**UNIVERSIDAD TECNOLÓGICA DE AGUASCALIENTES**

INGENIERÍA EN DESARROLLO Y GESTIÓN DE SOFTWARE



Desarrollo Web Integral

Roberto Correa Bravo

Actividad 4 Parcial 2

Martín Eduardo Estrada Salas 220190

IDGS –9A

Aguascalientes, AGS 04 de Julio de 2025

Contenido

[Implementación del Caso de Estudio 3](#_Toc202575539)

[Descripción del proyecto: 3](#_Toc202575540)

[Objetivo 3](#_Toc202575541)

[Alcance 3](#_Toc202575542)

[Requerimientos Funcionales 3](#_Toc202575543)

[Requerimientos no Funcionales 6](#_Toc202575544)

[Planificación Ágil – SCRUM 8](#_Toc202575545)

[1. Product Backlog 8](#_Toc202575546)

[2. Sprint Backlog para los registros en inicios de sesión. 9](#_Toc202575547)

[3. Daily Scrum 9](#_Toc202575548)

[4. Revisión del Sprint 10](#_Toc202575549)

[5. Retrospectiva del Sprint 10](#_Toc202575550)

[6. Informe de estado del Sprint 10](#_Toc202575551)

[Diseño de Arquitectura y patrones. 12](#_Toc202575552)

[Arquitectura 12](#_Toc202575553)

[Razones 12](#_Toc202575554)

[Patrón – Aggregator 12](#_Toc202575555)

[Razones 12](#_Toc202575556)

[Desarrollo con Frameworks 13](#_Toc202575557)

[Ionic-Angular 13](#_Toc202575558)

[Ejemplo de la página de perfil: 13](#_Toc202575559)

[Backend Supabase 13](#_Toc202575560)

[Integración de Servicios Web 15](#_Toc202575561)

[Servicios 15](#_Toc202575562)

[Carrito 15](#_Toc202575563)

[Pedidos 16](#_Toc202575564)

[Productos 17](#_Toc202575565)

[Servicio del perfil 18](#_Toc202575566)

[Supabase – Registro 19](#_Toc202575567)

[LINKS 20](#_Toc202575568)

[Página 20](#_Toc202575569)

[Github 20](#_Toc202575570)

[uLebW/principal at desarrollo 20](#_Toc202575571)

# Implementación del Caso de Estudio

## Descripción del proyecto:

Una empresa que vende productos ha decido aumentar su rango de venta por lo que han aumentado la cantidad de flotillas que utilizarán para la entrega de productos, a la vez con el objetivo de tener un punto de referencia van a necesitar desarrollar una página para la venta de sus productos en su E-commerce.

## Objetivo

Superar la cantidad de ganancias generadas por los métodos ya establecidos por la empresa, y alcanzar una cantidad de 10,000 usuarios simultáneos en la página de la empresa.

## Alcance

El proyecto se enfocará en desarrollar una página web que pueda realizar ventas de productos de forma segura, además de que este desarrollada para soportar una gran cantidad de usuarios simultáneos. El proyecto estará limitado por las políticas de la empresa y de aquello que el represente exprese durante las juntas previstas.

## Requerimientos Funcionales

|  |  |
| --- | --- |
| Identificación de Req. | RF01 |
| Nombre del Req. | Compras de Productos |
| Características | Los usuarios podrán adquirir productos disponibles. |
| Descripción del Req. | El sitio permitirá a los usuarios seleccionar, añadirlos a un carrito de compras, y completar el proceso de adquisición, incluyendo el manejo de pagos. |
| Prioridad del Req. | Alta |

|  |  |
| --- | --- |
| Identificación de Req. | RF02 |
| Nombre del Req. | Comentarios de Productos |
| Características | Los usuarios podrán dejar comentarios sobre los productos adquiridos o visualizados. |
| Descripción del Req. | El sistema debe permitir a los usuarios escribir y enviar comentarios sobre productos específicos, los cuales podrán ser visibles para otros usuarios. |
| Prioridad del Req. | Media |

|  |  |
| --- | --- |
| Identificación de Req. | RF03 |
| Nombre del Req. | Valoraciones de productos |
| Características | Los usuarios podrán calificar los productos con valoraciones. |
| Descripción del Req. | El sistema permitirá a los usuarios asignar una calificación usando estrellas a los productos, contribuyendo a la reputación y visibilidad de los mismos. |
| Prioridad del Req. | Media |

|  |  |
| --- | --- |
| Identificación de Req. | RF04 |
| Nombre del Req. | Actualización en tiempo real |
| Características | La información relevante del sistema se actualizará en tiempo real. |
| Descripción del Req. | El sistema debe reflejar cambios en el stock de productos, precios y otros datos críticos de forma inmediata para todos los usuarios. |
| Prioridad del Req. | Alta |

|  |  |
| --- | --- |
| Identificación de Req. | RF05 |
| Nombre del Req. | Seguridad de transacciones |
| Características | Las transacciones financieras y de datos sensibles deben ser seguras. |
| Descripción del Req. | El sistema debe implementar protocolos de seguridad para proteger la información de pago y datos personales durante las transacciones en línea. |
| Prioridad del Req. | Alta |

|  |  |
| --- | --- |
| Identificación de Req. | RF06 |
| Nombre del Req. | Sistema de búsqueda de productos |
| Características | Los usuarios podrán buscar productos de forma eficiente. |
| Descripción del Req. | El sistema debe proveer una funcionalidad de búsqueda que permita a los usuarios encontrar productos por nombre, descripción o palabras clave relevantes. |
| Prioridad del Req. | Alta |

|  |  |
| --- | --- |
| Identificación de Req. | RF07 |
| Nombre del Req. | Filtro de productos |
| Características | Los usuarios podrán refinar sus búsquedas de productos. |
| Descripción del Req. | El sistema debe ofrecer opciones de filtrado que permitan a los usuarios limitar los resultados de búsqueda por categoría, precio, marca u otros atributos relevantes. |
| Prioridad del Req. | Alta |

|  |  |
| --- | --- |
| Identificación de Req. | RF08 |
| Nombre del Req. | Registro e inicio de sesión de Usuarios |
| Características | Los usuarios podrán crear una cuenta y acceder al sistema. |
| Descripción del Req. | El sistema debe permitir a los nuevos usuarios registrarse proporcionando la información necesaria y a los usuarios existentes iniciar sesión con sus credenciales. |
| Prioridad del Req. | Alta |

## Requerimientos no Funcionales

|  |  |
| --- | --- |
| Identificación de Req. | RNF01 |
| Nombre del Req. | Cumplimiento de Normas de seguridad de datos |
| Características | La plataforma debe proteger la privacidad y la integridad de los datos. |
| Descripción del Req. | El sistema debe adherirse a las normativas y mejores prácticas de seguridad de datos (ej. GDPR, leyes de protección de datos locales), protegiendo la información personal y sensible de los usuarios. |
| Prioridad del Req. | Alta |

|  |  |
| --- | --- |
| Identificación de Req. | RNF02 |
| Nombre del Req. | Adaptabilidad a Dispositivos (Responsiveness) |
| Características | La interfaz de usuario debe adaptarse a diferentes tamaños de pantalla. |
| Descripción del Req. | La página web debe ser completamente responsiva, garantizando una experiencia de usuario óptima y una visualización correcta en computadoras de escritorio, laptops, tablets y teléfonos móviles. |
| Prioridad del Req. | Alta |

|  |  |
| --- | --- |
| Identificación de Req. | RNF03 |
| Nombre del Req. | Disponibilidad del sistema |
| Características | El sistema debe estar operativo la mayor parte del tiempo. |
| Descripción del Req. | La página web debe tener una alta disponibilidad, aspirando a un tiempo de actividad del 99.9% para asegurar que los usuarios puedan acceder al servicio cuando lo necesiten. |
| Prioridad del Req. | Alta |

|  |  |
| --- | --- |
| Identificación de Req. | RNF04 |
| Nombre del Req. | Tiempo de carga |
| Características | La página debe cargar rápidamente para mostrar los productos al usuario. |
| Descripción del Req. | El tiempo de carga de cualquier página dentro del sistema no debe exceder los 2 segundos bajo condiciones de red y servidor normales. |
| Prioridad del Req. | Alta |

|  |  |
| --- | --- |
| Identificación de Req. | RNF05 |
| Nombre del Req. | Compatibilidad de navegadores |
| Características | La página debe funcionar correctamente en diversos navegadores. |
| Descripción del Req. | El sistema debe ser compatible y funcionar sin errores visuales o funcionales en las versiones más recientes de los navegadores web principales (ej. Chrome, Firefox, Safari, Edge). |
| Prioridad del Req. | Alta |

# Planificación Ágil – SCRUM

## 1. Product Backlog

|  |  |  |  |
| --- | --- | --- | --- |
| ID de PBI | Actividad/Característica | Descripción | Prioridad (1=Alta) |
| PB001 | Configuración Inicial y Requerimientos | Definir herramientas, entorno de desarrollo y funcionalidades clave. | 1 |
| PB002 | Diseño de las relaciones y estructura de datos de la BD | Definir llaves y estructura de los datos a resguardar en esta. | 1 |
| PB003 | Diseño UX/UI Básico | Bocetos y diseños de las páginas principales (Home, Producto, Carrito). | 1 |
| PB004 | Autenticación de Usuario | Permitir registrarse, iniciar sesión y recuperar contraseña. | 1 |
| PB005 | Visualización de Productos | Mostrar lista de productos, página de detalle de producto. | 2 |
| PB006 | Carrito de Compras Básico | Añadir/eliminar productos del carrito, actualizar cantidades. | 1 |
| PB007 | Proceso de Checkout Básico | Seleccionar dirección de envío, método de pago (simulado). | 2 |
| PB008 | Gestión de Pedidos | Ver historial de pedidos, estado del pedido. | 3 |
| PB009 | Búsqueda y Filtros | Buscar productos por nombre, filtrar por categoría. | 2 |
| PB010 | Notificaciones por Correo | Confirmación de registro, pedido, envío. | 3 |
| PB011 | Integración de Pasarela de Pago | Integrar con Stripe/PayPal (simulado al principio). | 3 |
| PB012 | Deploy en Servidor | Subir el proyecto a un servidor público. | 1 |
| PB013 | Pruebas de Seguridad | Revisar vulnerabilidades comunes. | 2 |
| PB014 | SEO Básico | Optimización para motores de búsqueda. | 4 |

## 2. Sprint Backlog para los registros en inicios de sesión.

**Objetivo del Sprint:** Tener el proyecto configurado, la base de datos inicial lista y la funcionalidad básica de registro/login operativa.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID de PBI | Actividad | Descripción | Estimación (Horas) | Estado |
| PB001 | Configuración Inicial y Requerimientos | Configurar el proyecto (framework, BD), definir arquitectura básica. | 16 | Hecho |
| PB003 | Diseño UX/UI Básico (Solo Home/Login) | Bocetos y diseño de las páginas de inicio y autenticación. | 2 | Hecho |
| PB004 | Autenticación de Usuario | Implementar registro de usuario, login y logout. | 12 | En Progreso |
| Sub-tarea de PB002 | Configurar base de datos para usuarios | Crear modelo de usuario y migración. | 4 | Hecho |
| Sub-tarea | Rutas de API para autenticación | Definir endpoints para registro y login. | 8 | En Progreso |
| Sub-tarea | Interfaz de usuario para login/registro | Crear formularios HTML/CSS para login y registro. | 12 | En Progreso |

## 3. Daily Scrum

|  |  |  |  |
| --- | --- | --- | --- |
| Miembro del Equipo | Lo que hice ayer | Lo que haré hoy | Impedimentos |
| Yo | - Terminé la configuración inicial del proyecto (PB001). - Hice los bocetos de la página de inicio y login (parte de PB003). - Configuré la base de datos y el modelo de usuario. | - Comenzaré con las rutas de la API de autenticación (PB004). - Luego, trabajaré en la interfaz de usuario para el formulario de login. | Ninguno por ahora. |

## 4. Revisión del Sprint

|  |  |
| --- | --- |
| Elemento | Descripción |
| Sprint | Sprint 1 |
| Objetivo del Sprint | Tener el proyecto configurado, la base de datos inicial lista y la funcionalidad básica de registro/login operativa. |
| Incremento Entregado | - Proyecto inicial configurado y listo para desarrollo. - Diseños básicos de Home y páginas de autenticación completados. - Funcionalidad de registro de usuarios implementada. - Inicio de sesión básico (login) funcional. |
| Retroalimentación | - El flujo de registro es claro, pero la validación de contraseñas podría ser más robusta. - El diseño es funcional, pero necesita algunos ajustes visuales para ser más atractivo. |
| Acciones Futuras | - Mejorar la validación de contraseñas. - Refinar los estilos CSS para un aspecto más pulido. |

## 5. Retrospectiva del Sprint

|  |  |
| --- | --- |
| Pregunta | Respuesta |
| ¿Qué funcionó bien en este sprint? | - La definición clara de las tareas para el inicio del proyecto fue muy útil. - La capacidad de enfocarme en una sola cosa a la vez (configuración, luego autenticación) fue eficiente. |
| ¿Qué no funcionó bien? | - Subestimé el tiempo que iba a necesitar para los ajustes de CSS, tuve que dejar algunos para después. - No documenté algunos cambios de requerimientos al instante. |
| Compromisos de mejora | - Dedicar un poco más de tiempo a la estética desde el principio, o al menos dejar un margen. - Llevar mejor manejo de |

### 6. Informe de estado del Sprint

|  |  |
| --- | --- |
| Elemento | Descripción |
| Sprint | Sprint 1 |
| Objetivo del Sprint | Tener el proyecto configurado, la base de datos inicial lista y la funcionalidad básica de registro/login operativa. |
| Historias Completadas | - PB001: Configuración Inicial y Requerimientos  - PB002: (Parcial): Diseño de los datos y estructura en la BD (Usuarios) - PB003 (Parcial): Diseño UX/UI Básico (Home y Autenticación) - PB004: Autenticación de Usuario (Registro y Login básico) |
| Historias Incompletas | Ninguna de las comprometidas para este sprint. |
| Riesgos/Impedimentos | - Potencial complejidad al integrar múltiples RLS para las acciones de una tabla. - Posible sobrecarga de trabajo si no priorizo bien las características del carrito. |
| Próximos Pasos | - Comenzar Sprint 2: Enfocarse en la gestión y visualización de productos (PB005, PB006). - Abordar los ajustes visuales. |

# Diseño de Arquitectura y patrones.

## Arquitectura

La arquitectura a utilizar será la de Microservicios ya que una página de e-commerce puede necesitar de múltiples servicios a la vez para realizar tareas específicas, además de que se hace más fácil integrar más servicios a futuro. La razón principal de haberse elegido esta arquitectura es la orientación hacia múltiples servicios, ya que en un e-commerce no es raro la implementación de múltiples servicios independientes que trabajan juntos para dar información o realizar funciones específicas.

### Razones

* **Múltiples funciones a usar:** Con esto me refiero a la cantidad de servicios a implementar en la página, donde cada uno cumplirá con su función a la que fueron diseñadas.
* **Resiliencia**: En caso de que un de estos llegase a fallar esto no causaría un efecto domino en la página.
* **Carga:** Al dividirlo de esta manera esto ayuda a evitar una sobrecarga para un solo servicio y evitando cuellos de botellas con las consultas. Y así en caso de picos de usuarios la página podría estar disponible todo el tiempo necesario.

## Patrón – Aggregator

El patrón elegido para el desarrollo fue el aggregator ya que su enfoque se basa en interconectar y recolectar información y datos de múltiples servicios para construir respuestas validas. Por ende en la página sería útil para verificar disponibilidad y existencia de productos añadidos al carrito, poniendo en llamado 2 o 3 servicios, para cumplir esa meta.

### Razones

* **Optimizar el rendimiento del back:** Ya que al hacerlo de esta manera es posible implementar maneras o técnicas ya existentes para extraer la información de la BD sin afectar su rendimiento por las múltiples acciones realizadas por los usuarios.
* **Buen impacto en la Red:** Esto se refiere al impacto que se tiene en la latencia de las llamadas al back, ya que, en vez de esperar las respuestas de los servicios, el cliente solo esperaría la respuesta del Aggregator.

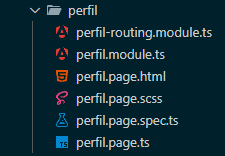
# Desarrollo con Frameworks

Las herramientas que se utilizarán durante el desarrollo son las siguientes:

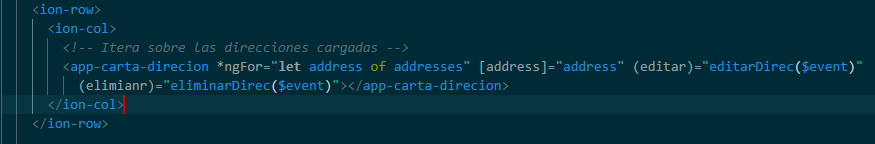
## Ionic-Angular

El framework de ionic permite realizar proyectos con diferentes tecnologías como react o vue. En este caso con Angular que utiliza un tipo de enfoque cercano al MVC con su estructura de páginas, para su demostración pondré 2 ejemplos:

### Ejemplo de la página de perfil:



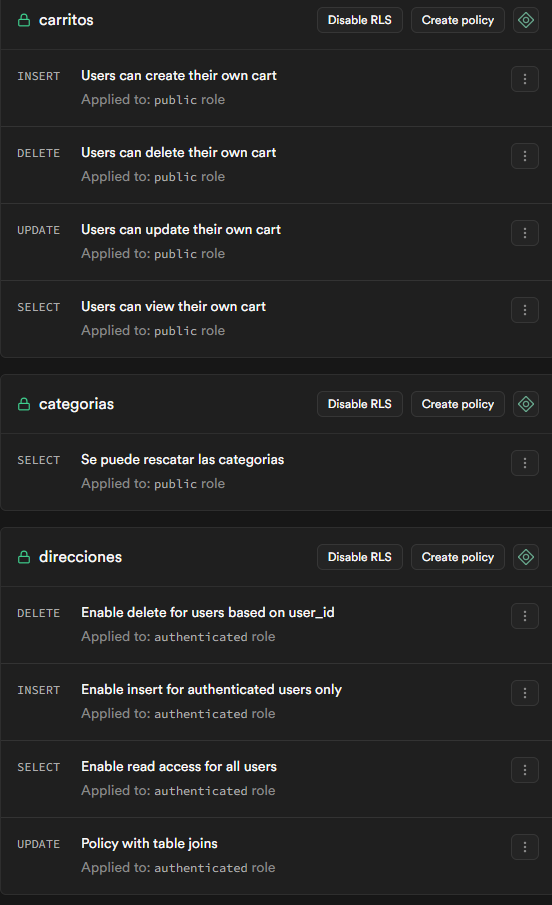
Como se puede observar al generar una página crea estos elementos, el routing para rutas, el module para importaciones de librerías para la estructura o funciones, el html para el diseño, scss para el estilo, y ts las funciones, mientras que el spec.ts para pruebas. Dentro de esta página hay algo a destacar y es lo siguiente:



Es la carga y uso de componentes globales que se pueden modificar a la necesidad que uno requiera mediante la carga de valores en sus atributos.

## Backend Supabase

Para las acciones del backend se implementará el uso del supabase, con el cual puedo realiza llamadas a la BD, autenticaciones, registros, inicios de sesión, tokens, etc. Para la BD el servicio usa Postgres, por lo que para cada tabla de se definen las RLS (Row Level Security) que son politicas necesarias para realizar acciones dentro de la tabla como select, insert, delete, update.



# Integración de Servicios Web

Por el momento el único servicio externo implementado es el de Supabase, el cual lo utilizó mediante llamadas desde los services en el proyecto, de los cuales hay 5 servicios únicos.

## Servicios

### Carrito

Como indica el nombre este servicios funge como el intermedio para el manejo de los carritos de los usuario. Para su demostración utilizaré solo la función de obtención de los ítems en el carrito:

*// CODIGO PARA OBTENER LOS ITEMS DEL CARRITO ACTUAL*

**async** getCarritoItems(): Promise<{ items: ItemCarrito[] | null; error: any }> {

**const** { Carrito, error: cartError } = await this.getOrCreateCart();

//consigo los valores de Carrito y el error de la función principal

    if (cartError || !Carrito) {

      return { items: null, error: cartError || { message: 'No se obtuvo el carrito' } } Verifico que no existan errores en la conexión

    }

*//Hacemos una union para conseguir los detalles del prodcuto*

**const** { data: items, error } = await this.supabase

      .from('items\_carrito')

      .select(`\*, productos (id, nombre, imagen\_url, precio)`)

      .eq('id\_carrito', Carrito.id\_carrito);

    if (error) {

      return { items: null, error };

    }

// Mapeo los valores de la interfaz de ItemCarrito

**const** mapeado: ItemCarrito[] = items.map((item:any)**=>**({

      ...item, // …item sirve para mapear de forma automatica todos lo valores cuando estos tiene el mismo nombre.

      producto:{ //Como rescato la información del producto también llamo a su interfaz y tomo los valores de item.

        id: item.productos.id,

        nombre: item.productos.nombre,

        imagen\_url: item.productos.imagen\_url,

        precio: item.productos.precio

      }as Producto

    }));

    return {items: mapeado, error:null};

  }

### Pedidos

Este servicio sirve para manejar los pedidos hechos por los usuarios. Para ejemplo utilizare la función de obtención.

**async** getPedidosDelUsuario(): Promise<{ data: Pedido[] | null; error: any }> {

**const** { data: user, error: userError } = await this.supabase.auth.getUser();

    if (userError || !user?.user) {

      console.error('Error al obtener usuario autenticado:', userError);

      return { data: null, error: userError || new Error('Usuario no autenticado.') };

    } // Primero identifico la existencia de un usuario

**const** userId = user.user.id; // Si hay uno obtengo su id

*// Obtener pedidos del usuario*

**const** { data: pedidos, error: pedidosError } = await this.supabase

      .from('pedidos')

      .select(`

        \*,

        items\_pedido ( //Este obtiene de otra tabla los ítems correspondientes

          id\_item\_pedido,

          cantidad,

          precio\_unitario,

          productos (

            id,

            nombre,

            imagen\_url

          )

        )

      `)

      .eq('id\_usuario', userId) //Busco los pedidos por bajo su id

      .order('fecha\_pedido', { ascending: false }); *// Ordenar por fecha, los más recientes primero*

    if (pedidosError) {

      console.error('Error al obtener los pedidos:', pedidosError);

      return { data: null, error: pedidosError };

    }

*// Mapear los datos para asegurar que los productos estén directamente anidados si es necesario*

**const** formattedPedidos: Pedido[] = pedidos.map((pedido: any) **=>** ({

      ...pedido,

      items\_pedido: pedido.items\_pedido.map((item: any) **=>** ({

        ...item,

        producto: item.productos *// Renombrar 'productos' a 'producto' para que coincida con la interfaz ItemPedido*

      }))

    }));

    return { data: formattedPedidos as Pedido[], error: null };

  }

### Productos

Este servicio se encarga de conseguir la información de los productos en la BD para luego ser procesadas en los TS para su muestra en el front. La función a explicar es la obtener un producto especifico, que se usa para cuando eliges un producto.

**async** getProductoById(id: string): Promise<{ data: Producto | null; error: any }> {

**const** { data, error } = await this.supabase // del select recoge la información como data y el error

      .from('productos')

      .select('\*')

      .eq('id', id)

      .single(); //Indica que será una sola fila

    if (error) {

      console.error(`Error al obtener el producto con ID ${id}:`, error.message);

    }

    return { data: data as Producto, error }; // Si es exitosa devuelve data como la interfaz producto.

  }

También para buscar la página se debió configurar la ruta para que aceptará parámetros que este caso sería el id.

### Servicio del perfil

Este se encarga de manejar la información del usuario la cual muestra dicha información en la página de perfil. Para este explicare la forma de subir avatares y la forma de recuperarlos.

*/\*\**

*\* Sube un archivo de avatar a Supabase Storage.*

*\*/*

**async** uploadAvatar(file: File): Promise<string> {

**const** userId = await this.getUserId();

    if (!userId) {

      throw new Error('No hay usuario autenticado para subir el avatar.');

    }

**const** fileExtension = file.name.split('.').pop();

**const** fileName = `${userId}-${Date.now()}.${fileExtension}`; *// Nombre único*

**const** filePath = `${userId}/${fileName}`; *// Carpeta por usuario*

**const** { data, error } = await this.supabase.storage

      .from('avatars')

      .upload(filePath, file, {

        cacheControl: '3600',

        upsert: true *// Sobrescribe si el archivo ya existe (útil para el mismo nombre)*

      });

    if (error) {

      console.error('Error al subir el avatar:', error.message);

      throw error;

    }

    return data.path; *// Retorna el path dentro del bucket, e.g., "usuario\_id/imagen.png"*

  }

*/\*\**

*\* Obtiene la URL pública de un archivo de avatar.*

*\*/*

  getPublicUrl(path: string): string | null {

    if (!path) return null;

**const** { data } = this.supabase.storage

      .from('avatars')

      .getPublicUrl(path);

    return data.publicUrl;

  }

### Supabase – Registro

Este servicio funciona para todas las acciones de registro, inicio, cambio de contraseña y cambios en el perfil. La función a explicar sería la actualización de datos al registrarte.

*//MOMDIFICACIONES PARA EL PERFIL*

**async** guardarPerfil(dataPerf: Omit<Perfil, 'id' | 'correo'>){

**const** user = await this.supabase.auth.getUser();

    if(!user.data.user){

      throw new Error('No existe usuario autenticado')

    }

**const** userId = user.data.user.id;

**const** userEmial = user.data.user.email;

**const** {data, error} = await this.supabase

    .from('profiles').upsert({ //Upsert actualiza e inserta valores

      id:userId,

      correo:userEmial,

      ...dataPerf

    },{onConflict: 'id'}); //En caso de existencia los actualiza.

    if (error){

      console.error('Error al gurdar el perfil');

      throw error;

    }

    return data

  }

# LINKS

|  |  |
| --- | --- |
| Página | <https://kolmeta-7314b.web.app/> |
| Github | [uLebW/principal at desarrollo](https://github.com/uLebW/principal/tree/desarrollo) |