio de sesión, o usuario poderá acceder á interface da aplicación AutocareHUB e dispoñer de todas as súas funcionalidades.



Para proceder ao rexistro dun vehículo, será necesario acceder á opción “Registrar Vehículo” e completar todos os campos requiridos no formulario correspondente.

Pedir Citas Registrar Vehículo Ver Facturas Chats

Registrar Vehículo

Marca:

Modelo:

Año:

Matrícula:

Registrar Vehículo

Vehículos Registrados

Marca- Modelo  
Año:  
Matrícula  
Editar Eliminar

Marca- Modelo  
Año:  
Matrícula  
Editar Eliminar

Despois de acceder á aplicación, poderemos consultar as facturas a través da opción “Ver facturas”, onde será posible seleccionar aquelas de interese e descargarlas en formato PDF para gardalas ou imprimilas segundo sexa necesario.

Pedir Citas Registrar Vehículo Ver Facturas Chats			
Fecha	Total	Estado	Acciones
-----	-----	-----	<div>Seleccionar Factura</div> <div>Editar PDF</div>
-----	-----	-----	<div>Seleccionar Factura</div> <div>Editar PDF</div>

Tamén teremos acceso ao apartado de chats, onde poderemos comunicarnos directamente cos talleres dispoñibles na aplicación para realizar consultas, solicitar información ou resolver dúbidas de maneira rápida e eficaz

Pedir Citas Registrar Vehículo Ver Facturas Chats	
Conversaciones	
<div>Nuevo Chat</div>	
Nombre taller	
Mensaje	
	<div>Mensaje de prueba!</div> <div>Hola! En que podemos ayudar?</div> <div><div>Escriba el mensaje</div><div>Enviar</div></div>

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

Na vista correspondente, atoparemos a opción de xerar facturas para as citas solicitadas polos usuarios, permitindo aos talleres xestionar os seus rexistros de servizos de maneira eficiente e organizada

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

---

### Generador de Facturas

Nueva factura

ID de la cita

Estado

Pendiente ▼

No hay items en la factura

Añadir Item

Descripción

Cantidad	Precio Unitario	Iva %
<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
----      -----      -----      -----      -----      -----
----      -----      -----      -----      -----      -----

Na segunda vista dispoñible, os usuarios do taller poderán consultar as citas asignadas, visualizando a data, hora e estado de cada unha, así como información relevante sobre o cliente.

Generar Factura

Ver Citas

Estadísticas

Chats

Citas del taller

Actualizar Citas

Usuario	Vehículo	Fecha Inicio	Fecha Fin	Descripción	Estado
-----	-----	-----	-----	-----	-----
-----	-----	-----	-----	-----	-----



Na vista estatística do taller, poderemos consultar datos clave sobre a facturación, como o número total de facturas emitidas, o importe global facturado e a media de cada factura. Ademais, visualizarase a cantidade de facturas cobradas e aquelas que aínda están pendentes, facilitando unha xestión eficiente da contabilidade e dos ingresos do taller.

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
-----	-----	-----	-----
-----	-----	-----	-----

Na sección de chat, os talleres poderán interactuar directamente cos usuarios que soliciten información sobre os seus servizos. Este espazo facilita unha comunicación áxil e personalizada, permitindo responder dúbidas, aclarar detalles sobre as citas e mellorar a atención ao cliente.

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, este tipo de situación adoita ser frecuente.

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, pode perder clientes e ter dificultades para manterse en funcionamento

Tamén se podería desenvolver a funcionalidade de citas recorrentes, permitindo aos usuarios programar revisións cada 2, 6 meses ou segundo necesiten, evitando así ter que xestionalas manualmente. Esta opción sería especialmente útil para aqueles que usan o coche ocasionalmente, pero precisan cambiar o aceite por tempo en lugar de por quilómetros.

Ademais, poderíase implementar un sistema de recordatorio para o cambio de aceite. Se un usuario realizou este servizo nun taller da aplicación, estimaríase un período adecuado para a próxima substitución e, antes de que se cumpra, enviaráselle unha notificación para lembrarlle a revisión.

Esta funcionalidade resulta fundamental para mellorar a experiencia do usuario e optimizar a xestión das reparacións no taller. A implementación dun sistema de notificacións permitiría informar o usuario en tempo real sobre o estado do seu vehículo, avisándoo cando a reparación estea en proceso, cando o coche estea listo para ser recollido e cando se xere a factura da reparación.

Con este sistema, os usuarios non terían que estar pendentes de consultar manualmente a aplicación, garantindo unha comunicación máis fluída e eficiente entre o taller e os clientes. Ademais, este tipo de recordatorios axudaría a reducir esquecementos ou atrasos na recollida do vehículo, facilitando unha mellor organización dos servizos.

### 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.

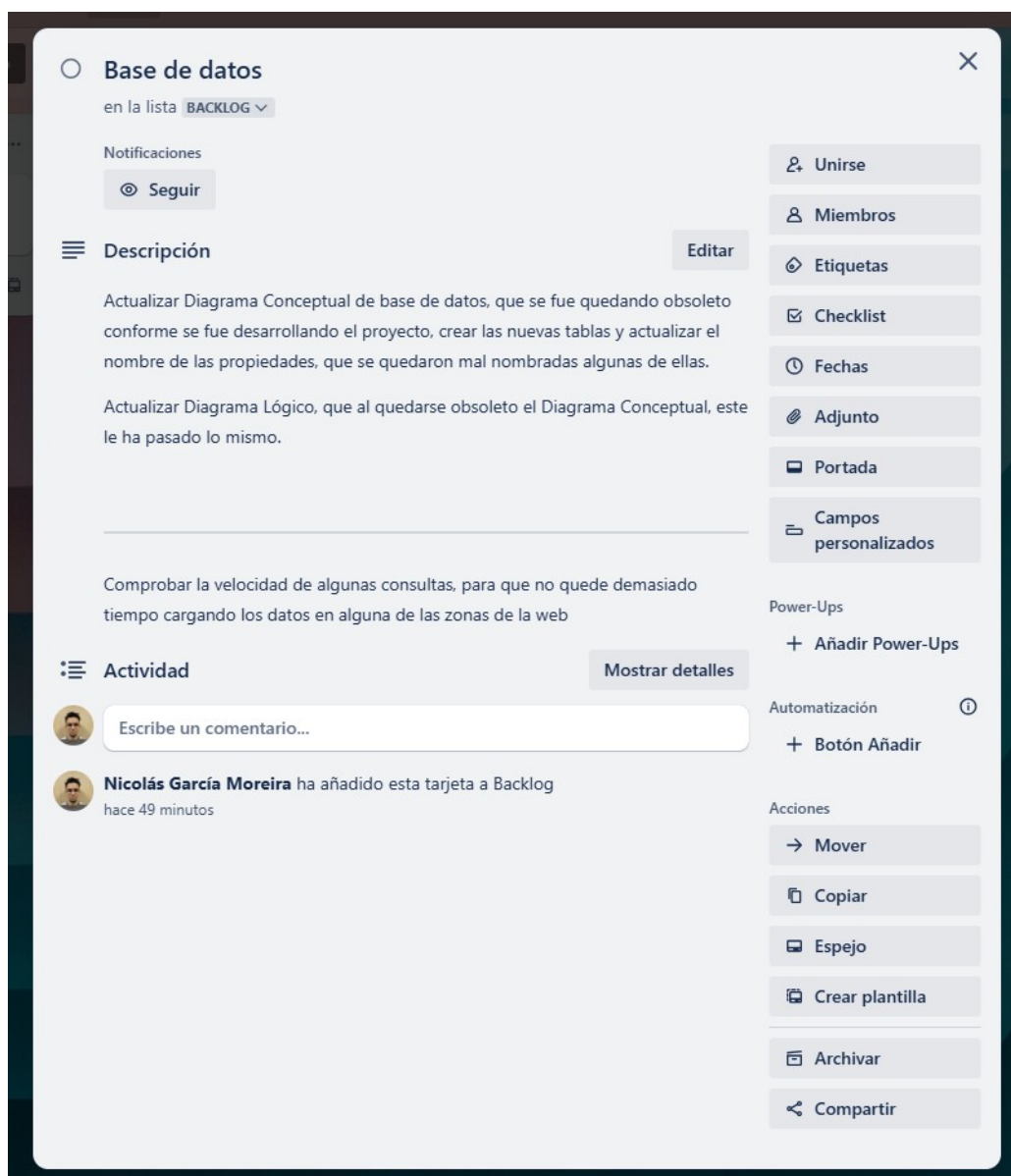
Aínda que Trello non ten por defecto conceptos como "epics" ou "sprints" como Jira, é posible estruturar estes elementos mediante listas e etiquetas. Por exemplo, un sprint pode representarse como unha lista específica no taboleiro, onde se agrupan todas as tarefas planificadas para ese período. Tamén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

Ademais, Trello conta con Power-Ups (extensiones 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.

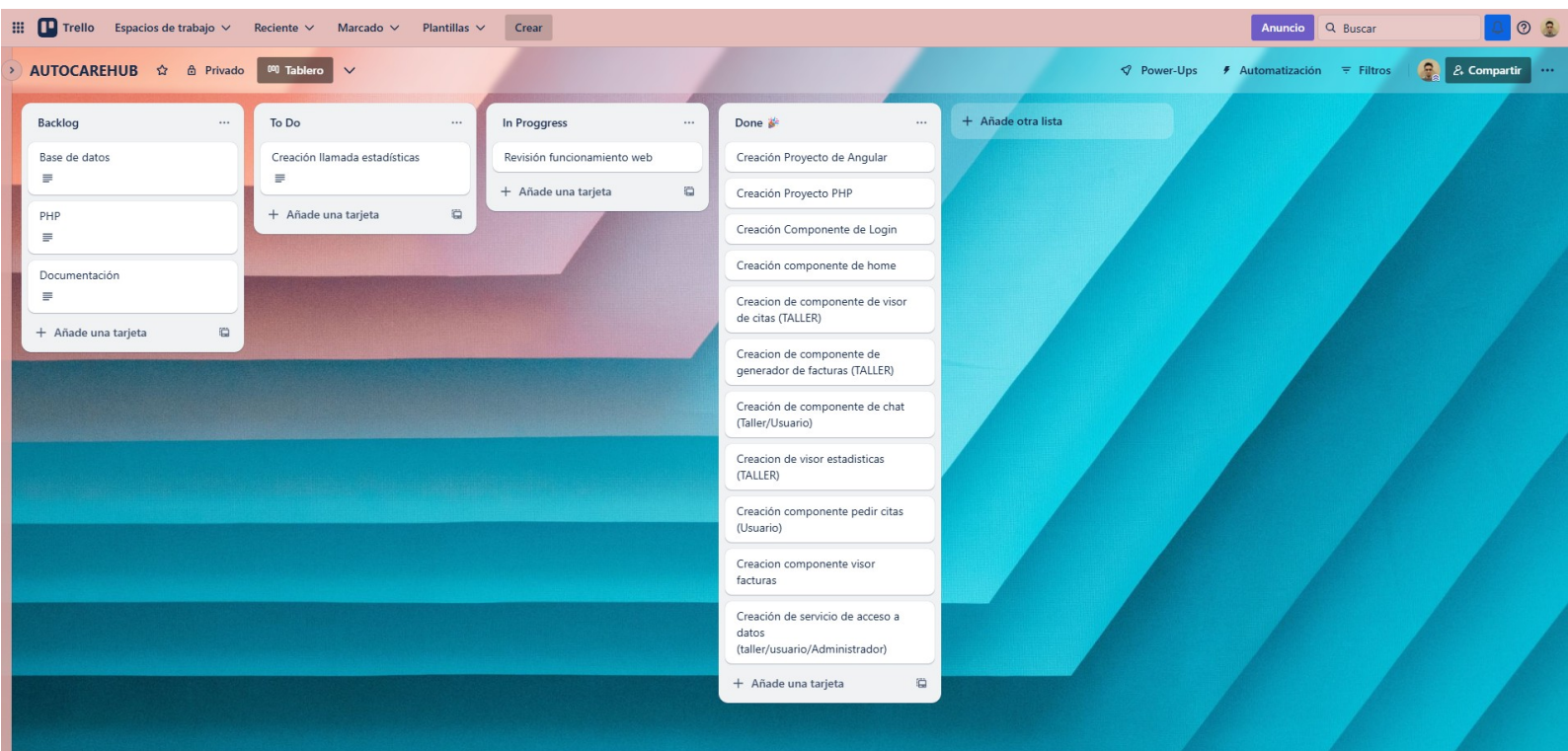
Unha funcionalidade clave é a xestión dinámica do fluxo de traballo, xa que as tarxetas poden moverse facilmente entre listas para reflectir o progreso real, como "Pendente", "En desenvolvemento", "En revisión" ou "Bloqueada". Isto ofrece unha visión clara e actualizada do estado do proxecto para todos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

Esto e o que sería unha das tarxetas de Trello



E este sería o panel de Trello, onde están todas as tarefas organizadas



\*APUNTE: Como non se ve correcta a imaxe, esta subida ([enlace](#))

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 interface 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 o desenvolvemento do código utilizarase VS Code, con algunha extensión para formatear o 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 18.2.19, 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 persoa, de unha duración de 2 meses e medio.

## 6. Presupuesto

O presuposto do proxecto é:

Man de Obra		
Duración do desenvolvemento	2 meses y medio(10 semanas)	
Horas Semanais	20	
Total de horas adicadas	200	
Prezo Hora	12,5€	
<b>Total</b>	<b>2500€</b>	
Software utilizado		
Ferramenta	Finalidade da mesma	Prezo
Dia	Desenvolvemento da base de datos, deseño conceptual e lóxico	0
Wireframe CC	Deseño da interface	0
VS Code	Editor de código para o desenvolvemento da aplicación	0
Total		0€

Planificación y elaboración de documentación		
Tarefa	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
<b>TOTAL PRESUPOSTO</b>		<b>3000€</b>

## 7. Título

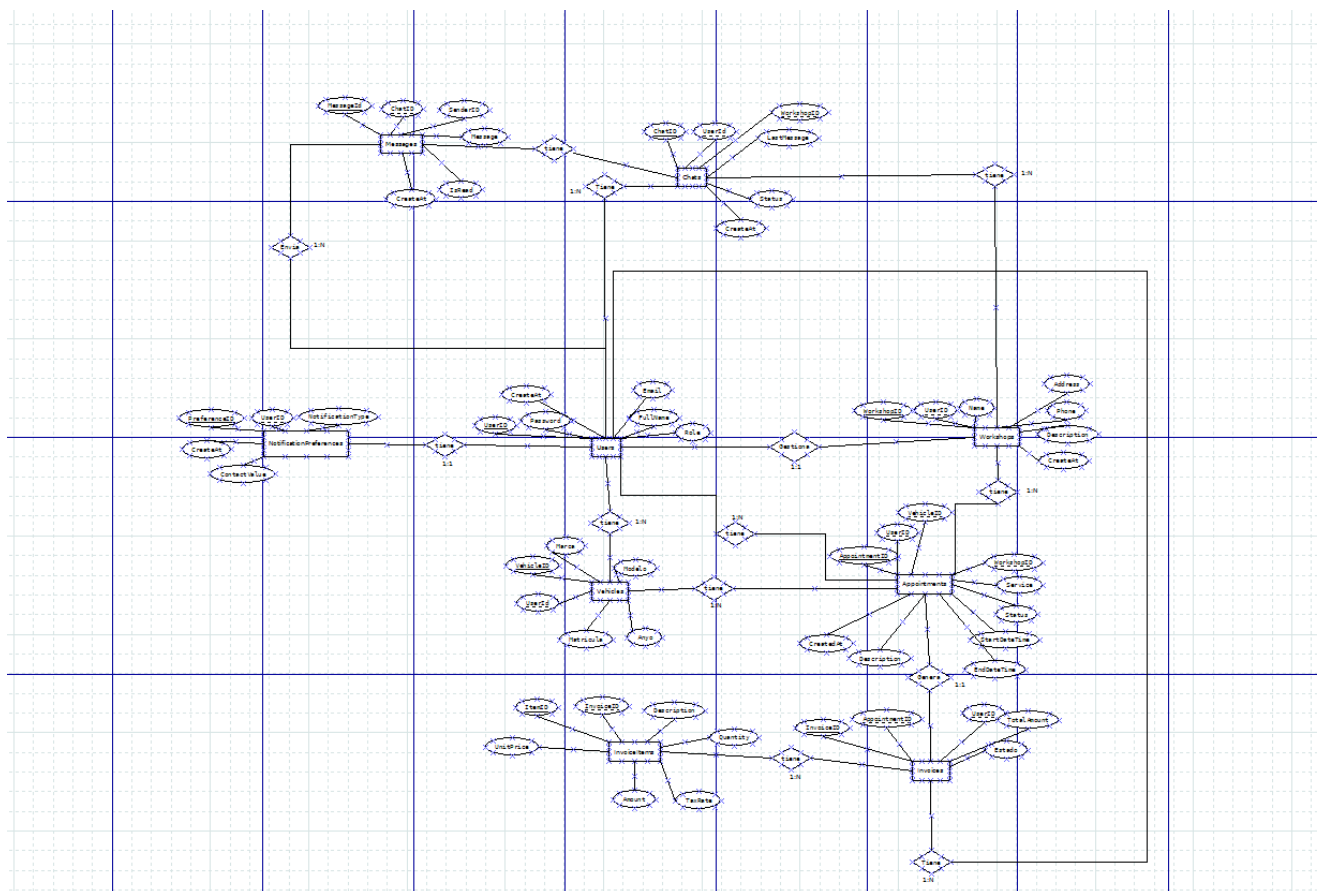
Autocare HUB

## 8. Execución

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

## 1.Diseño de base de datos

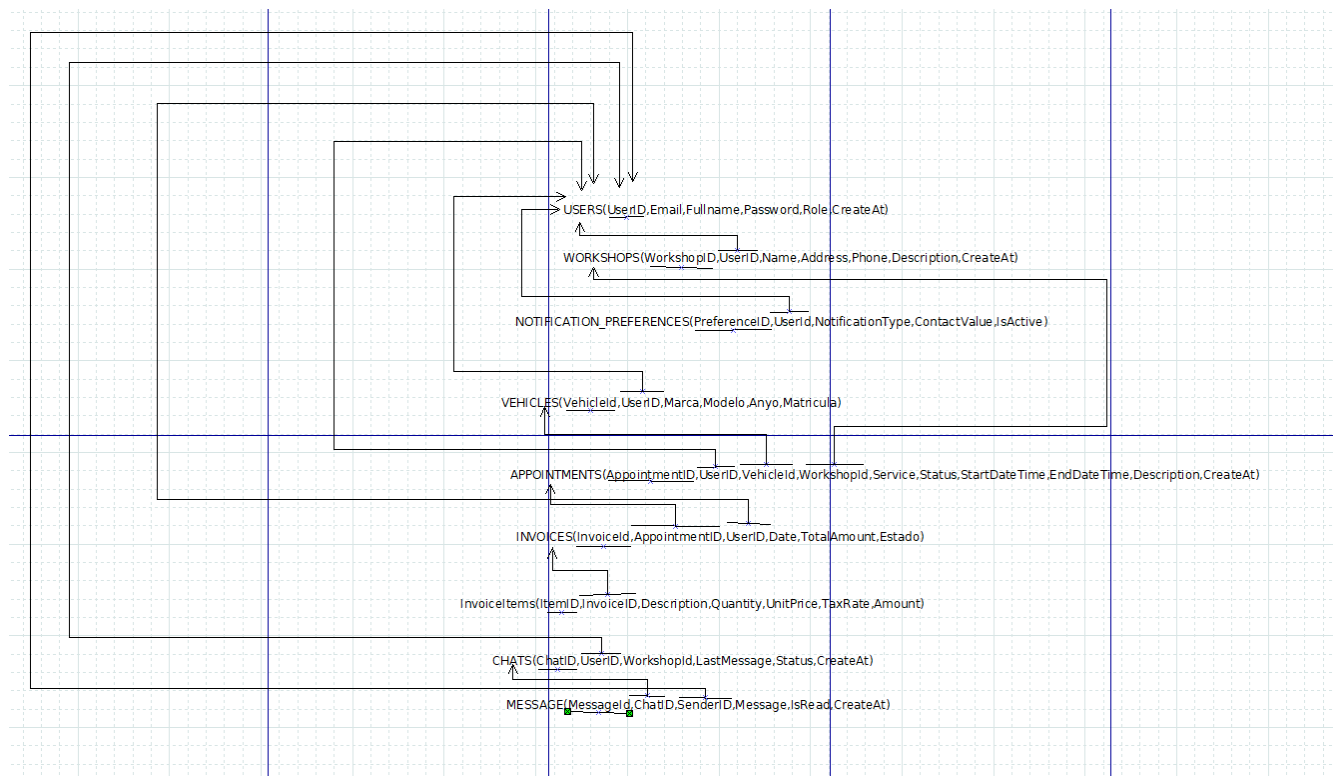
*Esquema Conceptual:*



\*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) [[ENLACE](#)]



## Esquema lóxico:



\*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) [[ENLACE](#)]

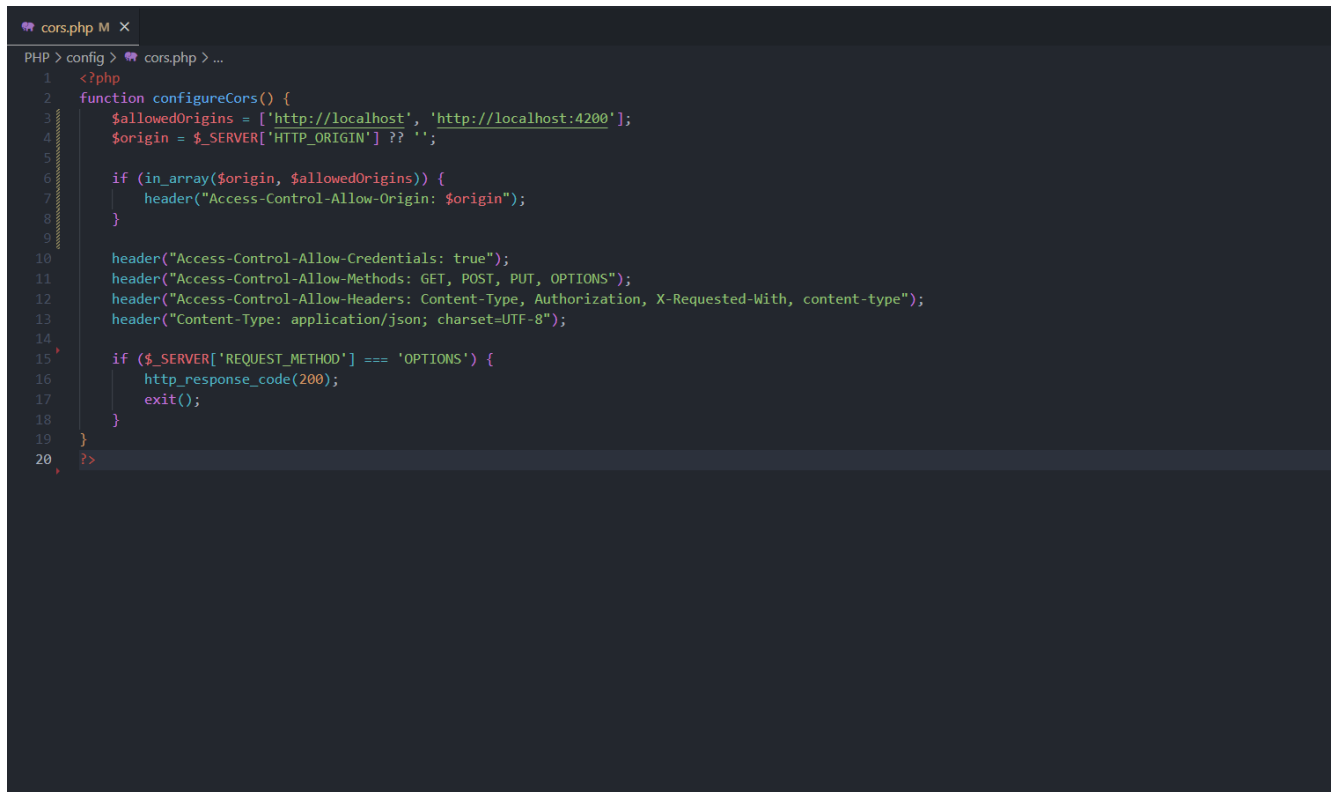
## Detalle das restricións da base de datos:

- Se un **usuario** é eliminado, elimínanse tamén todos os seus datos relacionados: os seus vehículos, citas, preferencias de notificación, facturas, 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 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 Orixen Cruzado" 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.



```
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 añadir-lle unhas cabeceiras, a todas as solicitudes, para solucionar o problema anterior mencionado.

Unha vez estas cabeceiras estean 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 contrasinal, teremos que modificalo en este ficheiro

Para controlar a autenticación do usuario para o 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 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 contrasinal incorrecta.

Tamén está a verificar que o correo electrónico que se está a introducir é un correo que cumpre os requisitos.

En caso de que o correo electrónico 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 ficheiro 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     } catch (Exception $e) {
101         return $this->response(500, $e->getMessage());
102     }
103 }
104
105 // Método para enviar una respuesta HTTP
106 private function response($code, $message, $data = []) {
107     http_response_code($code); // Establecer el código de respuesta HTTP
108     echo json_encode(array_merge([
109         "success" => $code < 400,
110         "message" => $message
111     ], $data ? ["user" => $data] : []));
112 }
```

No controlador de autenticación tamén temos a xestión de rexistro.

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 correo electrónico, como que o nome do usuario polo menos teña 3 caracteres, que a contrasinal 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, creárase 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 correo electrónico, 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 $result->fetch_assoc();
13     }
14 }
15
16 public function create($email, $fullName, $password, $notificationType, $contactValue) {
17     try {
18         $this->conn->begin_transaction();
19
20         // Verificar si el email existe
21         if ($this->emailExists($email)) {
22             throw new Exception("El email ya está registrado");
23         }
24
25         // Hash de la contraseña
26         $passwordHash = password_hash($password, PASSWORD_DEFAULT);
27
28         // Insertar usuario
29         $stmt = $this->conn->prepare(
30             "INSERT INTO {$this->table} (Email, FullName, Password) VALUES (?, ?, ?)"
31         );
32         $stmt->bind_param("sss", $email, $fullName, $passwordHash);
33
34         if (!$stmt->execute()) {
35             throw new Exception("Error al registrar el usuario");
36         }
37
38         $userId = $stmt->insert_id;
39
40         // Insertar preferencia de notificación
41         $stmtPref = $this->conn->prepare(
42             "INSERT INTO NotificationPreferences (UserID, NotificationType, ContactValue, IsActive)
43             VALUES (?, ?, ?, true)"
44         );
45         $stmtPref->bind_param("iss", $userId, $notificationType, $contactValue);
46
47         if (!$stmtPref->execute()) {
48             ...
49         }
50     }
51 }

```

Aquí temos o código para a inserción do usuario dentro da base de datos, temos tamén a verificación de si o usuario existe dentro da base de datos, para que non se poda volver a inserir, tamén temos o de password\_hash(), que o que fai e que a contrasinal como e un campo seguro, lle faga un hash: e un algoritmo matemático, que converte 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 ficheiro da ruta de rexistro, o que temos é que simplemente a configuración de CORS, e a verificación de que o método co que se está facendo a petición é correcta, e despois de esa verificación o que se fai é 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 está a verificarse é que o método co que se está facendo a petición é correcta, e tamén se está a realizar o login propiamente dito.

Tamén está 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 sesión foi correcto dende aquí, xa que podería fallar por varias cousas, polo que se está a facer dende o controlador.

```
register.php chats.php Invoices.php Vehicles.php appointments.php workshops.php Logger.php X
config > Logger.php
1 <?php
2 class Logger {
3     private static $logFile;
4     private static $instance = null;
5     private static $logTypes = ['INFO', 'WARNING', 'ERROR', 'AUTH', 'DATABASE', 'API'];
6
7     private function __construct() {
8         // Constructor privado para singleton
9     }
10
11     public static function getInstance() {
12         if (self::$instance === null) {
13             self::$instance = new self();
14             self::init();
15         }
16         return self::$instance;
17     }
18
19     public static function init($type = 'app') {
20         $logDir = __DIR__ . '/../logs';
21         if (!file_exists($logDir)) {
22             mkdir($logDir, 0777, true);
23         }
24         self::$logFile = $logDir . '/' . $type . '_' . date('Y-m-d') . '.log';
25     }
26
27     public static function log($message, $type = 'INFO', $additionalData = []) {
28         self::getInstance();
29
30         $timestamp = date('Y-m-d H:i:s');
```

```
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 $stmt->get_result()->num_rows > 0;
70 }
71
72 // Método para obtener slots semanales
73 public function getWeeklySlots($workshopId, $startDate, $endDate) {
74     $query = "SELECT StartDateTime
75               FROM " . $this->table . "
76               WHERE WorkshopID = ?
77               AND DATE(StartDateTime) BETWEEN ? AND ?
78               AND DAYOFWEEK(StartDateTime) BETWEEN 2 AND 6";
79
80     $stmt = $this->conn->prepare($query);
81     $stmt->bind_param("iss", $workshopId, $startDate, $endDate);
82     $stmt->execute();
83
84     return $stmt->get_result();
85 }
86

```

Implementouse tamén un Logger, que o que fai e que escribe as excepcións e información en un ficheiro, para facerlle ao usuario que esta a desenvolver a aplicación, mais fácil a busca de erros e de problemas que podan ocorrer.

```

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 método 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 descrición en caso de que a descrición 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()
    ]);
}
```

Aquí 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, están 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 márcalos 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;
}

```

Aquí 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($item['description']) ||
35                     !isset($item['quantity']) ||
36                     !isset($item['unit_price']) ||
37                     !isset($item['tax_rate'])) {
38                     throw new Exception("Faltan campos requeridos en los items de la factura");
39                 }
40
41                 return [
42                     'description' => $item['description'],
43                     'quantity' => floatval($item['quantity']),
44                     'unit_price' => floatval($item['unit_price']),
45                     'tax_rate' => floatval($item['tax_rate']),
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                 ];
50             }, $data['items']);
51             $this->model->TotalAmount = $this->calcularTotal($data['items']);
52
53             if ($this->model->create()) {
54                 return $this->sendResponse(200, true,
55                     "Factura creada correctamente",
56                     ['invoice_id' => $this->model->InvoiceID]);
57             }
58
59         } catch (Exception $e) {
60             return $this->sendResponse(500, false, $e->getMessage());
61         }
62     }
63 }

```

Aquí 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 estatísticas, 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           // Cargar facturas del periodo
65           this.dataAccess
66             .buscarFacturas({
67               startDate: this.startDate,
68               endDate: this.endDate,
69             })
70             .subscribe({
71               next: (response) => {
72                 this.invoices = response.invoices;
73               },
74               error: (error) => {
75                 this.error = 'Error al cargar facturas';
76               },
77             });
78         },
79       });
80
81   }
82
83   onChangeDate() {
84     this.loadData();
85   }
86
87   formatCurrency(amount: number): string {
88     return new Intl.NumberFormat('es-ES', {
89       style: 'currency',
90       currency: 'EUR',
91     }).format(amount);
92   }
93
94   formatDate(date: string): string {
95     return new Date(date).toLocaleDateString('es-ES');
96   }
97
98   getPendingPercentage(): number {
99     if (!this.stats) return 0;
100     return (this.stats.pendientes / this.stats.total_facturas) * 100;
101   }
102 }
```

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
18  <div *ngIf="loading" class="loading">
19    Cargando estadísticas...
20  </div>
21
22  <div *ngIf="stats && !loading" class="stats-grid">
23    <div class="stat-card">
24      <h3>Total Facturas</h3>
25      <p>{{ stats.total_facturas }}</p>
26    </div>
27    <div class="stat-card">
28      <h3>Total Facturado</h3>
29      <p>{{ formatCurrency(stats.total_facturado) }}</p>
30    </div>
31    <div class="stat-card">
32      <h3>Promedio por Factura</h3>
33      <p>{{ formatCurrency(stats.promedio_factura) }}</p>
34    </div>
35    <div class="stat-card">
36      <h3>Estado Facturas</h3>
37      <p>Pendientes: {{ stats.pendientes }}</p>
38      <p>Pagadas: {{ stats.pagadas }}</p>
39      <div class="progress-bar">
40        <div class="progress" [style.width.%]="getPendingPercentage()"></div>
41      </div>
42    </div>
43  </div>
44
45  <div *ngIf="invoices.length > 0" class="invoices-table">
46    <h3>Facturas del Período</h3>
47    <table>
48      <thead>
49        <tr>
50          <th>Fecha</th>
51          <th>Importe</th>
52          <th>Estado</th>
53          <th>Cliente/Taller</th>
54        </tr>
55      </thead>
56      <tbody>
57        <tr *ngFor="let invoice of invoices">
58          <td>{{ formatDate(invoice.Date) }}</td>
59          <td>{{ formatCurrency(invoice.TotalAmount) }}</td>
60          <td>{{ invoice.Estado }}</td>
61          <td>{{ invoice.UserName || invoice.WorkshopName }}</td>
62        </tr>
63      </tbody>
64    </table>
65  </div>
66</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 {
69           this.errorMessage = response.message || 'Error al crear el taller';
70         }
71       },
72       error: () => {
73         this.errorMessage = 'Error al crear el taller';
74       },
75     });
76   }
77
78   private resetForm(): void {
79     this.newWorkshop = {
80       workshopName: '',
81       address: '',
82       phone: '',
83       email: '',
84       fullName: '',
85       description: '',
86       password: '',
87     };
88   }
89 }
90
```

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

```
<!-- Componente -->
<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 tamén visualizar os que xa temos

Temos o compoñente tamén para poder pedir as citas

```
ts make-appointment.component.ts M X
AUTOCAREHUB > src > app > components > make-appointment > ts make-appointment.component.ts > 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 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 é o servizo para acceso a datos

```
TS dataAccess.service.ts X
AUTOCAREHUB > src > app > services > TS dataAccess.service.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
63
64
65
66
67
68
69
70
71
72 export interface Chat {
73   ChatID: number;
74   UserID: number;
75   WorkshopID: number;
76   LastMessage: string;
77   Status: 'Active' | 'Archived';
78   CreateAt: string;
79   WorkshopName?: string;
80   UserName?: string;
81   unreadCount: number;
82 }
83
84 export interface Message {
85   MessageID: number;
86   ChatID: number;
87   SenderID: number;
88   Message: string;
89   IsRead: boolean;
90   CreateAt: string;
91   SenderName: string;
92 }
93
94 export interface Workshop {
95   WorkshopID?: number;
96   UserID?: number;
97   Name: string;
98   Address: string;
99   Phone: string;
100   Description?: string;
101   Email?: string;
102   FullName?: string;
103 }
```

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 }
197 }

```



```

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

```

TS appointments-viewer.component.ts X
AUTOCAREHUB > src > app > components > appointments-viewer > TS appointments-viewer.component.ts > ...
1  import { Component } from '@angular/core';
2  import { DataAccessService } from '../../services/dataAccess.service';
3  import { CommonModule } from '@angular/common';
4  import { MenuComponent } from '../menu/menu.component';
5
6  interface Cita {
7      AppointmentID: number;
8      UserID: number;
9      Vehiculo: string;
10     WorkshopID: number;
11     Service: string;
12     Status: string;
13     StartDateTime: string;
14     EndDateTime: string;
15     Description: string;
16     UserName: string;
17 }
18
19 @Component({
20     selector: 'app-appointments-viewer',
21     standalone: true,
22     imports: [CommonModule, MenuComponent],
23     templateUrl: './appointments-viewer.component.html',
24     styleUrls: ['./appointments-viewer.component.css'],
25 })
26 export class AppointmentsViewerComponent {
27     citas: Cita[] = [];
28     errorCitas: string = '';
29     constructor(private dataAccessService: DataAccessService) {}
30
31     ngOnInit(): void {
32         this.obtenerCitasDelTaller();
33     }
34     /**Obtiene las citas del taller, pero solo en caso de ser un taller, y las añade dentro del array */
35     obtenerCitasDelTaller(): void {
36         this.dataAccessService.obtenerCitasTaller().subscribe({
37             next: (respuesta) => {
38                 if (respuesta) {
39                     this.citas = respuesta.citas || [];
40                 } else {
41                     this.citas =
42                         respuesta.message || 'No se pudieron obtener las citas del taller';
43                 }
44             },
45             error: (error) => {
46                 this.errorCitas = error.message || 'Error al obtener las citas';
47             },
48         });
49     }
50 }

```

O seguinte ficheiro é o de visor de citas creadas para os talleres, o que podemos ver é unha función que chama ao servizo de acceso a datos, para obter as citas que teñen os talleres rexistradas, só no caso de que sexa un taller.

```
appointments-viewer.component.html X
AUTOCAREHUB > src > app > components > appointments-viewer > appointments-viewer.component.html
Go to component
1 <app-menu></app-menu>
2 <div class="p-4">
3   <h2 class="text-2xl font-semibold mb-4">Citas del Taller</h2>
4
5   <div *ngIf="errorCitas" class="bg-red-100 border-l-4 border-red-500 text-red-700 p-4 mb-4">
6     {{ errorCitas }}
7   </div>
8
9
10  <button (click)="obtenerCitasDelTaller()" class="btn btn-primary btn-lg shadow-sm d-flex align-items-center justify-content-center gap-2 mb-2">
11    <i class="bi bi-arrow-clockwise me-1"></i>
12    Actualizar Citas
13  </button>
14  <table class="min-w-full border border-gray-300 rounded shadow-sm" *ngIf="citas.length > 0">
15    <thead class="bg-gray-100 text-left">
16      <tr>
17        <th class="px-4 py-2 border-b">Usuario</th>
18        <th class="px-4 py-2 border-b">Vehículo</th>
19        <th class="px-4 py-2 border-b">Fecha Inicio</th>
20        <th class="px-4 py-2 border-b">Fecha Fin</th>
21        <th class="px-4 py-2 border-b">Descripción</th>
22        <th class="px-4 py-2 border-b">Estado</th>
23      </tr>
24    </thead>
25    <tbody>
26      <tr *ngFor="let cita of citas" class="hover:bg-gray-50">
27        <td class="px-4 py-2 border-b">{{ cita.UserName }}</td>
28        <td class="px-4 py-2 border-b">{{ cita.Vehículo }}</td>
29        <td class="px-4 py-2 border-b">{{ cita.StartDateTime | date:'dd/MM/yyyy HH:mm' }}</td>
30        <td class="px-4 py-2 border-b">{{ cita.EndDateTime | date:'dd/MM/yyyy HH:mm' }}</td>
31        <td class="px-4 py-2 border-b">{{ cita.Description }}</td>
32        <td class="px-4 py-2 border-b">{{ cita.Status }}</td>
33      </tr>
34    </tbody>
35  </table>
36
37  <div *ngIf="citas.length === 0 && !errorCitas" class="mt-4 text-gray-500">
38    No hay citas registradas para este taller.
39  </div>
40 </div>
41
```

Este é o ficheiro de HTML asociado, onde se ven todas as propiedades para que o usuario de taller poda controlar quen lle vai e a que horas.

A continuación o componente de chat, para poder intercambiar e interactuar entre usuarios de taller e usuarios solicitantes de citas

```
TS chat.component.ts •
AUTOCAREHUB > src > app > components > chat > TS chat.component.ts > ChatComponent > actualizacionMensajes
1 import { Component, OnInit, OnDestroy } from '@angular/core';
2 import { CommonModule } from '@angular/common';
3 import { FormsModule } from '@angular/forms';
4 import {
5   DataAccessService,
6   Chat,
7   Message,
8 } from '../../services/dataAccess.service';
9 import { MenuComponent } from '../menu/menu.component';
10 import { interval, Subscription } from 'rxjs';
11 import { I18nService } from '../../services/i18n.service';
12
13 @Component({
14   selector: 'app-chat',
15   standalone: true,
16   imports: [CommonModule, FormsModule, MenuComponent],
17   templateUrl: './chat.component.html',
18   styleUrls: ['./chat.component.css'],
19 })
20 export class ChatComponent implements OnInit, OnDestroy {
21   chats: Chat[] = [];
22   mensajes: Message[] = [];
23   selectedChat: Chat | null = null;
24   nuevoMensaje: string = '';
25   userRole: string = '';
26   userId: number = 0;
27   mostrandoModalTalleres = false;
28   talleres: any[] = [];
29   talleresFiltrados: any[] = [];
30   busquedaTaller = '';
31   private actualizacionAutomatica: Subscription | undefined;
32   private actualizacionMensajes: Subscription | undefined;
33   constructor(
34     private dataService: DataAccessService,
35     public i18n: I18nService
36   ) {}
37
38   ngOnInit() {
39     const user = this.dataService.getCurrentUser();
40     if (user) {
41       this.userRole = user.role;
42       this.userId = user.id;
43       this.cargarChats();
44       if (this.userRole !== 'Taller') {
45         this.cargarTalleres();
46       }
47     }
48     this.iniciarActualizacionAutomatica();
```

O que se fai, e declarar varias propiedades que son necesarias para poder visualizar correctamente as mensaxes, pero a miga deste componente, son as subscricións que son as de “actualizaciónAutomatica” e “actualizaciónMensajes”

Unha subscrición sirve para poder actualizar as mensaxes todas, e a outra para só actualizar a mensaxe na que estamos dentro.

A continuación, temos o componente de visor de facturas, onde o que se fai, e como no resto de componentes, cargar uns datos iniciais, en este caso as facturas, e despois deixarlle ao usuario facer cousas, en este caso, xerar un PDF cos datos da factura.

```
TS invoice-viewer.component.ts X
AUTOCAREHUB > src > app > components > invoice-viewer > TS invoice-viewer.component.ts > ...
1  import { Component, OnInit } from '@angular/core';
2  import { jsPDF } from 'jspdf';
3  import { DataAccessService, Invoice } from '../../services/dataAccess.service';
4  import { CommonModule } from '@angular/common';
5  import { MenuComponent } from '../menu/menu.component';
6
7  @Component({
8    selector: 'app-invoice-viewer',
9    templateUrl: './invoice-viewer.component.html',
10   styleUrls: ['./invoice-viewer.component.css'],
11   standalone: true,
12   imports: [CommonModule, MenuComponent],
13 })
14 export class InvoiceViewerComponent implements OnInit {
15   invoices: Invoice[] = [];
16   selectedInvoice: Invoice | null = null;
17   errorMessage: string = '';
18   loading: boolean = false;
19
20   constructor(private dataService: DataAccessService) {}
21
22   ngOnInit(): void {
23     this.loadInvoices();
24   }
25
26   /**Carga las facturas del usuario */
27   loadInvoices(): void {
28     this.loading = true;
29     this.dataService.obtenerFacturas().subscribe(
30       (response) => {
31         this.invoices = response.invoices;
32         this.loading = false;
33       },
34       (error) => {
35         this.errorMessage = 'Error al cargar las facturas.';
36         this.loading = false;
37         console.error(error);
38       }
39     );
40   }
41
42   /**Método para seleccionar una factura*/
43   selectInvoice(invoice: Invoice): void {
44     this.selectedInvoice = invoice;
45   }
46 }
```

```

47  /**Método para generar el PDF de la factura seleccionada*/
48  generatePDF(): void {
49      if (!this.selectedInvoice) return;
50
51      const doc = new jsPDF();
52
53      // Agregar imagen de fondo
54      const imgUrl = 'img/LogoNaranjaAplicacionDecolorado.png';
55      doc.addImage(imgUrl, 'JPEG', 0, 0, 210, 297); // A4 completo
56
57      doc.setFont('helvetica', 'normal');
58      doc.setTextColor(0, 0, 0);
59
60      const marginLeft = 10;
61      const topMargin = 10;
62
63      // Título y datos de factura
64      doc.text('Factura', marginLeft, topMargin);
65      doc.text(
66          `Número de factura: ${this.selectedInvoice.InvoiceID}`,
67          marginLeft,
68          topMargin + 10
69      );
70      doc.text(`Fecha: ${this.selectedInvoice.Date}`, marginLeft, topMargin + 20);
71      doc.text(
72          `Estado: ${this.selectedInvoice.Estado}`,
73          marginLeft,
74          topMargin + 30
75      );
76
77      // Datos del taller (arriba a la derecha)
78      const tallerX = 130;
79      const tallerY = topMargin;
80      doc.text('Taller:', tallerX, tallerY);
81      doc.text(`${this.selectedInvoice.WorkshopName}`, tallerX, tallerY + 10);
82      doc.text(`${this.selectedInvoice.WorkshopAddress}`, tallerX, tallerY + 20);
83      doc.text(
84          `Tel: ${this.selectedInvoice.WorkshopPhone}`,
85          tallerX,
86          tallerY + 30
87      );
88
89      // Datos del vehículo
90      doc.text('Detalles del vehículo:', marginLeft, topMargin + 50);
91      doc.text(
92          `Marca: ${this.selectedInvoice.Marca}`,
93          marginLeft,
94          topMargin + 60
95      );
96      doc.text(
97          `Modelo: ${this.selectedInvoice.Modelo}`,
98          marginLeft,
99          topMargin + 70
100     );

```

O complicado de este componente, e que se ten que ir deseñando con cordeadas a localización de todos os elementos que queremos ter presentes no PDF, non e que sea moi complicado o de facer unha 1º versión, pero ir axustando tamaños e que vaia quedando ven, xa se vai ir complicando.

En este caso, temos o rexenerador de facturas, onde se pode ir creando facturas para as citas que xa ocorreron, e ir engadindo pouco a pouco os items para que despois se podan xerar correctamente as facturas.

Tamén se fai que antes de intentar recuperar as facturas, a verificación de rol de usuario, xa que esta función esta solo dispoñible en usuarios de taller

```
TS invoices-generator.component.ts X
AUTOCAREHUB > src > app > components > invoices-generator > TS invoices-generator.component.ts > ...
1 import { Component, OnInit } from '@angular/core';
2 import { FormsModule } from '@angular/forms';
3 import { CommonModule } from '@angular/common';
4 import { DataAccessService, Invoice } from '../../services/dataAccess.service';
5 import { MenuComponent } from '../menu/menu.component';
6 import { AppointmentsViewerComponent } from '../appointments-viewer/appointments-viewer.component';
7
8 interface InvoiceItemWorkshop {
9   description: string;
10  quantity: number;
11  unit_price: number;
12  tax_rate: number;
13 }
14
15 @Component({
16   selector: 'app-invoices-generator',
17   standalone: true,
18   templateUrl: './invoices-generator.component.html',
19   styleUrls: ['./invoices-generator.component.css'],
20   imports: [FormsModule, CommonModule, MenuComponent],
21 })
22 export class InvoicesGeneratorComponent implements OnInit {
23   facturas: Invoice[] = [];
24   appointment_id: number | null = null;
25   items: InvoiceItemWorkshop[] = [];
26   estado: string = 'Pendiente';
27   citasTaller: any[] = [];
28   errorCitas: string = '';
29
30   // Campos para el nuevo ítem
31   newItemDescription: string = '';
32   newItemQuantity: number = 1;
33   newItemPrice: number = 0;
34   newItemTaxRate: number = 21; // IVA predeterminado
35
36   isLoading: boolean = false;
37   errorMessage: string = '';
38   successMessage: string = '';
39   isTaller: boolean = false;
40
41   constructor(private dataAccessService: DataAccessService) {}
42
43   ngOnInit(): void {
44     this.obtenerFacturas();
45
46     // Comprobar si el usuario es un taller
47     const currentUser = this.dataAccessService.getCurrentUser();
48     if (currentUser && currentUser.role === 'Taller') {
49       this.isTaller = true;
50     } else {
51       console.warn(
52         'Este componente está diseñado para usuarios con rol de Taller'
53       );
54     }
55     if (this.isTaller) {
56       this.obtenerCitasDelTaller();
57     }
58   }
59 }
```



Aquí temos o componente de login, que ten a chamada para comprobar que os datos que o usuario pon, que sexan os correctos para poder permitir darlle paso a aplicación. Tamén verifica que se o usuario intenta iniciar sesión moitas veces, a conta bloquese e non deixa intentalo ata que pase un tempo

```
TS login.component.ts x
AUTOCAREHUB > src > app > components > login > TS login.component.ts > LoginComponent > constructor

14 @Component({
15   selector: 'app-login',
16   standalone: true,
17   imports: [CommonModule, ReactiveFormsModule, RouterModule, FormsModule],
18   templateUrl: './login.component.html',
19   styleUrls: ['./login.component.css'],
20 })
21 export class LoginComponent {
22   loginForm: FormGroup;
23   errorMessage: string = '';
24   loading: boolean = false;
25   selectedLang: string;
26
27   constructor(
28     private fb: FormBuilder,
29     private dataAccessService: DataAccessService,
30     private router: Router,
31     public i18n: I18nService
32   ) {
33     this.selectedLang = this.i18n.currentLang || 'es';
34     this.loginForm = this.fb.group({
35       email: ['', [Validators.required, Validators.email]],
36       password: ['', [Validators.required, Validators.minLength(6)]],
37     });
38   }
39
40   /** Cambia el idioma de la aplicación */
41   changeLang(lang: string) {
42     this.i18n.setLang(lang);
43     this.selectedLang = lang;
44   }
45
46   /** Comprueba si los datos del formulario son correctos */
47   checkAccount() {
48     if (this.loginForm.valid) {
49       this.loading = true;
50       this.errorMessage = '';
51
52       const email = this.loginForm.get('email')?.value;
53       const password = this.loginForm.get('password')?.value;
54
55       this.dataAccessService.checkUserAccount(email, password).subscribe({
56         next: (response) => {
57           this.loading = false;
58           if (response.success) {
59             localStorage.setItem('userName', response.user?.name || '');
60             this.router.navigate(['/home']);
61           } else {
62             this.errorMessage =
63               response.message || this.i18n.t('errorIniciarSesion');
64             this.loginForm.patchValue({ password: '' });
65           }
66         },
67         error: (error) => {
68           this.loading = false;
69           if (error.status === 429) {
70             this.errorMessage = this.i18n.t('demasiadosIntentos');
71           } else {
72             this.errorMessage = this.i18n.t('errorIniciarSesion');
73           }
74         }
75       });
76     }
77   }
78 }
```

Para poder iniciar sesión, e necesario ter o campo de correo electrónico cuberto, que teña unha arroba "@", para o campo de contrasinal, ten que ter unha lonxitude mínima de 6 caracteres.

No componente de menú, temos as diferentes rutas establecidas en diferentes ficheiros, para así poder cargar en función do rol de usuario unhas opcións ou outras.

Estas serían as rutas que temos para un usuario de conta de administrador do sistema.

```
TS admin-menu-routes.ts x
AUTOCAREHUB > src > app > components > menu > TS admin-menu-routes.ts > ...

1 export const ADMIN_MENU_ROUTES = [
2   { path: '/workshops-management', label: 'menu.workshopsManagement' },
3 ];
4
```

Estas son as rutas que teñen os usuarios con rol de talleres, o label, establece así, xa que para o servizo multiidoma fai que as propiedades estean separadas.

```
TS workshop-menu-routes.ts X
AUTOCAREHUB > src > app > components > menu > TS workshop-menu-routes.ts > ...
1 export const WORKSHOP_MENU_ROUTES = [
2   { path: '/invoiceGenerator', label: 'menu.invoiceGenerator' },
3   { path: '/viewAppointments', label: 'menu.viewAppointments' },
4   { path: '/statistics', label: 'menu.statistics' },
5   { path: '/chat', label: 'menu.chat' },
6 ];
7
```

E estas son as de un usuario solicitante tanto de facturas como de citas

```
TS user-menu-routes.ts X
AUTOCAREHUB > src > app > components > menu > TS user-menu-routes.ts > ...
1 export const USER_MENU_ROUTES = [
2   { path: '/makeAppointment', label: 'menu.makeAppointment' },
3   { path: '/registerVehicle', label: 'menu.registerVehicle' },
4   { path: '/invoiceViewer', label: 'menu.invoiceViewer' },
5   { path: '/chat', label: 'menu.chat' },
6 ];
7
```

E este e o componente, que establece as rutas anteriormente comentadas  
O que fai e que se recolle os datos que temos no localStorage, e parsea o perfil, para poder reco-  
ller o rol que ten o perfil, e cargar en función do rol as rutas.

```
TS menu.component.ts X
AUTOCAREHUB > src > app > components > menu > TS menu.component.ts > ...
1 import { Component, OnInit } from '@angular/core';
2 import { USER_MENU_ROUTES } from '../user-menu-routes';
3 import { WORKSHOP_MENU_ROUTES } from '../workshop-menu-routes';
4 import { ADMIN_MENU_ROUTES } from '../admin-menu-routes';
5 import { RouterLink } from '@angular/router';
6 import { CommonModule } from '@angular/common';
7 import { I18nService } from '../../services/i18n.service';
8
9 @Component({
10   selector: 'app-menu',
11   templateUrl: './menu.component.html',
12   styleUrls: ['./menu.component.css'],
13   standalone: true,
14   imports: [RouterLink, CommonModule],
15 })
16 export class MenuComponent implements OnInit {
17   menuRoutes: { path: string; label: string }[] = [];
18
19   constructor(public i18n: I18nService) {}
20
21   ngOnInit(): void {
22     const currentUser = localStorage.getItem('currentUser');
23
24     if (currentUser) {
25       const user = JSON.parse(currentUser);
26       const role = user.role.toLowerCase(); // "usuario", "taller" o "administrador"
27
28       switch (role) {
29         case 'taller':
30           this.menuRoutes = WORKSHOP_MENU_ROUTES;
31           break;
32         case 'administrador':
33           this.menuRoutes = ADMIN_MENU_ROUTES;
34           break;
35         default:
36           this.menuRoutes = USER_MENU_ROUTES;
37       }
38     }
39   }
40 }
41
```

No componente que temos para poder dar de alta un novo vehículo, temos a validación de que o ano do vehículo, non e superior a ao ano actual.

Tamén temos para visualizar os vehículos que xa temos dados de alta.

```
TS register-vehicle.component.ts x
AUTOCAREHUB > src > app > components > register-vehicle > TS register-vehicle.component.ts > RegisterVehicleComponent > crearVehiculo
1  import { Component, OnInit } from '@angular/core';
2  import { DataAccessService } from '../services/dataAccess.service';
3  import { FormsModule } from '@angular/forms';
4  import { CommonModule } from '@angular/common';
5  import { MenuComponent } from '../menu/menu.component';
6
7  @Component({
8    selector: 'app-register-vehicle',
9    standalone: true,
10   imports: [FormsModule, CommonModule, MenuComponent],
11   templateUrl: './register-vehicle.component.html',
12   styleUrls: ['./register-vehicle.component.css'],
13 })
14 export class RegisterVehicleComponent implements OnInit {
15   marca: string = '';
16   modelo: string = '';
17   anyo: string = '';
18   matricula: string = '';
19   mensajeError: string = '';
20   mensajeExito: string = '';
21   vehiculos: any[] = []; // Array para almacenar Los vehículos del usuario
22   vehiculoEnEdicion: any = null;
23
24   constructor(private dataAccessService: DataAccessService) {}
25
26   /**
27    * Inicializa el componente y carga Los vehículos del usuario al inicio
28    */
29   ngOnInit(): void {
30     this.obtenerVehiculos(); // Llamada a la API para obtener Los vehículos al iniciar el componente
31   }
32
33   /**
34    * Crea un nuevo vehículo con Los datos del formulario
35    * Realiza validaciones de matrícula y año antes de enviar la petición
36    * Actualiza Los mensajes de éxito o error según el resultado
37    */
38   crearVehiculo() {
39     const anyoActual = new Date().getFullYear();
40
41     // Validación de matrícula sin espacios
42     if (this.matricula.includes(' ')) {
43       this.mensajeError = 'La matrícula no puede contener espacios en blanco.';
44       this.mensajeExito = '';
45       return;
46     }
47
48     // Validación de año no superior al actual
49     const anyoNumero = parseInt(this.anyo, 10);
50     if (isNaN(anyoNumero) || anyoNumero > anyoActual) {
51       this.mensajeError = `El año del vehículo no puede ser mayor que ${anyoActual}`;
52       this.mensajeExito = '';
53       return;
54     }
55     const vehiculo = {
56       marca: this.marca,
57       modelo: this.modelo,
58       anyo: this.anyo,
59       matricula: this.matricula,
```

E este é o servizo que teño para a tradución dos textos, para que sexan multiidoma, non está implementado en toda a aplicación

```
TS i18n.service.ts X
AUTOCAREHUB > src > app > services > TS i18n.service.ts > I18nService
1  import { Injectable } from '@angular/core';
2  import { ES_TEXTS } from '../assets/es';
3  import { EN_TEXTS } from '../assets/en';
4
5  @Injectable({
6    providedIn: 'root'
7  })
8  export class I18nService {
9    public currentLang = 'es';
10
11    private translations: any = {
12      es: ES_TEXTS,
13      en: EN_TEXTS
14    };
15
16    setLang(lang: string) {
17      this.currentLang = lang;
18    }
19
20    t(key: string, vars?: { [key: string]: any }): string {
21      const keys = key.split('.');
22      let text = this.translations[this.currentLang];
23
24      // Navegar por el objeto de traducciones siguiendo la ruta de keys
25      for (const k of keys) {
26        if (text[k] === undefined) return key;
27        text = text[k];
28      }
29
30      if (vars) {
31        Object.keys(vars).forEach(k => {
32          text = text.replace(new RegExp(`{${k}}`, 'g'), vars[k]);
33        });
34      }
35      return text;
36    }
37  }
```

No servizo que temos para xestionar a internacionalización da aplicación, implementamos a funcionalidade para poder cambiar o idioma da interface en tempo real.

Tamén temos a posibilidade de obter calquera texto traducido segundo o idioma seleccionado, incluíndo a substitución de variables dinámicas dentro dos textos.

E aquí e onde temos as rutas da aplicación definidas, o que temos e o 'path' que e onde se define a ruta q se vai a visitar, e 'component' que e o componente que estamos a vincular coa ruta

```
TS app.routes.ts X
AUTOCAREHUB > src > app > TS app.routes.ts > ...
1  import { Routes } from '@angular/router';
2  import { LoginComponent } from '../components/login/login.component'; // Importa el componente de login
3  import { RegisterComponent } from '../components/register/register.component'; // Importa el componente de registro
4  import { HomeComponent } from '../components/home/home.component';
5  import { MakeAppointmentComponent } from '../components/make-appointment/make-appointment.component';
6  import { RegisterVehicleComponent } from '../components/register-vehicle/register-vehicle.component';
7  import { AppointmentsViewerComponent } from '../components/appointments-viewer/appointments-viewer.component';
8  import { InvoicesGeneratorComponent } from '../components/invoices-generator/invoices-generator.component';
9  import { InvoiceViewerComponent } from '../components/invoice-viewer/invoice-viewer.component';
10 import { StatisticsViewerComponent } from '../components/statistics-viewer/statistics-viewer.component';
11 import { ChatComponent } from '../components/chat/chat.component';
12 import { WorkshopsManagementComponent } from '../components/workshops-management/workshops-management.component';
13
14 export const routes: Routes = [
15   { path: 'login', component: LoginComponent },
16   { path: '', redirectTo: '/login', pathMatch: 'full' },
17   { path: 'register', component: RegisterComponent },
18   { path: 'home', component: HomeComponent },
19   { path: 'makeAppointment', component: MakeAppointmentComponent },
20   { path: 'registerVehicle', component: RegisterVehicleComponent },
21   { path: 'viewAppointments', component: AppointmentsViewerComponent },
22   { path: 'invoiceGenerator', component: InvoicesGeneratorComponent },
23   { path: 'invoiceViewer', component: InvoiceViewerComponent },
24   { path: 'statistics', component: StatisticsViewerComponent },
25   { path: 'chat', component: ChatComponent },
26   { path: 'workshops-management', component: WorkshopsManagementComponent },
27 ];
```