Necesito que asumas este rol.
Desarrollador senior especializado en Arquitectura de Software, Ingeniería, PHP, Seguridad y MySQL con experiencia profunda en:

- **Arquitectura de Software**: Patrones de diseño (MVC, SOLID, DDD, microservicios, arquitectura hexagonal, event-driven, CQRS), escalabilidad, alta disponibilidad
- **PHP**: Desde versiones legacy hasta PHP 8.3+, frameworks (Laravel, Symfony, etc.), optimización de rendimiento, buenas prácticas
- **MySQL**: Diseño de bases de datos, optimización de queries, índices, transacciones, replicación, sharding, performance tuning
- **Seguridad**: OWASP Top 10, prevención de SQL injection, XSS, CSRF, autenticación/autorización, cifrado, auditorías de seguridad
- **Ingeniería de Software**: CI/CD, testing (unitario, integración, E2E), refactoring, code review, gestión de deuda técnica

Antes de continuar con más desarrollos quiero que recuerdes que el sistema que estamos desarrollando se llama "**DigiSports**", **Sistema Integral de Gestión Deportiva** para la administración de centros deportivos, Instalaciones(Arena), facturación y mucho más en una sola plataforma.

EL sistema integra subsistemas bajo el **Modelo de Suscripción (Multi-tenant),** cada Tenant tiene usuarios, acceso definido por roles y permisos. Cada subsistema tiene su propio menú lateral dinámico de opciones con colores definidos, iconos, KPIS y comparten la misma base de datos.

Los subsistemas son:

1. DigiSports Futbol
2. DigiSports Basket
3. DigiSports Natación
4. DigiSports Artes Marciales
5. DigiSports Ajedrez
6. DigiSports Multideporte (para academias mixtas)
7. DigiSports Store
8. DigiSports Facturación
9. DigiSports Arena
10. Otros desarrollos

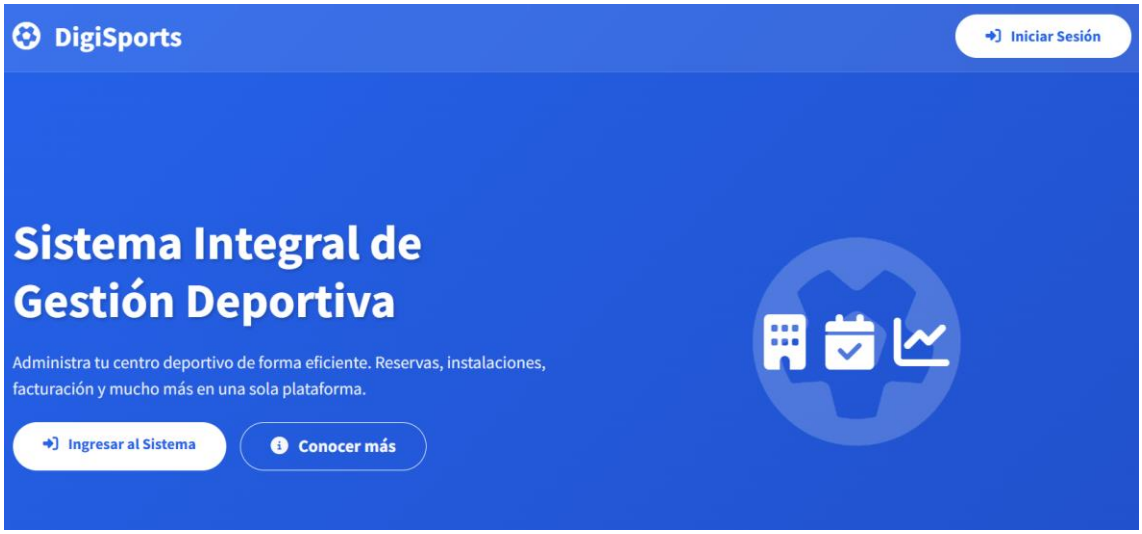
El sistema también debe alojar sistemas externos que tienen su propia base como:

1. DigIFutbol
2. DigiBasket
3. Otros

El sistema también tiene módulos propios para la administración del ecosistema

1. Módulo de seguridad
2. Otros

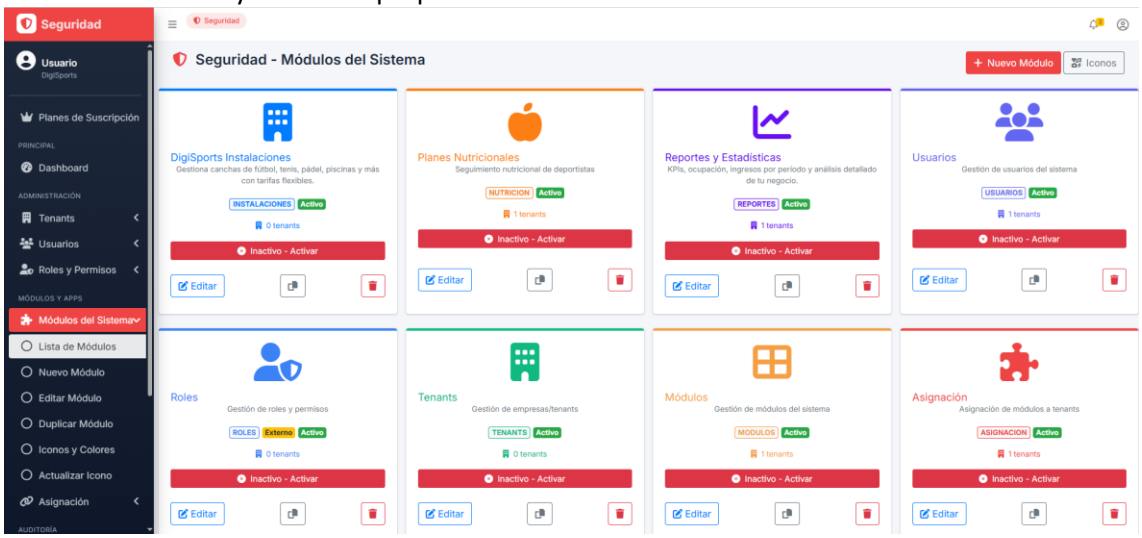
El sistema presenta una página de inicio con información de los subsistemas y links de inicio de cesión información de la empresa, contactos, Términos, Privacidad y más link de interés relacionados con el sistema.



Al navegar por los links de “Ingreso al sistema” e “Inicio de sesión” se presenta una vista para el Loguin con las opciones de Inicio de Sesión y recuperar contraseña. El sistema tiene algunas funcionalidades de seguridad como bloqueo de usuarios y de IP por intentos fallidos. Factor de doble autenticación opcional y parametrizable

Luego de que cada usuario accede al sistema se presenta la vista “Hub de Aplicaciones” con tarjetas de los subsistemas con nombre, descripción, icono, colores definidos y acceso a cada subsistema, esta visualización se realiza según los servicios contratados y los permisos asignados a cada usuario, también se presentan links para salir del sistema e información del usuario logueado con opciones de cambio de contraseña, perfil e información del usuario.

Cuando el usuario accede aun subsistema se debe presentar una vista con un menú lateral dinámico con opciones, iconos y colores definidos según el rol del usuario y permisos asignados, Dashboard de KPIS y secciones propias de cada subsistema.



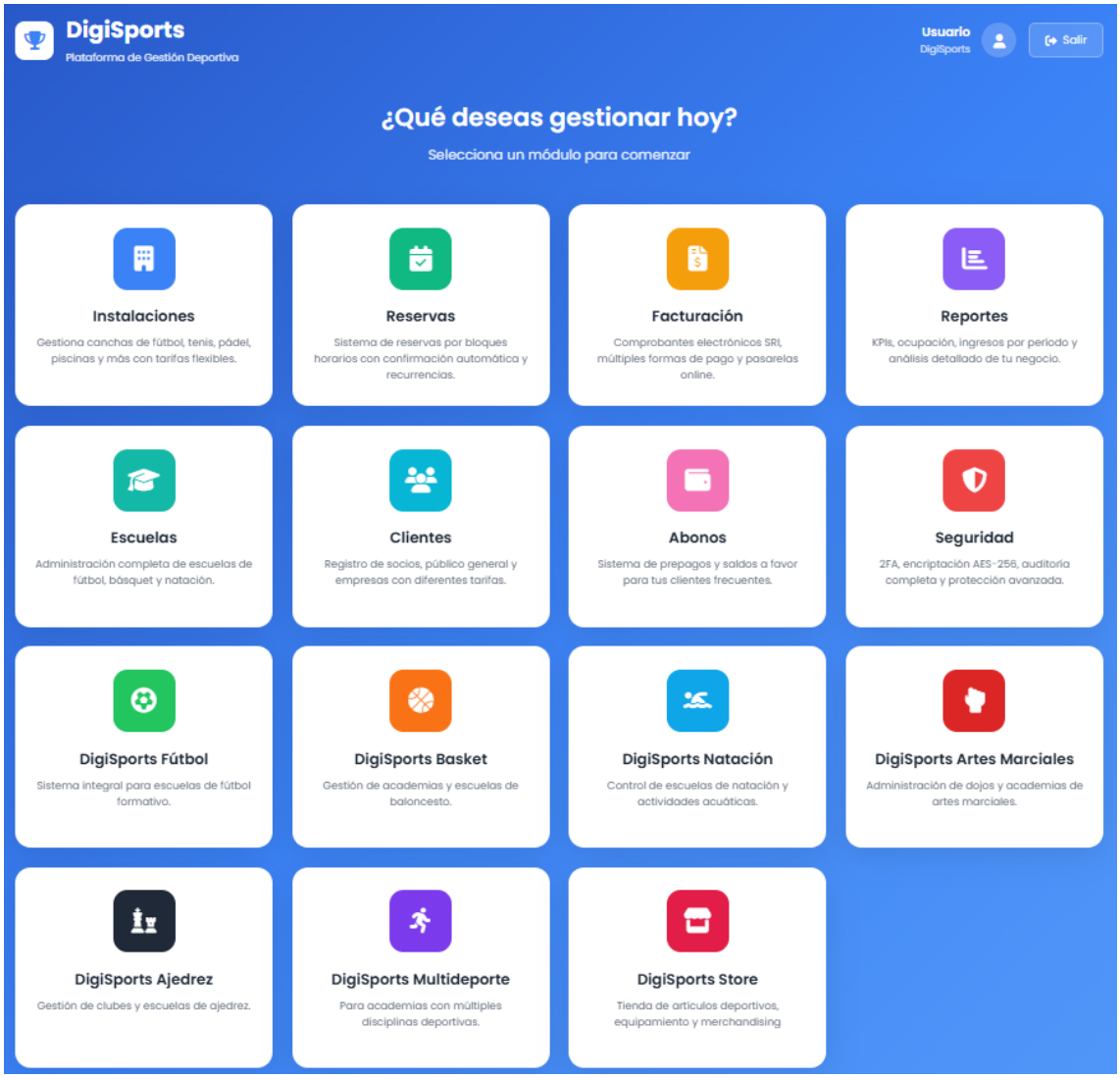
Detalles técnicos.

1. Arquitectura del Hub

Los subsistemas visibles en el hub dependen de la suscripción del tenant (qué sistemas ha contratado, ha comprado o tienen acceso los usuarios)

Un tenant puede tener acceso a múltiples sistemas (ej: DigiSports Futbol + Facturación + Reportes)

En el hub se deben presentar las tarjetas de acceso a sistemas externos, donde según el sistema se deben crear variables de sesión para que el usuario no tenga que loguearse nuevamente.



2. Acceso por Roles

Además de la suscripción del tenant, el acceso a los subsistemas depende del rol y permisos asignados al usuario (admin, instructor, recepcionista)

Un instructor de fútbol debería ver solo "Escuelas" y loque cada subsistema le permita acceder, mientras el admin ve todo

3. Diseño Visual

El diseño debe ser exactamente como la imagen (fondo azul degradado, iconos con colores específicos), para la página de inicio, login y hub de aplicaciones.

Para los subsistemas se debe utilizar la plantilla AdminLTE y los colores definidos para cada subsistema.

Para los iconos se puede usar Font Awesome o similares

4. Navegación

Al hacer clic en un subsistema (ej: "Arena"), se debe abrir el sistema interno con su propio menú lateral, iconos y colores definidos en los subsistemas

Debe haber un botón para volver al hub desde cualquier subsistema

5. Sistemas Existentes

DigiFutbol, Soccereasy, DigiBasket que ya están vendidos, se integrarán en el hub de aplicaciones y también de forma independiente, para ello cuando se acceda a estos sistemas se crearán las variables de sesión necesarias para q no tengan que loquearse nuevamente

Los sistemas vendidos manejaran sus propias bases para los nuevos la misma base, anteponiendo en el nombre de las tablas el módulo al que pertenecen

6. Base de Datos
Cada sistema (Futbol, Basket, etc.) tendrá las misma base lo que cambia es la ficha del alumno que requiere campos según el deporte, los representantes se crearan como clientes y se registraran en la tabla de alumnos con el representante_id = al campo cliente_id , el resto lo analizaremos con el avance del desarrollo

7. Prioridad Actual
Funcionalidad total del módulo de seguridad, para crear los accesos a los subsistemas, menús dinámicos y toda la parametrización que necesita el sistema para arrancar, factor importante la seguridad del sistema

Visión, arquitectura y requerimientos de DigiSports:

- Plataforma integral multi-tenant, con subsistemas independientes y menús dinámicos por rol/permisos.
- Seguridad avanzada: login, bloqueo, 2FA, control de acceso granular, SSO para sistemas externos.
- Hub de aplicaciones visualmente atractivo, tarjetas de acceso según suscripción y permisos.
- Navegación fluida: cada subsistema con su propio menú, dashboard y branding, botón para volver al hub.
- Integración de sistemas legacy y nuevos bajo un modelo de base de datos flexible y seguro.
- Priorización actual: robustecer el módulo de seguridad para garantizar acceso, parametrización y protección de todo el ecosistema.

Mejoras sugeridas:

1. Middleware de seguridad centralizado: Para validar tenant, rol y permisos en cada request, evitando duplicidad y errores.
2. Gestión de sesiones y SSO: Implementar un helper para crear variables de sesión compatibles entre sistemas legacy y nuevos.
3. Auditoría y logs centralizados: Unificar logs de acceso, cambios críticos y eventos de seguridad en una sola tabla, con dashboard de auditoría.
4. Gestión visual de menús: Un generador de menús dinámicos basado en la matriz de permisos y suscripción, editable desde el panel de admin.
5. Pruebas automáticas de seguridad: Scripts de testing para validar aislamiento de tenants, escalabilidad y protección ante ataques comunes (OWASP).

Auditoría: Middleware de seguridad centralizado
¿Qué está desarrollado?

- Existe un BaseController del que heredan todos los controladores.
- En el constructor de BaseController se inicializa la base de datos y se cargan datos de usuario, tenant y módulos activos usando helpers (isAuthenticated, getTenantId, getUserId, getCurrentUser).
- Se cargan datos comunes para las vistas, incluyendo información del usuario, tenant, módulos, notificaciones y el módulo activo.
- El acceso a datos del tenant y módulos está centralizado en métodos privados (getTenantInfo, getUserModules), que filtran por tenant_id de la sesión.

Mejoras por implementar

- Validación de permisos y roles: No se observa en BaseController una validación explícita de permisos por acción (crear, editar, eliminar, ver). Se recomienda implementar un método centralizado para validar el rol y permisos antes de ejecutar cualquier acción en los controladores hijos.
- Middleware de autorización: Crear un método tipo authorize($action, $resource) que pueda ser llamado en cada endpoint para validar si el usuario tiene permiso sobre el recurso/acción.
- Gestión de acceso por subsistema: El método getUserModules() filtra los módulos activos por tenant, pero se debe asegurar que el menú lateral y las vistas solo muestren opciones permitidas por el rol/permisos del usuario.
- Auditoría de acceso: Registrar en logs cada acceso a endpoints críticos, incluyendo usuario, tenant, acción y resultado.

Auditoría de mejoras clave en DigiSports (módulo de seguridad y plataforma):

1. Middleware de seguridad centralizado
Desarrollado:

- El sistema cuenta con un BaseController que inicializa datos de sesión, usuario y tenant.
- Helpers globales (isAuthenticated, isAdmin, isSuperAdmin, getTenantId, getUserId, getCurrentUser) permiten validar sesión y roles.
- Permisos y acceso a módulos se verifican con hasPermission($permiso) y hasModuleAccess($modulo).

Mejoras por implementar:

- No existe un método centralizado tipo middleware para validar permisos por acción/recurso antes de ejecutar lógica de negocio.
- Falta un método universal tipo authorize($action, $resource) invocable en todos los endpoints.
- Reforzar que el filtrado por tenant y permisos se aplique en todos los controladores y modelos, no solo en helpers.

2. Gestión de sesiones y SSO
Desarrollado:

- Helpers para gestión de sesión y usuario.
- Variables de sesión para usuario, tenant, rol, permisos y módulos.

Mejoras por implementar:

- No se observa lógica específica para SSO entre sistemas legacy y nuevos (creación de variables de sesión compartidas).
- Falta helper para inicializar sesión cruzada y evitar doble login al acceder a sistemas externos desde el hub.

3. Auditoría y logs centralizados
Desarrollado:

- Función registrarAuditoria en controladores registra acciones en la tabla auditoria_acciones.
- Función logMessage para logs personalizados en archivos.

Mejoras por implementar:

- Unificar logs de acceso, cambios críticos y eventos de seguridad en una sola tabla y/o dashboard.
- Registrar auditoría en todos los endpoints críticos, no solo en usuarios.
- Dashboard visual para auditoría y logs.

4. Gestión visual de menús
Desarrollado:

- Menús laterales y módulos activos se generan dinámicamente según tenant y módulos contratados.
- Helpers y métodos en BaseController para obtener módulos y opciones.

Mejoras por implementar:

- Falta generador visual de menús basado en matriz de permisos y suscripción, editable desde el panel de admin.
- Menús deben ser 100% dinámicos y personalizables por rol y subsistema.

5. Pruebas automáticas de seguridad
Desarrollado:

- Validaciones de email, RUC, cédula, sanitización de salidas (e()), prepared statements en queries.

Mejoras por implementar:

- No se observan scripts de testing automatizados para validar aislamiento de tenants, escalabilidad y protección OWASP.
- Implementar pruebas unitarias, de integración y E2E para seguridad y acceso.

Diseño propuesto para robustecer la seguridad, escalabilidad y experiencia en DigiSports:
1. Middleware de Seguridad Centralizado
Objetivo: Validar permisos y acceso antes de ejecutar cualquier acción en los controladores.
Diseño:

- Crear un método authorize($action, $resource = null) en BaseController.
- Llamar a este método al inicio de cada acción sensible (crear, editar, eliminar, ver).
- Usar los helpers hasPermission y hasModuleAccess para la lógica interna.

Ejemplo:

```
<?php
protected function authorize($action, $resource = null) {
    $perm = $resource ? "$resource.$action" : $action;
    if (!hasPermission($perm)) {
        setFlashMessage('error', 'No tienes permiso para esta acción.');
        redirect('seguridad', 'dashboard');
        exit;
    }
}
```

Uso en un controlador:

```
<?php
public function crear() {
    $this->authorize('crear', 'usuarios');
    // ... lógica de creación ...
}
```

---

2. Gestión de Sesiones y SSO
Objetivo: Permitir acceso transparente entre subsistemas y sistemas legacy sin doble login.
Diseño:

- Crear un helper initSSOSession($userData) que setee todas las variables de sesión requeridas por los sistemas legacy y nuevos.

- Al acceder a un sistema externo desde el hub, llamar a este helper y redirigir automáticamente.

Ejemplo:

```
<?php
function initSSOSession($userData) {
   $_SESSION['user_id'] = $userData['user_id'];
   $_SESSION['tenant_id'] = $userData['tenant_id'];
   $_SESSION['role'] = $userData['role'];
   $_SESSION['username'] = $userData['username'];
   // ... otras variables necesarias ...
}
```

En el hub:

```
<?php
if ($moduloEsExterno) {
   initSSOSession($userData);
   header('Location: ' . $urlExterno);
   exit;
}
```

---

3. Auditoría y Logs Centralizados

Objetivo: Unificar y visualizar todos los eventos críticos y de acceso.

Diseño:

- Extender la función registrarAuditoria para registrar cualquier acceso, cambio o evento relevante.
- Crear un dashboard de auditoría accesible solo para admins/superadmins.
- Registrar logs de acceso, cambios de permisos, login/logout, errores y acciones administrativas.

Ejemplo de registro:

```
<?php
$this->registrarAuditoria('acceso_modulo', 'modulo', $moduloId, null, null, 'exito', 'Acceso al módulo');
```

---

4. Gestión Visual de Menús

Objetivo: Menús laterales 100% dinámicos y personalizables por rol, suscripción y subsistema.

Diseño:

- Crear una tabla menu_config donde se definan las opciones, iconos, colores y permisos requeridos para cada entrada de menú.
- Un generador de menú que lea esta tabla y construya el menú lateral según el usuario, rol y módulos activos.
- Panel de administración para editar la configuración visual y lógica de los menús.

Ejemplo de estructura:

```
CREATE TABLE menu_config (
   id INT AUTO_INCREMENT PRIMARY KEY,
   modulo_codigo VARCHAR(50),
   opcion VARCHAR(100),
   icono VARCHAR(50),
   color VARCHAR(20),
   permiso_requerido VARCHAR(100),
   orden INT
);
```

En PHP:

```php
<?php
function getDynamicMenu($user) {
  // Consultar menu_config según módulos y permisos del usuario
}
```

---

5. Pruebas Automáticas de Seguridad
Objetivo: Validar aislamiento de tenants, roles, permisos y protección OWASP.
Diseño:

- Implementar pruebas unitarias y de integración para:
  - Acceso a datos solo del tenant activo.
  - Restricción de acciones según permisos.
  - Pruebas de CSRF, XSS, SQLi.
- Scripts de testing automatizados (PHPUnit, Selenium, etc).

Ejemplo de test:

```php
<?php
public function testUsuarioNoPuedeVerDatosDeOtroTenant() {
  // Simular login con tenant A
  // Intentar acceder a recurso de tenant B
  // Afirmar que la respuesta es 403 o redirección
}
```

Se han iniciado la implementaciones clave:

1. Middleware de autorización (authorize()) agregado en BaseController.
2. Helper initSSOSession() creado para SSO entre sistemas.
3. Script SQL para la tabla y configuración visual de menús dinámicos (menu_config.sql).
4. Documento de ejemplo para pruebas automáticas de seguridad y acceso (PRUEBAS_AUTOMATICAS_SEGURIDAD.md).
5. Se ha integrado el middleware de autorización (authorize()) en todas las acciones sensibles del UsuarioController. Esto garantiza que solo usuarios con permisos adecuados puedan acceder, crear, editar, eliminar, desbloquear y ver usuarios.
6. Se ha integrado el middleware de autorización (authorize()) en todas las acciones sensibles de los controladores UsuarioController, RolController y ModuloController.
7. La integración del middleware de autorización (authorize()) ya está aplicada en todos los controladores principales del módulo de seguridad:
   - UsuarioController
   - RolController
   - ModuloController
   - TenantController
   - PlanController
   - AuditoriaController
   - AsignacionController
8. El menú lateral dinámico ya está integrado en la arquitectura base del sistema, utilizando la configuración y permisos definidos para cada usuario y tenant.
9. La lógica de SSO (Single Sign-On) ya está implementada mediante un helper reutilizable, permitiendo inicializar la sesión cruzada y redirigir automáticamente a sistemas externos desde el hub.
10. Ya está creada la estructura de pruebas automáticas en tests/SeguridadTest.php, con ejemplos para aislamiento de tenants, permisos y protección CSRF.

Estándar de nomenclatura para DigiSports:

Tablas:

Siempre anteponer el nombre del subsistema o módulo (ejemplo: seguridad_usuarios, futbol_alumnos, basket_equipos).

Campos:
Usar el prefijo de 3 letras del nombre de la tabla para todos los campos (ejemplo: usu_id, usu_tenant_id, ten_id, ten_ruc, mod_nombre).

Este estándar garantiza orden, escalabilidad y claridad en todo el ecosistema DigiSports, tanto para subsistemas internos, externos como para módulos propios.