



SE

Manual técnico del sistema

Suite Educativa Integral SaaS · Documentación de arquitectura

Arquitectura, modelo de datos y procesos internos

Ficheros y tablas que participan en cada módulo

PHP 8 · Laravel 11 · MySQL 8 · Tailwind CSS

Versión 1.0 · Documento técnico

Contenido

- 1 Panorama general del sistema**
Stack tecnológico · Estructura de carpetas · Convenciones
- 2 Arquitectura multi-institución (multi-tenant)**
Aislamiento por empresa_id · Scopes · Autorización
- 3 Seguridad y control de acceso**
Middleware · Roles · Activación de módulos
- 4 Ciclo de vida de una petición**
- 5 Modelo de datos completo**
Las 32 tablas agrupadas por dominio · Mapa de relaciones
- 6 Catálogo de módulos: ficheros, tablas y procesos**
Panel SaaS: Empresas · Planes · Suscripciones · Mantenimiento
Académico: Admisiones · Estudiantes · Docentes · Programas
Aulas · Horarios · Asistencia · Evaluaciones · Certificados
Campus Virtual · Biblioteca
Finanzas: Pagos · Caja
Administración: CRM · Comunicaciones · RRHH · Inventario · Reportes
Sistema: Configuración · Mantenimiento del tenant
- 7 Procesos transversales**
Dashboard · Reportes y exportación · Copias de seguridad
- 8 Instalación y puesta en marcha**
- 9 Cómo agregar un módulo nuevo**
- 10 Anexos**
Índice de tablas · Índice de rutas · Inventario de ficheros

1. Panorama general del sistema

La Suite Educativa Integral es una aplicación web **SaaS multi-institución** construida sobre Laravel 11. Una sola instalación y una sola base de datos atienden a un número indefinido de instituciones educativas, cada una con sus datos completamente aislados de las demás.

Stack tecnológico

Capa	Tecnología	Observaciones
Lenguaje	PHP 8.2 o superior	Requiere las extensiones pdo_mysql, mbstring, openssl y fileinfo.
Framework	Laravel 11	Esqueleto ligero: la configuración vive en <code>bootstrap\app.php</code> , sin <code>Http\Kernel.php</code> .
Base de datos	MySQL 8 / MariaDB 10.4+	Base <code>suite_saas_educacion_integral</code> , motor InnoDB, cotejamiento utf8mb4.
Plantillas	Blade	76 vistas organizadas por módulo.
Estilos	Tailwind CSS vía CDN	Configuración en línea dentro del layout. No requiere Node ni compilación.
Gráficos	Chart.js 4.4.1 vía CDN	Usado en el dashboard y en Reportes.
Servidor	Apache, Nginx o <code>php artisan serve</code>	La raíz pública es la carpeta <code>public\</code> .

Por qué Tailwind por CDN y no compilado

La decisión es deliberada: elimina la dependencia de Node.js y del proceso de compilación. El sistema se despliega copiando ficheros y ejecutando las migraciones, sin cadena de build. A cambio, la hoja de estilos se descarga completa; para producción con mucho tráfico conviene migrar a compilación con Vite.

Estructura de carpetas

Raíz del proyecto

```
app\Http\Controllers\ → 30 controladores (lógica de cada módulo)
app\Http\Controllers\Admin\ → controladores del panel Super Admin SaaS
app\Http\Controllers\Configuracion\ → configuración interna del tenant
app\Http\Middleware\ → 3 middleware de control de acceso
app\Http\Requests\ → validaciones reutilizables (FormRequest)
app\Models\ → 27 modelos Eloquent
app\Providers\ → AppServiceProvider (composer del menú)
bootstrap/app.php → rutas, middleware y alias (Laravel 11)
config\ → configuración de la aplicación
database\Migrations\ → definición de las 32 tablas
database\seeders\ → 11 seeders (catálogos y datos demo)
database\sql\ → scripts SQL auxiliares
```

docs\ → manuales en PDF

public\ → raíz web, punto de entrada index.php

resources\views\ → 76 vistas Blade

routes\web.php → 163 rutas nombradas

storage\app\backups\ → copias de seguridad generadas

storage\framework\views\ → caché de vistas Blade compiladas

Convenciones que sigue todo el código

Convención	Regla aplicada
Idioma	Todo el dominio está en español: tablas, columnas, modelos y rutas. Sólo las palabras reservadas del framework quedan en inglés.
Nombre de tablas	Plural en minúscula y sin tildes: <code>estudiantes</code> , <code>matriculas</code> , <code>movimientos_caja</code> .
Nombre de modelos	Singular en PascalCase: <code>Estudiante</code> , <code>Matricula</code> . Cuando el plural no es regular se declara <code>\$table</code> de forma explícita.
Nombre de rutas	Punteado por módulo: <code>estudiantes.index</code> , <code>pagos.store</code> , <code>admin.empresas.edit</code> .
Claves foráneas	Siempre <code>_id</code> con <code>constrained()</code> y política explícita de borrado (<code>cascadeOnDelete</code> o <code>nullOnDelete</code>).
Aislamiento	Toda tabla operativa lleva <code>empresa_id</code> y su modelo expone el scope <code>scopeDeMiEmpresa()</code> .
Estados	Se declaran como constantes públicas en el modelo (<code>public const ESTADOS</code>) y se validan con <code>Rule::in()</code> .

2. Arquitectura multi-institución (multi-tenant)

Este es el concepto central del sistema y conviene entenderlo antes que cualquier otra cosa. El sistema usa el patrón **base de datos compartida con columna discriminante**: todas las instituciones conviven en la misma base, y cada fila operativa lleva marcada a qué institución pertenece.

Las tres capas de aislamiento

1

Capa 1 — La columna `empresa_id`

Toda tabla operativa incluye `empresa_id` como clave foránea hacia `empresas` con `cascadeOnDelete`. Es la marca física del propietario del dato.

2

Capa 2 — El scope de consulta

Cada modelo define `scopeDeMiEmpresa()`, que añade automáticamente el filtro por la institución del usuario autenticado. Todos los listados lo usan.

3

Capa 3 — La verificación por registro

Cada controlador implementa un método `autorizar()` que se ejecuta al acceder a un registro concreto. Si el registro no pertenece a la institución del usuario, responde 403.

Implementación del scope

Patrón presente en los 20 modelos operativos

```
public function scopeDeMiEmpresa(Builder $query): Builder
{
    return $query->where('empresa_id', auth()->user()->empresa_id);
}
```

Implementación de la verificación por registro

Patrón presente en todos los controladores del tenant

```
protected function autorizar(Estudiante $estudiante): void
{
    abort_if(
        $estudiante->empresa_id !== auth()->user()->empresa_id
        && ! auth()->user()->esSuperadmin(),
        403,
        'Este registro pertenece a otra institucion.'
    );
}
```

Por qué hacen falta las tres capas

El scope protege los **listados**, pero no impide que alguien escriba a mano la dirección `/panel/estudiantes/451` con el identificador de un alumno de otra institución. Ahí actúa `autorizar()`. Y la columna `empresa_id` es lo que hace posibles a las otras dos. Quitar cualquiera de las tres abre una fuga de datos entre clientes.

Tabla empresas: la raíz de todo

Tabla `empresas` · migración `0001_01_01_000001_create_empresas_table.php`

Columna	Tipo	Restricciones / relación
<code>id</code>	bigint UNSIGNED	PK autoincremental
<code>tipo_negocio_id</code>	bigint UNSIGNED (FK)	→ <code>tipos_negocio</code>
<code>nombre</code>	string	—
<code>slug</code>	string	único
<code>ruc</code>	string	nullable
<code>email</code>	string	nullable
<code>telefono</code>	string	nullable
<code>direccion</code>	string	nullable
<code>logo</code>	string	nullable
<code>color_primario</code>	string	default #2f6bff
<code>plan</code>	string	default pro
<code>activo</code>	boolean	default true
<code>created_at</code> / <code>updated_at</code>	timestamp	gestionados por Eloquent

Los dos ámbitos de la aplicación

Aspecto	Panel SaaS (plataforma)	Panel del tenant (institución)
Prefijo de URI	<code>/admin</code>	<code>/dashboard</code> y <code>/panel/...</code>
Middleware	<code>['auth', 'superadmin']</code>	<code>'auth' + modulo:</code> por módulo
Controladores	<code>app\Http\Controllers\Admin\</code>	<code>app\Http\Controllers\</code>
Layout	<code>admin\layouts\app.blade.php</code>	<code>layouts\app.blade.php</code>
Menú lateral	<code>admin\partials\sidebar.blade.php</code> (arreglo fijo)	<code>partials\sidebar.blade.php</code> (dinámico desde BD)
Alcance de los datos	Todas las instituciones	Sólo la propia

3. Seguridad y control de acceso

Los tres middleware del sistema

Se registran como alias en `bootstrap\app.php` dentro del bloque `withMiddleware()`:

Alias	Clase	Qué verifica	Respuesta si falla
superadmin	EnsureSuperadmin	Que el usuario tenga rol superadmin.	Error 403
rol:a,b	EnsureUserHasRole	Que el rol del usuario esté entre los indicados.	Error 403
modulo:slug	EnsureModuleIsActive	Que la institución tenga ese módulo activo en <code>empresa_modulo</code> .	Error 403 o redirección

Registro de los alias · `bootstrap\app.php`

```
->withMiddleware(function (Middleware $middleware) {  
    $middleware->alias([  
        'modulo' => EnsureModuleIsActive::class,  
        'rol' => EnsureUserHasRole::class,  
        'superadmin' => EnsureSuperadmin::class,  
    ]);  
});
```

Los seis roles

El rol se guarda como cadena en la columna `rol` de la tabla `users`.

Rol	Ámbito	Alcance funcional
superadmin	Plataforma	Único rol que accede a <code>/admin</code> . No pertenece a ninguna institución.
admin	Institución	Acceso total dentro de su institución, incluida Configuración y Mantenimiento.
coordinador	Institución	Gestión académica: programas, horarios, docentes y evaluaciones.
docente	Institución	Asistencia de sus clases, notas y publicación en el Campus.
secretaria	Institución	Admisiones, estudiantes y cobranza.
estudiante	Institución	Consulta de su propia información.

Activación de módulos por institución

El menú de cada institución no está escrito en el código: se construye consultando la base de datos. Intervienen tres piezas.

Tabla `modulos` · migración `0001_01_01_000003_create_modulos_table.php`

Columna	Tipo	Restricciones / relación
id	bigint UNSIGNED	PK autoincremental
nombre	string	—
slug	string	único
icono	string	default cube
descripcion	string	nullable
grupo	string	default Academico
orden	unsignedInteger	default 0
nucleo	boolean	default true
created_at / updated_at	timestamp	gestionados por Eloquent

Tabla `empresa_modulo` · migración `0001_01_01_000003_create_modulos_table.php`

Columna	Tipo	Restricciones / relación
id	bigint UNSIGNED	PK autoincremental
empresa_id	bigint UNSIGNED (FK)	→ empresas, cascade
modulo_id	bigint UNSIGNED (FK)	→ modulos, cascade
activo	boolean	default true
created_at / updated_at	timestamp	gestionados por Eloquent

Únicos: `empresa_id, modulo_id`

Cómo se construye el menú en tiempo de ejecución

- 1 AppServiceProvider registra un View Composer**
Un `View::composer('*)` se ejecuta antes de renderizar cualquier vista.
- 2 Consulta los módulos activos de la institución**
Cruza `modulos` con `empresa_modulo` filtrando por `activo = 1` y por la institución del usuario.
- 3 Comparte la variable `$menuModulos`**
Queda disponible en todas las vistas, ordenada por grupo y por el campo `orden`.
- 4 El sidebar la recorre y resuelve la URL**
Un `match($modulo->slug)` en `partials\sidebar.blade.php` traduce cada slug a su ruta. Lo no contemplado cae en la ruta genérica `modulo.show`.

Doble barrera para los módulos

El menú **oculta** lo que no está activo, pero eso es sólo cosmética. La protección real es el middleware `modulo:slug` declarado en cada grupo de rutas: aunque alguien escriba la URL a mano, no pasa.

4. Ciclo de vida de una petición

Recorrido completo de una petición típica, tomando como ejemplo la apertura del listado de estudiantes en `GET /panel/estudiantes`.

- 1 El servidor web entrega la petición a `public/index.php`**
Único punto de entrada. Arranca el framework leyendo `bootstrap/app.php`.
- 2 Se resuelve la ruta en `routes/web.php`**
Coincide con el grupo `prefix('panel/estudiantes')->name('estudiantes.')` y la acción `EstudianteController::index`.
- 3 Se ejecuta la cadena de middleware**
Primero `auth` comprueba la sesión; luego `modulo:estudiantes` verifica que la institución tenga el módulo activo.
- 4 El controlador construye la consulta**
Aplica `deMiEmpresa()`, luego los scopes de búsqueda y filtro, y finalmente `paginate()`.
- 5 Eloquent traduce a SQL y consulta MySQL**
La cláusula `where empresa_id = ?` viaja siempre incorporada.
- 6 El `AppServiceProvider` inyecta el menú**
El View Composer añade `$menuModulos` antes de renderizar.
- 7 Blade compila y renderiza la vista**
`estudiantes\index.blade.php` extiende `layouts/app.blade.php`. El resultado compilado se cachea en `storage/framework/views\`.
- 8 Se devuelve el HTML al navegador**
Tailwind llega por CDN y Chart.js sólo en las pantallas con gráficos.

Nota sobre la caché de vistas

Blade compila cada vista a PHP plano y la guarda en `storage/framework/views\`. Si editas una vista y el cambio no aparece, ejecuta `php artisan view:clear`. Esto es especialmente relevante en Windows, donde la comparación por fecha de modificación puede fallar.

5. Modelo de datos completo

La base de datos `suite_saas_educacion_integral` contiene **32 tablas** y **349 columnas** documentadas. Se agrupan en seis dominios funcionales.

Las tablas por dominio

Dominio	Tablas	Propósito
Plataforma SaaS	<code>empresas</code> , <code>tipos_negocio</code> , <code>planes</code> , <code>suscripciones</code> , <code>modulos</code> , <code>empresa_modulo</code>	Definen la estructura comercial: quién es cliente, qué contrató y qué puede ver.
Identidad y sesión	<code>users</code> , <code>sessions</code> , <code>password_reset_tokens</code> , <code>cache</code> , <code>cache_locks</code>	Usuarios y tablas de infraestructura del framework.
Académico	<code>estudiantes</code> , <code>docentes</code> , <code>programas</code> , <code>matriculas</code> , <code>aulas</code> , <code>horarios</code> , <code>asistencias</code> , <code>evaluaciones</code> , <code>calificaciones</code> , <code>certificados</code>	El núcleo de la operación educativa.
Financiero	<code>pagos</code> , <code>cajas</code> , <code>movimientos_caja</code>	Cobranza a estudiantes y control de caja por turno.
Administrativo	<code>prospectos</code> , <code>comunicados</code> , <code>empleados</code> , <code>articulos</code> , <code>movimientos_inventario</code>	Captación, comunicación, personal y almacén.
Contenidos	<code>recursos</code> , <code>libros</code> , <code>prestamos</code>	Campus virtual y biblioteca.

Mapa de relaciones principales

Cadena de dependencias (→ significa "es padre de")

```
tipos_negocio → empresas
planes → empresas → suscripciones
empresas → users
empresas → empresa_modulo ← modulos

empresas → programas → matriculas ← estudiantes
→ horarios ← docentes, aulas
→ evaluaciones → calificaciones ← estudiantes
→ recursos

horarios → asistencias ← estudiantes
estudiantes → pagos
estudiantes → certificados
estudiantes → prospectos (conversión opcional)
estudiantes → prestamos ← libros

users → cajas → movimientos_caja
users → comunicados
```

Regla de borrado en cascada

Todas las claves foráneas hacia **empresas** usan **cascadeOnDelete**. Eliminar una institución elimina absolutamente todos sus datos en una sola operación. Las claves hacia catálogos opcionales (programa, docente, aula) usan **nullOnDelete** para no perder el registro histórico cuando se da de baja un catálogo.

6. Catálogo de módulos: ficheros, tablas y procesos

Cada módulo se documenta con la misma estructura: **propósito** → **proceso interno** → **ficheros que participan** → **modelo y tablas** → **rutas expuestas**. Los datos provienen directamente del código fuente.

Cómo leer las fichas

La sección **Ficheros** lista las rutas reales dentro del proyecto. La sección **Tablas** muestra el esquema tal como lo crea la migración, con tipos y restricciones. La sección **Rutas** incluye el verbo HTTP, la URI, el nombre de ruta, la acción del controlador y el middleware que la protege.

Módulos del panel Super Admin SaaS

6.1 Empresas (instituciones)

Alta y gestión del ciclo de vida de cada institución cliente. Es el módulo que da origen a todo el resto: sin una empresa creada no existe ningún dato operativo.

Proceso interno

1

Alta

El formulario valida los datos con `EmpresaRequest` y crea el registro en `empresas` con su `tipo_negocio_id` y `plan_id`.

2

Copia de módulos del plan

Al asignar el plan, el sistema replica su lista de módulos en la tabla pivote `empresa_modulo` con `activo = 1`.

3

Listado con conteos

El índice usa `withCount()` con alias para contar usuarios y módulos activos sin cargar las colecciones completas.

4

Suspensión

`toggle()` invierte la bandera `activo`. El usuario de esa institución deja de poder operar.

5

Eliminación

El borrado en cascada de MySQL elimina todos los registros dependientes en una sola transacción.

Ficheros que participan

Rutas de fichero

```
app\Http\Controllers\Admin\EmpresaController.php
app\Models\Empresa.php
app\Models\TipoNegocio.php
database/migrations/0001_01_01_000001_create_empresas_table.php
database/migrations/0001_01_01_000000_create_tipos_negocio_table.php
resources\views\admin\empresas\form.blade.php
resources\views\admin\empresas\index.blade.php
resources\views\admin\empresas\modulos.blade.php
resources\views\admin\empresas\show.blade.php
resources\views\admin\empresas\suscripcion.blade.php
resources\views\admin\empresas\ticket.blade.php
routes\web.php (grupo admin.empresas)
```

Modelo Empresa	Detalle
Fichero	app\Models\Empresa.php
Tabla	(convención)
Campos asignables	tipo_negocio_id, plan_id, nombre, razon_social, slug, ruc, email, telefono, direccion, ciudad, logo, color_primario, moneda, simbolo, plan, activo
Relaciones	tipoNegocio() — belongsTo(TipoNegocio) planActual() — belongsTo(Plan) suscripciones() — hasMany(Suscripcion) usuarios() — hasMany(User) modulos() — belongsToMany(Modulo)
Métodos propios	suscripcionActiva(), modulosActivos()

Modelo TipoNegocio	Detalle
Fichero	app\Models\TipoNegocio.php
Tabla	tipos_negocio
Campos asignables	nombre, slug, descripcion, icono, activo
Relaciones	empresas() — hasMany(Empresa)

Tablas de datos

Tabla [empresas](#) · migración `0001_01_01_000001_create_empresas_table.php`

Columna	Tipo	Restricciones / relación
id	bigint UNSIGNED	PK autoincremental
tipo_negocio_id	bigint UNSIGNED (FK)	→ tipos_negocio
nombre	string	—
slug	string	único
ruc	string	nullable
email	string	nullable
telefono	string	nullable
direccion	string	nullable
logo	string	nullable
color_primario	string	default #2f6bff
plan	string	default pro
activo	boolean	default true
created_at / updated_at	timestamp	gestionados por Eloquent

Tabla [tipos_negocio](#) · migración `0001_01_01_000000_create_tipos_negocio_table.php`

Columna	Tipo	Restricciones / relación
id	bigint UNSIGNED	PK autoincremental
nombre	string	—
slug	string	único
descripcion	string	nullable
icono	string	default academic
activo	boolean	default true
created_at / updated_at	timestamp	gestionados por Eloquent

Rutas expuestas

Verbo	URI	Nombre de ruta	Acción	Middleware
GET	/admin/empresas/crear	admin.empresas.create	AdminEmpresaController::create()	—
POST	/admin/empresas	admin.empresas.store	AdminEmpresaController::store()	—
GET	/admin/empresas/{empresa}	admin.empresas.show	AdminEmpresaController::show()	—
GET	/admin/empresas/{empresa}/editar	admin.empresas.edit	AdminEmpresaController::edit()	—
PUT	/admin/empresas/{empresa}	admin.empresas.update	AdminEmpresaController::update()	—
PATCH	/admin/empresas/{empresa}/toggle	admin.empresas.toggle	AdminEmpresaController::toggle()	—
DELETE	/admin/empresas/{empresa}	admin.empresas.destroy	AdminEmpresaController::destroy()	—
GET	/admin/empresas/{empresa}/modulos	admin.empresas.modulos	AdminEmpresaController::modulos()	—
PUT	/admin/empresas/{empresa}/modulos	admin.empresas.modulos.update	AdminEmpresaController::updateModulos()	—
GET	/admin/empresas/{empresa}/suscripcion	admin.empresas.suscripcion	AdminEmpresaController::suscripcionForm()	—
POST	/admin/empresas/{empresa}/suscripcion	admin.empresas.suscripcion.store	AdminEmpresaController::suscripcionStore()	—
GET	/admin/empresas/{empresa}/suscripcion/{suscripcion}/ticket	admin.empresas.suscripcion.ticket	AdminEmpresaController::ticket()	—

Nota técnica

El conteo de módulos activos requiere calificar la columna del pivote: `fn (\$q) =>`

`\$q->where('empresa_modulo.activo', true)`. Sin el prefijo de tabla, MySQL responde `Unknown column 'pivot'`. Además, la relación hacia el plan se llama `planActual()` y no `plan()`, porque la tabla tiene una columna de texto llamada `plan` que ensombrece la relación.

6.2 Planes de suscripción

Define el catálogo comercial: precio, límites y la lista de módulos que cada paquete habilita.

Proceso interno

1

Listado

Consulta los planes con `withCount('empresas')` para mostrar cuántos clientes tiene cada uno.

2

Alta y edición

La lista de módulos se guarda como **JSON** en la columna `modulos`, gracias al cast `'modulos' => 'array'` del modelo.

3

Efecto sobre las empresas

Cambiar un plan no re-sincroniza a los clientes existentes: sus módulos ya viven en `empresa_modulo`. El cambio aplica a nuevas asignaciones.

4

Eliminación

Bloqueada si el plan tiene instituciones asociadas, para no dejar huérfanos.

Ficheros que participan

Rutas de fichero

```
app\Http\Controllers\Admin\PlanController.php
app\Models\Plan.php
database\migrations\0001_01_01_000011_create_planes_table.php
resources\views\admin\planes\form.blade.php
resources\views\admin\planes\index.blade.php
routes\web.php (grupo admin.planes)
```

Modelo Plan	Detalle
Fichero	app\Models\Plan.php
Tabla	planes
Campos asignables	nombre, slug, descripcion, precio_mensual, precio_anual, limite_estudiantes, limite_usuarios, modulos, destacado, activo, orden
Relaciones	empresas() — hasMany(Empresa) suscripciones() — hasMany(Suscripcion)
Métodos propios	limiteEstudiantesTexto(), limiteUsuariosTexto()

Tablas de datos

Tabla `planes` · migración `0001_01_01_000011_create_planes_table.php`

Columna	Tipo	Restricciones / relación
id	bigint UNSIGNED	PK autoincremental
nombre	string	—

Columna	Tipo	Restricciones / relación
slug	string	único
descripcion	string	nullable
precio_mensual	decimal,10,2	default 0
precio_anual	decimal,10,2	default 0
limite_estudiantes	unsignedInteger	nullable
limite_usuarios	unsignedInteger	nullable
modulos	json	nullable
destacado	boolean	default false
activo	boolean	default true
orden	unsignedInteger	default 0
created_at / updated_at	timestamp	gestionados por Eloquent

Rutas expuestas

Verbo	URI	Nombre de ruta	Acción	Middleware
GET	/admin/planes/crear	admin.planes.create	PlanController::create()	—
POST	/admin/planes	admin.planes.store	PlanController::store()	—
GET	/admin/planes/{plan}/editar	admin.planes.edit	PlanController::edit()	—
PUT	/admin/planes/{plan}	admin.planes.update	PlanController::update()	—
DELETE	/admin/planes/{plan}	admin.planes.destroy	PlanController::destroy()	—

6.3 Suscripciones

Registra el contrato vigente entre la plataforma y cada institución, con su historial completo.

Proceso interno

- 1 Cálculo automático del precio**
Al elegir plan y ciclo, la vista toma `precio_mensual` o `precio_anual` del plan seleccionado.
- 2 Cálculo de la fecha de fin**
Se deriva de la fecha de inicio sumando un mes o un año según el ciclo.
- 3 Historial no destructivo**
Cada renovación inserta una fila nueva. La suscripción vigente se obtiene con `suscripcionActiva()`, que filtra por estado y toma la más reciente.
- 4 Ticket**
Vista imprimible del comprobante de suscripción.

Ficheros que participan

Rutas de fichero

```
app\Http\Controllers\Admin\SuscripcionController.php
app\Models\Suscripcion.php
database\migrations\0001_01_01_000013_create_suscripciones_table.php
resources\views\admin\suscripciones\index.blade.php
routes\web.php (grupo admin.suscripciones)
```

Modelo Suscripcion	Detalle
Fichero	app\Models\Suscripcion.php
Tabla	suscripciones
Campos asignables	empresa_id, plan_id, ciclo, precio, estado, fecha_inicio, fecha_fin
Relaciones	empresa() — belongsTo(Empresa) plan() — belongsTo(Plan)
Scopes	scopeVigente
Constantes	ESTADOS
Métodos propios	mrr(), diasRestantes()

Tablas de datos

Tabla `suscripciones` · migración `0001_01_01_000013_create_suscripciones_table.php`

Columna	Tipo	Restricciones / relación
id	bigint UNSIGNED	PK autoincremental
empresa_id	bigint UNSIGNED (FK)	→ empresas, cascade

Columna	Tipo	Restricciones / relación
plan_id	bigint UNSIGNED (FK)	nullable, → planes, set null
ciclo	string	default mensual
precio	decimal,10,2	default 0
estado	string	default prueba
fecha_inicio	date	nullable
fecha_fin	date	nullable
created_at / updated_at	timestamp	gestionados por Eloquent

Índices: [empresa_id](#), [estado](#)

Rutas expuestas

Verbo	URI	Nombre de ruta	Acción	Middleware
PATCH	/admin/suscripciones/{suscripcion}/estado	admin.suscripciones.estado	SuscripcionController::cambiarEstado()	—

6.4 Mantenimiento de la plataforma

Copia de seguridad, restauración y reseteo a nivel de toda la plataforma. Es el módulo más sensible del sistema.

Proceso interno

1 Inventario del esquema

Ejecuta `SHOW TABLES` y descarta las tablas transitorias (`sessions`, `cache`, `cache_locks`).

2 Volcado de estructura

Para la copia completa emite `DROP TABLE IF EXISTS` seguido del resultado de `SHOW CREATE TABLE` aplanado a una línea.

3 Volcado de datos

Recorre cada tabla por lotes y escapa cada valor con `$pdo->quote()`, generando sentencias `INSERT` agrupadas de 200 filas.

4 Copia por institución

Filtra por `empresa_id` y antepone un `DELETE` acotado. Las calificaciones, que no tienen `empresa_id`, se filtran por subconsulta sobre `evaluaciones`.

5 Restauración

Valida la firma del archivo, desactiva `FOREIGN_KEY_CHECKS`, separa las sentencias y las ejecuta dentro de un bloque protegido.

6 Reseteo

Elimina por grupos de tablas (hijas primero) filtrando por institución, conservando empresa, usuarios, plan y módulos.

Ficheros que participan

Rutas de fichero

```
app\Http\Controllers\Admin\MantenimientoController.php
resources\views\admin\mantenimiento\index.blade.php
routes\web.php (grupo admin.mantenimiento)
```

Tablas de datos

Rutas expuestas

Verbo	URI	Nombre de ruta	Acción	Middleware
POST	/admin/mantenimiento/backup	admin.mantenimiento.backup	AdminMantenimientoController::backup()	—
GET	/admin/mantenimiento/backup/{archivo}/descargar	admin.mantenimiento.descargar	AdminMantenimientoController::descargar()	—
DELETE	/admin/mantenimiento/backup/{archivo}	admin.mantenimiento.eliminar	AdminMantenimientoController::eliminar()	—

Verbo	URI	Nombre de ruta	Acción	Middleware
POST	/admin/mantenimiento/restaurar	admin.mantenimiento.restaurar	AdminMantenimientoController::restaurar()	—
POST	/admin/mantenimiento/reset	admin.mantenimiento.reset	AdminMantenimientoController::reset()	—

Nota técnica

El volcado está escrito en PHP puro y **no depende de `mysqldump`**. Esto permite generar copias desde el navegador en hostings compartidos donde no hay acceso al binario de MySQL. El coste es mayor consumo de memoria en bases muy grandes, mitigado con el procesamiento por lotes.

Módulos del dominio académico

6.5 Admisiones y matrículas

Vincula a un estudiante con un programa y, opcionalmente, genera de forma automática los cargos económicos derivados. Es el módulo con la lógica de negocio más rica del sistema.

Proceso interno

1 Validación

Comprueba que estudiante y programa pertenezcan a la institución del usuario antes de continuar.

2 Transacción atómica

Todo el proceso corre dentro de `DB::transaction()`: si algo falla, no queda una matrícula sin cargos ni cargos sin matrícula.

3 Generación del código

Formato `MAT-{año}-{correlativo}` con relleno a cuatro dígitos, calculado sobre el total de matrículas de la institución.

4 Sincronización del estudiante

Actualiza su programa, nivel, estado y fecha de ingreso.

5 Cargo de matrícula

Si se marcó la opción, crea un registro en `pagos` con concepto `matricula` y estado `pendiente`.

6 Generación de pensiones

Bucle de hasta 12 iteraciones que crea una cuota por mes, desplazando la fecha de vencimiento con `addMonths($i)`.

7 Anulación en cascada

Marca la matrícula como anulada, registra el motivo en la observación y anula los pagos pendientes que tengan `monto_pagado = 0`. Lo ya cobrado no se toca.

8 Reactivación

Sólo permitida sobre matrículas anuladas; devuelve el estado a `matriculado`.

Ficheros que participan

Rutas de fichero

```
app\Http\Controllers\AdmissionController.php
app\Models\Matricula.php
database\migrations\0001_01_01_000018_create_matriculas_table.php
resources\views\admisiones\create.blade.php
resources\views\admisiones\edit.blade.php
resources\views\admisiones\index.blade.php
```

```
resources\views\admisiones\show.blade.php
routes\web.php (grupo admisiones)
```

Modelo Matricula	Detalle
Fichero	app\Models\Matricula.php
Tabla	matriculas
Campos asignables	empresa_id, estudiante_id, programa_id, codigo, curso, nivel, periodo, fecha, monto_matricula, estado, observacion
Relaciones	empresa() — belongsTo(Empresa) estudiante() — belongsTo(Estudiante) programa() — belongsTo(Programa)
Scopes	scopeDeMiEmpresa, scopeBuscar
Constantes	ESTADOS
Métodos propios	colorEstado()

Tablas de datos

Tabla `matriculas` · migración `0001_01_01_000018_create_matriculas_table.php`

Columna	Tipo	Restricciones / relación
id	bigint UNSIGNED	PK autoincremental
empresa_id	bigint UNSIGNED (FK)	→ empresas, cascade
estudiante_id	bigint UNSIGNED (FK)	→ estudiantes, cascade
programa_id	bigint UNSIGNED (FK)	nullable, → programas, set null
codigo	string	nullable
curso	string	nullable
nivel	string	nullable
periodo	string	nullable
fecha	date	nullable
monto_matricula	decimal,10,2	default 0
estado	string	default matriculado
observacion	string	nullable
created_at / updated_at	timestamp	gestionados por Eloquent

Índices: `empresa_id`, `estado`

Rutas expuestas

Verbo	URI	Nombre de ruta	Acción	Middleware
GET	/panel/admisiones	admisiones.index	AdmisionController::index()	—
GET	/panel/admisiones/matricular	admisiones.create	AdmisionController::create()	—
POST	/panel/admisiones	admisiones.store	AdmisionController::store()	—
GET	/panel/admisiones/{matricula}	admisiones.show	AdmisionController::show()	—
GET	/panel/admisiones/{matricula}/editar	admisiones.edit	AdmisionController::edit()	—
PUT	/panel/admisiones/{matricula}	admisiones.update	AdmisionController::update()	—
PATCH	/panel/admisiones/{matricula}/anular	admisiones.anular	AdmisionController::anular()	—
PATCH	/panel/admisiones/{matricula}/reactivar	admisiones.reactivar	AdmisionController::reactivar()	—

Nota técnica

La generación automática de cargos es el punto de integración entre el dominio académico y el financiero. Es también el lugar donde más conviene ser cuidadoso al modificar código: un fallo aquí produce cuotas duplicadas o inexistentes.

6.6 Estudiantes

Ficha maestra del alumnado con búsqueda, filtro por estado, paginación y datos del apoderado.

Proceso interno

- 1 Listado**
Encadena `deMiEmpresa()`, `buscar($q)`, `estado($e)` y `paginate()`, preservando los filtros con `withQueryString()`.
- 2 Búsqueda**
El scope `scopeBuscar` agrupa las condiciones **OR** dentro de un closure para no romper el filtro de institución.
- 3 Alta y edición**
Validación de unicidad del documento acotada a la institución.
- 4 Atributos calculados**
`nombre_completo` concatena nombres y apellidos; `edad()` se deriva de la fecha de nacimiento; `colorEstado()` devuelve las clases CSS del distintivo.

Ficheros que participan

Rutas de fichero

```
app\Http\Controllers\EstudianteController.php
app\Models\Estudiante.php
database\migrations\0001_01_01_000005_create_estudiantes_table.php
resources\views\estudiantes\form.blade.php
resources\views\estudiantes\index.blade.php
resources\views\estudiantes\show.blade.php
routes\web.php (grupo estudiantes)
```

Modelo Estudiante	Detalle
Fichero	app\Models\Estudiante.php
Tabla	estudiantes
Campos asignables	empresa_id, codigo, nombres, apellidos, tipo_documento, documento, email, telefono, fecha_nacimiento, genero, direccion, programa, nivel, estado, fecha_ingreso, apoderado, apoderado_telefono, foto, observaciones
Relaciones	empresa() — belongsTo(Empresa)
Scopes	scopeDeMiEmpresa, scopeBuscar, scopeEstado
Constantes	ESTADOS
Métodos propios	getNombreCompletoAttribute(), iniciales(), edad(), colorEstado()

Tablas de datos

Tabla `estudiantes` · migración `0001_01_01_000005_create_estudiantes_table.php`

Columna	Tipo	Restricciones / relación
id	bigint UNSIGNED	PK autoincremental
empresa_id	bigint UNSIGNED (FK)	→ empresas, cascade
codigo	string	nullable
nombres	string	—
apellidos	string	—
tipo_documento	string	default DNI
documento	string	nullable
email	string	nullable
telefono	string	nullable
fecha_nacimiento	date	nullable
genero	string	nullable
direccion	string	nullable
programa	string	nullable
nivel	string	nullable
estado	string	default matriculado
fecha_ingreso	date	nullable
apoderado	string	nullable
apoderado_telefono	string	nullable
foto	string	nullable
observaciones	text	nullable
created_at / updated_at	timestamp	gestionados por Eloquent

Índices: [empresa_id](#), [estado](#)

Rutas expuestas

Verbo	URI	Nombre de ruta	Acción	Middleware
GET	/panel/estudiantes	estudiantes.index	EstudianteController::index()	—
GET	/panel/estudiantes/crear	estudiantes.create	EstudianteController::create()	—
POST	/panel/estudiantes	estudiantes.store	EstudianteController::store()	—
GET	/panel/estudiantes/{estudiante}	estudiantes.show	EstudianteController::show()	—
GET	/panel/estudiantes/{estudiante}/editar	estudiantes.edit	EstudianteController::edit()	—

Verbo	URI	Nombre de ruta	Acción	Middleware
PUT	/panel/estudiantes/{estudiante}	estudiantes.update	EstudianteController::update()	—
DELETE	/panel/estudiantes/{estudiante}	estudiantes.destroy	EstudianteController::destroy()	—

6.7 Docentes

Plana docente con profesión, especialidad, tipo de contrato y estado.

Proceso interno

1

Listado y búsqueda

Mismo patrón de scopes encadenados que Estudiantes.

2

Uso desde otros módulos

Horarios y Evaluaciones referencian `docente_id` con `nullOnDelete`, de modo que dar de baja a un docente no borra el historial.

Ficheros que participan

Rutas de fichero

```
app\Http\Controllers\DocenteController.php
app\Models\Docente.php
database\migrations\0001_01_01_000007_create_docentes_table.php
resources\views\docentes\form.blade.php
resources\views\docentes\index.blade.php
resources\views\docentes\show.blade.php
routes\web.php (grupo docentes)
```

Modelo Docente	Detalle
Fichero	app\Models\Docente.php
Tabla	docentes
Campos asignables	empresa_id, codigo, nombres, apellidos, tipo_documento, documento, email, telefono, profesion, especialidad, tipo_contrato, fecha_ingreso, estado, foto, observaciones
Relaciones	empresa() — belongsTo(Empresa)
Scopes	scopeDeMiEmpresa, scopeBuscar, scopeEstado
Constantes	CONTRATOS, ESTADOS
Métodos propios	getNombreCompletoAttribute(), iniciales(), etiquetaContrato()

Tablas de datos

Tabla `docentes` · migración `0001_01_01_000007_create_docentes_table.php`

Columna	Tipo	Restricciones / relación
<code>id</code>	bigint UNSIGNED	PK autoincremental
<code>empresa_id</code>	bigint UNSIGNED (FK)	→ empresas, cascade
<code>codigo</code>	string	nullable

Columna	Tipo	Restricciones / relación
nombres	string	—
apellidos	string	—
tipo_documento	string	default DNI
documento	string	nullable
email	string	nullable
telefono	string	nullable
profesion	string	nullable
especialidad	string	nullable
tipo_contrato	string	default planilla
fecha_ingreso	date	nullable
estado	string	default activo
foto	string	nullable
observaciones	text	nullable
created_at / updated_at	timestamp	gestionados por Eloquent

Índices: `empresa_id`, `estado`

Rutas expuestas

Verbo	URI	Nombre de ruta	Acción	Middleware
GET	<code>/panel/docentes</code>	<code>docentes.index</code>	<code>DocenteController::index()</code>	—
GET	<code>/panel/docentes/crear</code>	<code>docentes.create</code>	<code>DocenteController::create()</code>	—
POST	<code>/panel/docentes</code>	<code>docentes.store</code>	<code>DocenteController::store()</code>	—
GET	<code>/panel/docentes/{docente}</code>	<code>docentes.show</code>	<code>DocenteController::show()</code>	—
GET	<code>/panel/docentes/{docente}/editar</code>	<code>docentes.edit</code>	<code>DocenteController::edit()</code>	—
PUT	<code>/panel/docentes/{docente}</code>	<code>docentes.update</code>	<code>DocenteController::update()</code>	—
DELETE	<code>/panel/docentes/{docente}</code>	<code>docentes.destroy</code>	<code>DocenteController::destroy()</code>	—

6.8 Programas y cursos

El catálogo académico. Es la entidad central del modelo de datos: matrículas, horarios, evaluaciones y recursos dependen de ella.

Proceso interno

1

Tipo polimórfico

El campo `tipo` admite carrera, curso, taller, idioma, programa o diplomado. Es el mecanismo que adapta el sistema a los nueve tipos de negocio.

2

Precio de referencia

El campo `precio` se propone automáticamente al matricular, tanto para la matrícula como para las pensiones.

3

Control de cupo

El campo `cupo` se contrasta con las matrículas activas para advertir cuando se agota.

Ficheros que participan

Rutas de fichero

```
app\Http\Controllers\ProgramaController.php
app\Models\Programa.php
database\Migrations\0001_01_01_000006_create_programas_table.php
resources\views\programas\form.blade.php
resources\views\programas\index.blade.php
resources\views\programas\show.blade.php
routes\web.php (grupo programas)
```

Modelo Programa	Detalle
Fichero	app\Models\Programa.php
Tabla	programas
Campos asignables	empresa_id, codigo, nombre, tipo, modalidad, nivel, duracion, cupo, precio, coordinador, descripcion, estado
Relaciones	empresa() — belongsTo(Empresa)
Scopes	scopeDeMiEmpresa, scopeBuscar, scopeTipo
Constantes	TIPOS, MODALIDADES, ESTADOS
Métodos propios	matriculados(), colorTipo()

Tablas de datos

Tabla `programas` · migración `0001_01_01_000006_create_programas_table.php`

Columna	Tipo	Restricciones / relación
id	bigint UNSIGNED	PK autoincremental

Columna	Tipo	Restricciones / relación
empresa_id	bigint UNSIGNED (FK)	→ empresas, cascade
codigo	string	nullable
nombre	string	—
tipo	string	default curso
modalidad	string	default presencial
nivel	string	nullable
duracion	string	nullable
cupo	unsignedInteger	nullable
precio	decimal,10,2	default 0
coordinador	string	nullable
descripcion	text	nullable
estado	string	default activo
created_at / updated_at	timestamp	gestionados por Eloquent

Índices: [empresa_id](#), [estado](#)

Rutas expuestas

Verbo	URI	Nombre de ruta	Acción	Middleware
GET	/panel/programas	programas.index	ProgramaController::index()	—
GET	/panel/programas/crear	programas.create	ProgramaController::create()	—
POST	/panel/programas	programas.store	ProgramaController::store()	—
GET	/panel/programas/{programa}	programas.show	ProgramaController::show()	—
GET	/panel/programas/{programa}/editar	programas.edit	ProgramaController::edit()	—
PUT	/panel/programas/{programa}	programas.update	ProgramaController::update()	—
DELETE	/panel/programas/{programa}	programas.destroy	ProgramaController::destroy()	—

6.9 Aulas y ambientes

Inventario de espacios físicos y virtuales donde se dictan las clases.

Proceso interno

1

Tipos

aula, laboratorio, taller, auditorio y virtual. El tipo virtual permite programar clases en línea con la misma mecánica que las presenciales.

2

Uso en Horarios

Cada horario referencia un aula; así se detecta la ocupación de los espacios.

Ficheros que participan

Rutas de fichero

```
app\Http\Controllers\AulaController.php
app\Models\Aula.php
database\migrations\0001_01_01_000009_create_aulas_table.php
resources\views\aulas\form.blade.php
resources\views\aulas\index.blade.php
routes\web.php (grupo aulas)
```

Modelo Aula	Detalle
Fichero	app\Models\Aula.php
Tabla	aulas
Campos asignables	empresa_id, codigo, nombre, tipo, capacidad, ubicacion, equipamiento, estado
Relaciones	empresa() — belongsTo(Empresa) horarios() — hasMany(Horario)
Scopes	scopeDeMiEmpresa, scopeBuscar
Constantes	TIPOS, ESTADOS
Métodos propios	etiquetaTipo()

Tablas de datos

Tabla aulas · migración 0001_01_01_000009_create_aulas_table.php

Columna	Tipo	Restricciones / relación
id	bigint UNSIGNED	PK autoincremental
empresa_id	bigint UNSIGNED (FK)	→ empresas, cascade
codigo	string	nullable
nombre	string	—
tipo	string	default aula

Columna	Tipo	Restricciones / relación
capacidad	unsignedInteger	default 0
ubicacion	string	nullable
equipamiento	text	nullable
estado	string	default disponible
created_at / updated_at	timestamp	gestionados por Eloquent

Índices: `empresa_id`, `estado`

Rutas expuestas

Verbo	URI	Nombre de ruta	Acción	Middleware
GET	<code>/panel/aulas</code>	<code>aulas.index</code>	<code>AulaController::index()</code>	—
GET	<code>/panel/aulas/crear</code>	<code>aulas.create</code>	<code>AulaController::create()</code>	—
POST	<code>/panel/aulas</code>	<code>aulas.store</code>	<code>AulaController::store()</code>	—
GET	<code>/panel/aulas/{aula}/editar</code>	<code>aulas.edit</code>	<code>AulaController::edit()</code>	—
PUT	<code>/panel/aulas/{aula}</code>	<code>aulas.update</code>	<code>AulaController::update()</code>	—
DELETE	<code>/panel/aulas/{aula}</code>	<code>aulas.destroy</code>	<code>AulaController::destroy()</code>	—

6.10 Horarios

Programación semanal que cruza programa, docente, aula, día y rango horario. Es el insumo del módulo de Asistencia.

Proceso interno

1 Modelo de datos

El día se almacena como entero de 1 a 7 con la constante `DIAS` del modelo como diccionario. Las horas usan el tipo `time` de MySQL.

2 Vista de calendario

La vista agrupa por día y ordena por hora de inicio para dibujar la rejilla.

3 Clases de hoy

El dashboard consulta los horarios cuyo día coincide con el día actual.

Ficheros que participan

Rutas de fichero

```
app\Http\Controllers\HorarioController.php
app\Models\Horario.php
database\Migrations\0001_01_01_000010_create_horarios_table.php
resources\views\horarios\form.blade.php
resources\views\horarios\index.blade.php
routes\web.php (grupo horarios)
```

Modelo Horario	Detalle
Fichero	app\Models\Horario.php
Tabla	horarios
Campos asignables	empresa_id, programa_id, docente_id, aula_id, curso, seccion, dia, hora_inicio, hora_fin, estado
Relaciones	empresa() — belongsTo(Empresa) programa() — belongsTo(Programa) docente() — belongsTo(Docente) aula() — belongsTo(Aula)
Scopes	scopeDeMiEmpresa
Constantes	DIAS
Métodos propios	nombreDia(), rango(), titulo()

Tablas de datos

Tabla `horarios` · migración `0001_01_01_000010_create_horarios_table.php`

Columna	Tipo	Restricciones / relación
id	bigint UNSIGNED	PK autoincremental

Columna	Tipo	Restricciones / relación
empresa_id	bigint UNSIGNED (FK)	→ empresas, cascade
programa_id	bigint UNSIGNED (FK)	nullable, → programas, set null
docente_id	bigint UNSIGNED (FK)	nullable, → docentes, set null
aula_id	bigint UNSIGNED (FK)	nullable, → aulas, set null
curso	string	nullable
seccion	string	nullable
dia	unsignedTinyInteger	default 1
hora_inicio	time	—
hora_fin	time	—
estado	string	default activo
created_at / updated_at	timestamp	gestionados por Eloquent

Índices: `empresa_id`, `dia`

Rutas expuestas

Verbo	URI	Nombre de ruta	Acción	Middleware
GET	<code>/panel/horarios</code>	<code>horarios.index</code>	<code>HorarioController::index()</code>	—
GET	<code>/panel/horarios/crear</code>	<code>horarios.create</code>	<code>HorarioController::create()</code>	—
POST	<code>/panel/horarios</code>	<code>horarios.store</code>	<code>HorarioController::store()</code>	—
GET	<code>/panel/horarios/{horario}/editar</code>	<code>horarios.edit</code>	<code>HorarioController::edit()</code>	—
PUT	<code>/panel/horarios/{horario}</code>	<code>horarios.update</code>	<code>HorarioController::update()</code>	—
DELETE	<code>/panel/horarios/{horario}</code>	<code>horarios.destroy</code>	<code>HorarioController::destroy()</code>	—

6.11 Asistencia

Registro diario de la presencia de los estudiantes en cada clase programada.

Proceso interno

1 Selección de la clase

El índice lista los horarios del día actual como puntos de entrada.

2 Carga de la lista

Al abrir una clase, obtiene los estudiantes matriculados en el programa de ese horario.

3 Registro masivo

El formulario envía todos los estados de una vez; el controlador los guarda con `updateOrCreate()` para permitir la corrección posterior sin duplicar filas.

4 Estados

presente, tarde, ausente y justificado.

Ficheros que participan

Rutas de fichero

```
app\Http\Controllers\AsistenciaController.php
app\Models\Asistencia.php
database\migrations\0001_01_01_000015_create_asistencias_table.php
resources\views\asistencia\index.blade.php
resources\views\asistencia\tomar.blade.php
routes\web.php (grupo asistencia)
```

Modelo Asistencia	Detalle
Fichero	app\Models\Asistencia.php
Tabla	asistencias
Campos asignables	empresa_id, horario_id, estudiante_id, fecha, estado, observacion
Relaciones	empresa() — belongsTo(Empresa) horario() — belongsTo(Horario) estudiante() — belongsTo(Estudiante)
Scopes	scopeDeMiEmpresa
Constantes	ESTADOS

Tablas de datos

Tabla `asistencias` · migración `0001_01_01_000015_create_asistencias_table.php`

Columna	Tipo	Restricciones / relación
id	bigint UNSIGNED	PK autoincremental

Columna	Tipo	Restricciones / relación
empresa_id	bigint UNSIGNED (FK)	→ empresas, cascade
horario_id	bigint UNSIGNED (FK)	nullable, → horarios, set null
estudiante_id	bigint UNSIGNED (FK)	→ estudiantes, cascade
fecha	date	—
estado	string	default presente
observacion	string	nullable
created_at / updated_at	timestamp	gestionados por Eloquent

Índices: `empresa_id, fecha` | Únicos: `horario_id, estudiante_id, fecha`

Rutas expuestas

Verbo	URI	Nombre de ruta	Acción	Middleware
GET	<code>/panel/asistencia</code>	<code>asistencia.index</code>	<code>AsistenciaController::index()</code>	—
GET	<code>/panel/asistencia/clase/{horario}</code>	<code>asistencia.tomar</code>	<code>AsistenciaController::tomar()</code>	—
POST	<code>/panel/asistencia/clase/{horario}</code>	<code>asistencia.store</code>	<code>AsistenciaController::store()</code>	—

Nota técnica

La clave única sobre (`horario_id`, `estudiante_id`, `fecha`) impide el doble registro de la misma clase, y es lo que hace seguro el uso de `updateOrCreate()`.

6.12 Evaluaciones y calificaciones

Gestión de exámenes y notas. Usa dos tablas: la cabecera de la evaluación y el detalle por estudiante.

Proceso interno

- 1 Creación**
Se define nombre, tipo, fecha, nota máxima y peso relativo.
- 2 Pantalla de calificación**
Carga a todos los matriculados del programa y presenta un campo de nota por fila, en una sola pantalla.
- 3 Guardado masivo**
Recorre el arreglo enviado y aplica `updateOrCreate()` sobre `calificaciones`.
- 4 Cálculos derivados**
`notaAprobatoria()` obtiene el umbral a partir de la nota máxima; el promedio y la tasa de aprobación se calculan al vuelo para el reporte de rendimiento.

Ficheros que participan

Rutas de fichero

```
app\Http\Controllers\EvaluacionController.php
app\Models\Evaluacion.php
app\Models\Calificacion.php
database\Migrations\0001_01_01_000016_create_evaluaciones_table.php
database\Migrations\0001_01_01_000017_create_calificaciones_table.php
resources\views\evaluaciones\calificar.blade.php
resources\views\evaluaciones\form.blade.php
resources\views\evaluaciones\index.blade.php
resources\views\evaluaciones\show.blade.php
routes\web.php (grupo evaluaciones)
```

Modelo Evaluacion	Detalle
Fichero	app\Models\Evaluacion.php
Tabla	evaluaciones
Campos asignables	empresa_id, programa_id, docente_id, curso, nombre, tipo, fecha, nota_maxima, peso, estado
Relaciones	empresa() — belongsTo(Empresa) programa() — belongsTo(Programa) docente() — belongsTo(Docente) calificaciones() — hasMany(Calificacion)
Scopes	scopeDeMiEmpresa, scopeBuscar
Constantes	TIPOS

Modelo Evaluacion	Detalle
Métodos propios	notaAprobatoria(), promedio(), totalCalificados(), totalAprobados(), etiquetaTipo()

Modelo Calificacion	Detalle
Fichero	app\Models\Calificacion.php
Tabla	calificaciones
Campos asignables	evaluacion_id, estudiante_id, nota, observacion
Relaciones	evaluacion() — belongsTo(Evaluacion) estudiante() — belongsTo(Estudiante)

Tablas de datos

Tabla `evaluaciones` · migración `0001_01_01_000016_create_evaluaciones_table.php`

Columna	Tipo	Restricciones / relación
id	bigint UNSIGNED	PK autoincremental
empresa_id	bigint UNSIGNED (FK)	→ empresas, cascade
programa_id	bigint UNSIGNED (FK)	nullable, → programas, set null
docente_id	bigint UNSIGNED (FK)	nullable, → docentes, set null
curso	string	nullable
nombre	string	—
tipo	string	default examen
fecha	date	nullable
nota_maxima	decimal,5,2	default 20
peso	unsignedTinyInteger	default 1
estado	string	default abierta
created_at / updated_at	timestamp	gestionados por Eloquent

Índices: `empresa_id`, `estado`

Tabla `calificaciones` · migración `0001_01_01_000017_create_calificaciones_table.php`

Columna	Tipo	Restricciones / relación
id	bigint UNSIGNED	PK autoincremental
evaluacion_id	bigint UNSIGNED (FK)	→ evaluaciones, cascade
estudiante_id	bigint UNSIGNED (FK)	→ estudiantes, cascade
nota	decimal,5,2	nullable
observacion	string	nullable

Columna	Tipo	Restricciones / relación
created_at / updated_at	timestamp	gestionados por Eloquent

Únicos: `evaluacion_id`, `estudiante_id`

Rutas expuestas

Verbo	URI	Nombre de ruta	Acción	Middleware
GET	<code>/panel/evaluaciones</code>	<code>evaluaciones.index</code>	<code>EvaluacionController::index()</code>	—
GET	<code>/panel/evaluaciones/crear</code>	<code>evaluaciones.create</code>	<code>EvaluacionController::create()</code>	—
POST	<code>/panel/evaluaciones</code>	<code>evaluaciones.store</code>	<code>EvaluacionController::store()</code>	—
GET	<code>/panel/evaluaciones/{evaluacion}</code>	<code>evaluaciones.show</code>	<code>EvaluacionController::show()</code>	—
GET	<code>/panel/evaluaciones/{evaluacion}/editar</code>	<code>evaluaciones.edit</code>	<code>EvaluacionController::edit()</code>	—
PUT	<code>/panel/evaluaciones/{evaluacion}</code>	<code>evaluaciones.update</code>	<code>EvaluacionController::update()</code>	—
GET	<code>/panel/evaluaciones/{evaluacion}/calificar</code>	<code>evaluaciones.calificar</code>	<code>EvaluacionController::calificar()</code>	—
POST	<code>/panel/evaluaciones/{evaluacion}/calificar</code>	<code>evaluaciones.calificar.store</code>	<code>EvaluacionController::guardarCalificaciones()</code>	—
DELETE	<code>/panel/evaluaciones/{evaluacion}</code>	<code>evaluaciones.destroy</code>	<code>EvaluacionController::destroy()</code>	—

Nota técnica

`calificaciones` es la **única tabla operativa sin columna `empresa_id`**: hereda la pertenencia a través de `evaluacion_id`. Por eso los procesos de copia de seguridad y reseteo la tratan como caso especial, filtrándola con una subconsulta sobre `evaluaciones`.

6.13 Certificados

Emisión de constancias y diplomas con código de verificación y versión imprimible.

Proceso interno

1

Tipos

Cinco tipos definidos como diccionario en la constante **TIPOS** del modelo.

2

Código único

Se genera con un correlativo por institución y se imprime en el documento.

3

Vista imprimible

Plantilla independiente del layout, con reglas `@media print` y los datos e identidad visual de la institución.

4

Anulación

Cambia el estado sin eliminar el registro, preservando la trazabilidad.

Ficheros que participan

Rutas de fichero

```
app\Http\Controllers\CertificadoController.php
app\Models\Certificado.php
database\Migrations\0001_01_01_000019_create_certificados_table.php
resources\views\certificados\create.blade.php
resources\views\certificados\index.blade.php
resources\views\certificados\show.blade.php
routes\web.php (grupo certificados)
```

Modelo Certificado	Detalle
Fichero	app\Models\Certificado.php
Tabla	certificados
Campos asignables	empresa_id, estudiante_id, codigo, tipo, titulo, programa, descripcion, fecha_emision, estado
Relaciones	empresa() — belongsTo(Empresa) estudiante() — belongsTo(Estudiante)
Scopes	scopeDeMiEmpresa, scopeBuscar
Constantes	TIPOS
Métodos propios	etiquetaTipo()

Tablas de datos

Tabla `certificados` · migración `0001_01_01_000019_create_certificados_table.php`

Columna	Tipo	Restricciones / relación
id	bigint UNSIGNED	PK autoincremental
empresa_id	bigint UNSIGNED (FK)	→ empresas, cascade
estudiante_id	bigint UNSIGNED (FK)	→ estudiantes, cascade
codigo	string	nullable
tipo	string	default constancia_estudios
titulo	string	—
programa	string	nullable
descripcion	text	nullable
fecha_emision	date	nullable
estado	string	default emitido
created_at / updated_at	timestamp	gestionados por Eloquent

Índices: `empresa_id`, `tipo`

Rutas expuestas

Verbo	URI	Nombre de ruta	Acción	Middleware
GET	<code>/panel/certificados</code>	<code>certificados.index</code>	<code>CertificadoController::index()</code>	—
GET	<code>/panel/certificados/emitter</code>	<code>certificados.create</code>	<code>CertificadoController::create()</code>	—
POST	<code>/panel/certificados</code>	<code>certificados.store</code>	<code>CertificadoController::store()</code>	—
GET	<code>/panel/certificados/{certificado}</code>	<code>certificados.show</code>	<code>CertificadoController::show()</code>	—
PATCH	<code>/panel/certificados/{certificado}/anular</code>	<code>certificados.anular</code>	<code>CertificadoController::anular()</code>	—
DELETE	<code>/panel/certificados/{certificado}</code>	<code>certificados.destroy</code>	<code>CertificadoController::destroy()</code>	—

6.14 Campus Virtual

Publicación de material, videos, enlaces y tareas por curso.

Proceso interno

1 Publicación

El docente crea un recurso asociado a un programa, con tipo, URL y fechas de publicación y entrega.

2 Tipos

material, video, enlace y tarea. El tipo tarea es el único que usa la fecha de entrega.

3 Visibilidad

Los recursos se filtran por institución y programa.

Ficheros que participan

Rutas de fichero

```
app\Http\Controllers\CampusController.php
app\Models\Recurso.php
database\migrations\0001_01_01_000022_create_recursos_table.php
resources\views\campus\form.blade.php
resources\views\campus\index.blade.php
resources\views\campus\show.blade.php
routes\web.php (grupo campus)
```

Modelo Recurso	Detalle
Fichero	app\Models\Recurso.php
Tabla	recursos
Campos asignables	empresa_id, programa_id, docente_id, curso, titulo, tipo, descripcion, url, fecha_publicacion, fecha_entrega, estado
Relaciones	empresa() — belongsTo(Empresa) programa() — belongsTo(Programa) docente() — belongsTo(Docente)
Scopes	scopeDeMiEmpresa, scopeBuscar
Constantes	TIPOS
Métodos propios	etiquetaTipo(), iconoTipo(), entregaVencida()

Tablas de datos

Tabla `recursos` · migración `0001_01_01_000022_create_recursos_table.php`

Columna	Tipo	Restricciones / relación
id	bigint UNSIGNED	PK autoincremental
empresa_id	bigint UNSIGNED (FK)	→ empresas, cascade

Columna	Tipo	Restricciones / relación
programa_id	bigint UNSIGNED (FK)	nullable, → programas, set null
docente_id	bigint UNSIGNED (FK)	nullable, → docentes, set null
curso	string	nullable
titulo	string	—
tipo	string	default material
descripcion	text	nullable
url	string	nullable
fecha_publicacion	date	nullable
fecha_entrega	date	nullable
estado	string	default publicado
created_at / updated_at	timestamp	gestionados por Eloquent

Índices: `empresa_id`, `tipo`

Rutas expuestas

Verbo	URI	Nombre de ruta	Acción	Middleware
GET	<code>/panel/campus</code>	<code>campus.index</code>	<code>CampusController::index()</code>	—
GET	<code>/panel/campus/nuevo</code>	<code>campus.create</code>	<code>CampusController::create()</code>	—
POST	<code>/panel/campus</code>	<code>campus.store</code>	<code>CampusController::store()</code>	—
GET	<code>/panel/campus/{recurso}</code>	<code>campus.show</code>	<code>CampusController::show()</code>	—
GET	<code>/panel/campus/{recurso}/editar</code>	<code>campus.edit</code>	<code>CampusController::edit()</code>	—
PUT	<code>/panel/campus/{recurso}</code>	<code>campus.update</code>	<code>CampusController::update()</code>	—
DELETE	<code>/panel/campus/{recurso}</code>	<code>campus.destroy</code>	<code>CampusController::destroy()</code>	—

6.15 Biblioteca

Catálogo bibliográfico y control de préstamos con actualización automática de disponibilidad.

Proceso interno

- 1 Catálogo**
Cada libro registra **ejemplares** (total) y **disponibles** (no prestados).
- 2 Préstamo**
Crea el registro y decrementa **disponibles** en la misma operación.
- 3 Devolución**
Registra la fecha real e incrementa **disponibles**.
- 4 Control de vencidos**
Compara la fecha pactada de devolución con la fecha actual.

Ficheros que participan

Rutas de fichero

```
app\Http\Controllers\BibliotecaController.php
app\Models\Libro.php
app\Models\Prestamo.php
database\migrations\0001_01_01_000024_create_biblioteca_tables.php
database\migrations\0001_01_01_000024_create_biblioteca_tables.php
resources\views\biblioteca\index.blade.php
resources\views\biblioteca\libro.blade.php
resources\views\biblioteca\prestamos.blade.php
routes\web.php (grupo biblioteca)
```

Modelo Libro	Detalle
Fichero	app\Models\Libro.php
Tabla	libros
Campos asignables	empresa_id, codigo, titulo, autor, editorial, isbn, categoria, anio, ejemplares, disponibles, ubicacion, estado
Relaciones	empresa() — belongsTo(Empresa) prestamos() — hasMany(Prestamo)
Scopes	scopeDeMiEmpresa, scopeBuscar
Métodos propios	prestados(), hayDisponibles()

Modelo Prestamo	Detalle
Fichero	app\Models\Prestamo.php
Tabla	prestamos

Modelo Prestamo	Detalle
Campos asignables	<code>empresa_id</code> , <code>libro_id</code> , <code>estudiante_id</code> , <code>docente_id</code> , <code>fecha_prestamo</code> , <code>fecha_prevista</code> , <code>fecha_devolucion</code> , <code>estado</code> , <code>observacion</code>
Relaciones	<code>empresa()</code> — <code>belongsTo(Empresa)</code> <code>libro()</code> — <code>belongsTo(Libro)</code> <code>estudiante()</code> — <code>belongsTo(Estudiante)</code> <code>docente()</code> — <code>belongsTo(Docente)</code>
Scopes	<code>scopeDeMiEmpresa</code>
Métodos propios	<code>persona()</code> , <code>tipoPersona()</code> , <code>estaVencido()</code> , <code>diasRetraso()</code>

Tablas de datos

Tabla `libros` · migración `0001_01_01_000024_create_biblioteca_tables.php`

Columna	Tipo	Restricciones / relación
id	bigint UNSIGNED	PK autoincremental
empresa_id	bigint UNSIGNED (FK)	→ empresas, cascade
codigo	string	nullable
titulo	string	—
autor	string	nullable
editorial	string	nullable
isbn	string	nullable
categoría	string	nullable
anio	unsignedInteger	nullable
ejemplares	unsignedInteger	default 1
disponibles	unsignedInteger	default 1
ubicacion	string	nullable
estado	string	default activo
created_at / updated_at	timestamp	gestionados por Eloquent

Índices: `empresa_id`, `estado`

Tabla `prestamos` · migración `0001_01_01_000024_create_biblioteca_tables.php`

Columna	Tipo	Restricciones / relación
id	bigint UNSIGNED	PK autoincremental
empresa_id	bigint UNSIGNED (FK)	→ empresas, cascade
libro_id	bigint UNSIGNED (FK)	→ libros, cascade
estudiante_id	bigint UNSIGNED (FK)	nullable, → estudiantes, set null

Columna	Tipo	Restricciones / relación
docente_id	bigint UNSIGNED (FK)	nullable, → docentes, set null
fecha_prestamo	date	—
fecha_prevista	date	nullable
fecha_devolucion	date	nullable
estado	string	default prestado
observacion	string	nullable
created_at / updated_at	timestamp	gestionados por Eloquent

Índices: [empresa_id](#), [estado](#)

Rutas expuestas

Verbo	URI	Nombre de ruta	Acción	Middleware
GET	/panel/biblioteca	biblioteca.index	BibliotecaController::index()	—
GET	/panel/biblioteca/libros/nuevo	biblioteca.libros.create	BibliotecaController::createLibro()	—
POST	/panel/biblioteca/libros	biblioteca.libros.store	BibliotecaController::storeLibro()	—
GET	/panel/biblioteca/libros/{libro}/editar	biblioteca.libros.edit	BibliotecaController::editLibro()	—
PUT	/panel/biblioteca/libros/{libro}	biblioteca.libros.update	BibliotecaController::updateLibro()	—
DELETE	/panel/biblioteca/libros/{libro}	biblioteca.libros.destroy	BibliotecaController::destroyLibro()	—
GET	/panel/biblioteca/prestamos	biblioteca.prestamos	BibliotecaController::prestamos()	—
POST	/panel/biblioteca/prestamos	biblioteca.prestar	BibliotecaController::prestar()	—
PATCH	/panel/biblioteca/prestamos/{prestamo}/devolver	biblioteca.devolver	BibliotecaController::devolver()	—

Módulos del dominio financiero

6.16 Pagos y cobranzas

Gestión de los cargos económicos de cada estudiante, con soporte de pagos parciales.

Proceso interno

- 1 Origen de los cargos**
La mayoría se crean automáticamente desde Admisiones. También pueden registrarse manualmente.
- 2 Cálculo del saldo**
El método `saldo()` devuelve `monto - monto_pagado`. No se almacena: se calcula, evitando incoherencias.
- 3 Registro de abono**
Suma al `monto_pagado` y recalcula el estado: `parcial` si queda saldo, `pagado` si se cubrió el total.
- 4 Detección de vencidos**
El scope correspondiente filtra por estado pendiente o parcial con fecha de vencimiento anterior a hoy.
- 5 Anulación**
Cambia el estado a `anulado` sin borrar, preservando el histórico.

Ficheros que participan

Rutas de fichero

```
app\Http\Controllers\PagoController.php
app\Models\Pago.php
database/migrations/0001_01_01_000008_create_pagos_table.php
resources\views\pagos\form.blade.php
resources\views\pagos\index.blade.php
resources\views\pagos\show.blade.php
routes/web.php (grupo pagos)
```

Modelo Pago	Detalle
Fichero	app\Models\Pago.php
Tabla	pagos
Campos asignables	empresa_id, estudiante_id, numero_recibo, concepto, programa, periodo, monto, monto_pagado, estado, fecha_vencimiento, fecha_pago, metodo_pago, observaciones
Relaciones	empresa() — belongsTo(Empresa) estudiante() — belongsTo(Estudiante)
Scopes	scopeDeMiEmpresa, scopeEstado, scopeBuscar

Modelo Pago	Detalle
Constantes	CONCEPTOS, METODOS, ESTADOS
Métodos propios	saldo(), estaVencido(), recalcularEstado(), etiquetaConcepto(), etiquetaMetodo()

Tablas de datos

Tabla pagos · migración 0001_01_01_000008_create_pagos_table.php

Columna	Tipo	Restricciones / relación
id	bigint UNSIGNED	PK autoincremental
empresa_id	bigint UNSIGNED (FK)	→ empresas, cascade
estudiante_id	bigint UNSIGNED (FK)	→ estudiantes, cascade
numero_recibo	string	nullable
concepto	string	—
programa	string	nullable
periodo	string	nullable
monto	decimal,10,2	default 0
monto_pagado	decimal,10,2	default 0
estado	string	default pendiente
fecha_vencimiento	date	nullable
fecha_pago	date	nullable
metodo_pago	string	nullable
observaciones	text	nullable
created_at / updated_at	timestamp	gestionados por Eloquent

Índices: empresa_id, estado

Rutas expuestas

Verbo	URI	Nombre de ruta	Acción	Middleware
GET	/panel/pagos	pagos.index	PagoController::index()	—
GET	/panel/pagos/crear	pagos.create	PagoController::create()	—
POST	/panel/pagos	pagos.store	PagoController::store()	—
GET	/panel/pagos/{pago}	pagos.show	PagoController::show()	—
POST	/panel/pagos/{pago}/pagar	pagos.pagar	PagoController::registrarPago()	—
PATCH	/panel/pagos/{pago}/anular	pagos.anular	PagoController::anular()	—

Verbo	URI	Nombre de ruta	Acción	Middleware
DELETE	/panel/pagos/{pago}	pagos.destroy	PagoController::destroy()	—

Nota técnica

El saldo se calcula en PHP y no en SQL, lo que impide ordenar directamente por saldo en la base de datos. En los reportes de deudores se resuelve trayendo los pagos pendientes y agrupando en memoria con `groupBy()`. Para volúmenes muy grandes convendría añadir una columna calculada.

6.17 Caja

Control de caja chica por turno, con apertura, movimientos y cierre con arqueo.

Proceso interno

- 1 Apertura**
Crea la caja con un monto inicial y la asocia al usuario que la abre.
- 2 Movimientos**
Cada ingreso o egreso se registra en `movimientos_caja` con su método de pago.
- 3 Monto esperado**
Se calcula como monto inicial más ingresos menos egresos.
- 4 Cierre con arqueo**
El usuario declara el monto contado; el sistema guarda la diferencia entre lo esperado y lo real.

Ficheros que participan

Rutas de fichero

```
app\Http\Controllers\CajaController.php
app\Models\Caja.php
app\Models\MovimientoCaja.php
database\migrations\0001_01_01_000023_create_cajas_table.php
database\migrations\0001_01_01_000023_create_cajas_table.php
resources\views\caja\index.blade.php
resources\views\caja\show.blade.php
routes\web.php (grupo caja)
```

Modelo Caja	Detalle
Fichero	app\Models\Caja.php
Tabla	cajas
Campos asignables	empresa_id, user_id, codigo, fecha_apertura, fecha_cierre, monto_inicial, monto_final, monto_esperado, diferencia, estado, observacion
Relaciones	empresa() — belongsTo(Empresa) responsable() — belongsTo(User) movimientos() — hasMany(MovimientoCaja)
Scopes	scopeDeMiEmpresa
Métodos propios	totalIngresos(), totalEgresos(), esperado(), estaAbierta()

Modelo MovimientoCaja	Detalle
Fichero	app\Models\MovimientoCaja.php
Tabla	movimientos_caja

Modelo MovimientoCaja	Detalle
Campos asignables	caja_id, empresa_id, user_id, tipo, concepto, monto, metodo, referencia
Relaciones	caja() — belongsTo(Caja) usuario() — belongsTo(User)
Constantes	TIPOS, METODOS
Métodos propios	etiquetaMetodo()

Tablas de datos

Tabla `cajas` · migración `0001_01_01_000023_create_cajas_table.php`

Columna	Tipo	Restricciones / relación
id	bigint UNSIGNED	PK autoincremental
empresa_id	bigint UNSIGNED (FK)	→ empresas, cascade
user_id	bigint UNSIGNED (FK)	nullable, → users, set null
codigo	string	nullable
fecha_apertura	dateTime	—
fecha_cierre	dateTime	nullable
monto_inicial	decimal,10,2	default 0
monto_final	decimal,10,2	nullable
monto_esperado	decimal,10,2	nullable
diferencia	decimal,10,2	nullable
estado	string	default abierta
observacion	string	nullable
created_at / updated_at	timestamp	gestionados por Eloquent

Índices: `empresa_id`, `estado`

Tabla `movimientos_caja` · migración `0001_01_01_000023_create_cajas_table.php`

Columna	Tipo	Restricciones / relación
id	bigint UNSIGNED	PK autoincremental
caja_id	bigint UNSIGNED (FK)	→ cajas, cascade
empresa_id	bigint UNSIGNED (FK)	→ empresas, cascade
user_id	bigint UNSIGNED (FK)	nullable, → users, set null
tipo	string	default ingreso
concepto	string	—
monto	decimal,10,2	default 0

Columna	Tipo	Restricciones / relación
metodo	string	default efectivo
referencia	string	nullable
created_at / updated_at	timestamp	gestionados por Eloquent

Índices: `caja_id`, `tipo`

Rutas expuestas

Verbo	URI	Nombre de ruta	Acción	Middleware
GET	<code>/panel/caja</code>	<code>caja.index</code>	<code>CajaController::index()</code>	—
POST	<code>/panel/caja/abrir</code>	<code>caja.abrir</code>	<code>CajaController::abrir()</code>	—
GET	<code>/panel/caja/{caja}</code>	<code>caja.show</code>	<code>CajaController::show()</code>	—
POST	<code>/panel/caja/{caja}/movimiento</code>	<code>caja.movimiento</code>	<code>CajaController::movimiento()</code>	—
POST	<code>/panel/caja/{caja}/cerrar</code>	<code>caja.cerrar</code>	<code>CajaController::cerrar()</code>	—

Módulos del dominio administrativo

6.18 CRM / Prospectos

Embudo de captación con seguimiento y conversión a estudiante.

Proceso interno

- 1 Registro**
Datos de contacto, programa de interés y canal de origen.
- 2 Pipeline**
Estados nuevo, contactado, interesado, convertido y perdido, presentados como columnas en la vista.
- 3 Seguimiento**
El campo `proximo_seguimiento` alimenta la lista de tareas del día.
- 4 Conversión**
Al convertir, se crea el estudiante y se guarda su identificador en `estudiante_id`, enlazando el origen comercial con la ficha académica.

Ficheros que participan

Rutas de fichero

```
app\Http\Controllers\ProspectoController.php
app\Models\Prospecto.php
database\Migrations\0001_01_01_000020_create_prospectos_table.php
resources\views\crm\form.blade.php
resources\views\crm\index.blade.php
resources\views\crm\show.blade.php
routes\web.php (grupo crm)
```

Modelo Prospecto	Detalle
Fichero	app\Models\Prospecto.php
Tabla	prospectos
Campos asignables	empresa_id, estudiante_id, nombres, apellidos, telefono, email, programa_interes, origen, estado, notas, fecha_contacto, proximo_seguimiento
Relaciones	empresa() — belongsTo(Empresa) estudiante() — belongsTo(Estudiante)
Scopes	scopeDeMiEmpresa, scopeBuscar
Constantes	ESTADOS, ORIGENES
Métodos propios	getNombreCompletoAttribute(), iniciales(), seguimientoVencido()

Tablas de datos

Tabla `prospectos` · migración `0001_01_01_000020_create_prospectos_table.php`

Columna	Tipo	Restricciones / relación
id	bigint UNSIGNED	PK autoincremental
empresa_id	bigint UNSIGNED (FK)	→ empresas, cascade
estudiante_id	bigint UNSIGNED (FK)	nullable, → estudiantes, set null
nombres	string	—
apellidos	string	nullable
telefono	string	nullable
email	string	nullable
programa_interes	string	nullable
origen	string	default web
estado	string	default nuevo
notas	text	nullable
fecha_contacto	date	nullable
proximo_seguimiento	date	nullable
created_at / updated_at	timestamp	gestionados por Eloquent

Índices: `empresa_id`, `estado`

Rutas expuestas

Verbo	URI	Nombre de ruta	Acción	Middleware
GET	<code>/panel/crm</code>	<code>crm.index</code>	<code>ProspectoController::index()</code>	—
GET	<code>/panel/crm/nuevo</code>	<code>crm.create</code>	<code>ProspectoController::create()</code>	—
POST	<code>/panel/crm</code>	<code>crm.store</code>	<code>ProspectoController::store()</code>	—
GET	<code>/panel/crm/{prospecto}</code>	<code>crm.show</code>	<code>ProspectoController::show()</code>	—
GET	<code>/panel/crm/{prospecto}/editar</code>	<code>crm.edit</code>	<code>ProspectoController::edit()</code>	—
PUT	<code>/panel/crm/{prospecto}</code>	<code>crm.update</code>	<code>ProspectoController::update()</code>	—
PATCH	<code>/panel/crm/{prospecto}/estado</code>	<code>crm.estado</code>	<code>ProspectoController::cambiarEstado()</code>	—
POST	<code>/panel/crm/{prospecto}/convertir</code>	<code>crm.convertir</code>	<code>ProspectoController::convertir()</code>	—
DELETE	<code>/panel/crm/{prospecto}</code>	<code>crm.destroy</code>	<code>ProspectoController::destroy()</code>	—

6.19 Comunicaciones

Emisión de anuncios y comunicados segmentados por audiencia.

Proceso interno

- 1 Composición**
Título, mensaje, canal y destinatario.
- 2 Segmentación**
todos, estudiantes, docentes o un programa específico.
- 3 Cálculo de alcance**
Antes de enviar, cuenta los destinatarios y lo guarda en `total_destinatarios`.
- 4 Registro de autoría**
El campo `user_id` guarda quién lo emitió.

Ficheros que participan

Rutas de fichero

```
app\Http\Controllers\ComunicadoController.php
app\Models\Comunicado.php
database\Migrations\0001_01_01_000021_create_comunicados_table.php
resources\views\comunicaciones\form.blade.php
resources\views\comunicaciones\index.blade.php
resources\views\comunicaciones\show.blade.php
routes\web.php (grupo comunicaciones)
```

Modelo Comunicado	Detalle
Fichero	app\Models\Comunicado.php
Tabla	comunicados
Campos asignables	empresa_id, user_id, titulo, mensaje, canal, destinatario, programa, total_destinatarios, estado, fecha_envio
Relaciones	empresa() — belongsTo(Empresa) autor() — belongsTo(User)
Scopes	scopeDeMiEmpresa, scopeBuscar
Constantes	CANALES, DESTINARIOS
Métodos propios	etiquetaCanal(), etiquetaDestinatario(), destinatarios()

Tablas de datos

Tabla `comunicados` · migración `0001_01_01_000021_create_comunicados_table.php`

Columna	Tipo	Restricciones / relación
id	bigint UNSIGNED	PK autoincremental
empresa_id	bigint UNSIGNED (FK)	→ empresas, cascade
user_id	bigint UNSIGNED (FK)	nullable, → users, set null
titulo	string	—
mensaje	text	—
canal	string	default anuncio
destinatario	string	default todos
programa	string	nullable
total_destinatarios	unsignedInteger	default 0
estado	string	default enviado
fecha_envio	dateTime	nullable
created_at / updated_at	timestamp	gestionados por Eloquent

Índices: [empresa_id](#), [estado](#)

Rutas expuestas

Verbo	URI	Nombre de ruta	Acción	Middleware
GET	/panel/comunicaciones	comunicaciones.index	ComunicadoController::index()	—
GET	/panel/comunicaciones/nuevo	comunicaciones.create	ComunicadoController::create()	—
POST	/panel/comunicaciones	comunicaciones.store	ComunicadoController::store()	—
GET	/panel/comunicaciones/{comunicado}	comunicaciones.show	ComunicadoController::show()	—
DELETE	/panel/comunicaciones/{comunicado}	comunicaciones.destroy	ComunicadoController::destroy()	—

Nota técnica

Los canales email, SMS y WhatsApp quedan **registrados pero no enviados**: el sistema documenta la comunicación, no la transmite. Integrar un proveedor real requeriría añadir un servicio y encolar el envío con las tablas [jobs](#) que ya existen en el esquema.

6.20 Recursos Humanos

Gestión del personal completo de la institución, no sólo docente.

Proceso interno

- 1 Áreas**
dirección, académica, administración, soporte, limpieza y otro.
- 2 Contratos**
planilla, honorarios, part time y prácticas.
- 3 Masa salarial**
Se calcula sumando el sueldo del personal activo.
- 4 Antigüedad**
El método `antigüedad()` la deriva de la fecha de ingreso.

Ficheros que participan

Rutas de fichero

```
app\Http\Controllers\RrhhController.php
app\Models\Empleado.php
database\Migrations\0001_01_01_000025_create_empleados_table.php
resources\views\rrhh\form.blade.php
resources\views\rrhh\index.blade.php
resources\views\rrhh\show.blade.php
routes\web.php (grupo rrhh)
```

Modelo Empleado	Detalle
Fichero	app\Models\Empleado.php
Tabla	empleados
Campos asignables	empresa_id, codigo, nombres, apellidos, tipo_documento, documento, email, telefono, cargo, area, tipo_contrato, fecha_ingreso, fecha_cese, sueldo, estado, observaciones
Relaciones	empresa() — belongsTo(Empresa)
Scopes	scopeDeMiEmpresa, scopeBuscar
Constantes	AREAS, CONTRATOS, ESTADOS
Métodos propios	getNombreCompletoAttribute(), iniciales(), etiquetaArea(), etiquetaContrato(), antigüedad()

Tablas de datos

Tabla `empleados` · migración `0001_01_01_000025_create_empleados_table.php`

Columna	Tipo	Restricciones / relación
id	bigint UNSIGNED	PK autoincremental
empresa_id	bigint UNSIGNED (FK)	→ empresas, cascade
codigo	string	nullable
nombres	string	—
apellidos	string	—
tipo_documento	string	default DNI
documento	string	nullable
email	string	nullable
telefono	string	nullable
cargo	string	nullable
area	string	default administracion
tipo_contrato	string	default planilla
fecha_ingreso	date	nullable
fecha_cese	date	nullable
sueldo	decimal,10,2	default 0
estado	string	default activo
observaciones	text	nullable
created_at / updated_at	timestamp	gestionados por Eloquent

Índices: [empresa_id](#), [estado](#)

Rutas expuestas

Verbo	URI	Nombre de ruta	Acción	Middleware
GET	/panel/rrhh	rrhh.index	RrhhController::index()	—
GET	/panel/rrhh/nuevo	rrhh.create	RrhhController::create()	—
POST	/panel/rrhh	rrhh.store	RrhhController::store()	—
GET	/panel/rrhh/{empleado}	rrhh.show	RrhhController::show()	—
GET	/panel/rrhh/{empleado}/editar	rrhh.edit	RrhhController::edit()	—
PUT	/panel/rrhh/{empleado}	rrhh.update	RrhhController::update()	—
DELETE	/panel/rrhh/{empleado}	rrhh.destroy	RrhhController::destroy()	—

6.21 Inventario

Control de activos, materiales y consumibles con kardex de movimientos.

Proceso interno

- 1 Catálogo**
Cada artículo tiene tipo, unidad, stock, stock mínimo, ubicación y costo.
- 2 Movimientos**
entrada, salida y ajuste. Cada uno actualiza el stock del artículo.
- 3 Alerta de stock**
Se comparan `stock` y `stock_minimo` para señalar los artículos por reponer.
- 4 Valorización**
El costo unitario permite calcular el valor total del inventario.

Ficheros que participan

Rutas de fichero

```
app\Http\Controllers\InventarioController.php
app\Models\Articulo.php
app\Models\MovimientoInventario.php
database\migrations\0001_01_01_000026_create_inventario_tables.php
database\migrations\0001_01_01_000026_create_inventario_tables.php
resources\views\inventario\form.blade.php
resources\views\inventario\index.blade.php
resources\views\inventario\show.blade.php
routes\web.php (grupo inventario)
```

Modelo Artículo	Detalle
Fichero	app\Models\Articulo.php
Tabla	articulos
Campos asignables	empresa_id, codigo, nombre, categoria, tipo, unidad, stock, stock_minimo, ubicacion, costo, estado
Relaciones	empresa() — belongsTo(Empresa) movimientos() — hasMany(MovimientoInventario)
Scopes	scopeDeMiEmpresa, scopeBuscar
Constantes	TIPOS, UNIDADES
Métodos propios	stockBajo(), valorInventario(), etiquetaTipo()

Modelo	Detalle
MovimientoInventario	
Fichero	app\Models\MovimientoInventario.php
Tabla	movimientos_inventario
Campos asignables	articulo_id, empresa_id, user_id, tipo, cantidad, stock_resultante, motivo, referencia
Relaciones	articulo() — belongsTo(Articulo) usuario() — belongsTo(User)
Constantes	TIPOS

Tablas de datos

Tabla `articulos` · migración `0001_01_01_000026_create_inventario_tables.php`

Columna	Tipo	Restricciones / relación
id	bigint UNSIGNED	PK autoincremental
empresa_id	bigint UNSIGNED (FK)	→ empresas, cascade
codigo	string	nullable
nombre	string	—
categoria	string	nullable
tipo	string	default material
unidad	string	default unidad
stock	integer	default 0
stock_minimo	unsignedInteger	default 0
ubicacion	string	nullable
costo	decimal,10,2	default 0
estado	string	default activo
created_at / updated_at	timestamp	gestionados por Eloquent

Índices: `empresa_id`, `estado`

Tabla `movimientos_inventario` · migración `0001_01_01_000026_create_inventario_tables.php`

Columna	Tipo	Restricciones / relación
id	bigint UNSIGNED	PK autoincremental
articulo_id	bigint UNSIGNED (FK)	→ articulos, cascade
empresa_id	bigint UNSIGNED (FK)	→ empresas, cascade
user_id	bigint UNSIGNED (FK)	nullable, → users, set null
tipo	string	default entrada

Columna	Tipo	Restricciones / relación
cantidad	integer	default 0
stock_resultante	integer	default 0
motivo	string	nullable
referencia	string	nullable
created_at / updated_at	timestamp	gestionados por Eloquent

Índices: `articulo_id`, `tipo`

Rutas expuestas

Verbo	URI	Nombre de ruta	Acción	Middleware
GET	<code>/panel/inventario</code>	<code>inventario.index</code>	<code>InventarioController::index()</code>	—
GET	<code>/panel/inventario/nuevo</code>	<code>inventario.create</code>	<code>InventarioController::create()</code>	—
POST	<code>/panel/inventario</code>	<code>inventario.store</code>	<code>InventarioController::store()</code>	—
GET	<code>/panel/inventario/{articulo}</code>	<code>inventario.show</code>	<code>InventarioController::show()</code>	—
GET	<code>/panel/inventario/{articulo}/editar</code>	<code>inventario.edit</code>	<code>InventarioController::edit()</code>	—
PUT	<code>/panel/inventario/{articulo}</code>	<code>inventario.update</code>	<code>InventarioController::update()</code>	—
POST	<code>/panel/inventario/{articulo}/movimiento</code>	<code>inventario.movimiento</code>	<code>InventarioController::movimiento()</code>	—
DELETE	<code>/panel/inventario/{articulo}</code>	<code>inventario.destroy</code>	<code>InventarioController::destroy()</code>	—

6.22 Reportes

Centro de análisis con indicadores, gráficos, exportación a CSV y versión imprimible.

Proceso interno

1 Filtro temporal

Todo el módulo trabaja sobre un rango de fechas `desde–hasta`, con el año en curso por defecto.

2 Series mensuales

El método privado `serieMensual()` recorre los meses del rango aplicando una función de agregación recibida como closure. Se reutiliza para matrículas y recaudación.

3 Indicadores

Recaudado, por cobrar, vencido y matrículas del periodo.

4 Exportación CSV

Cinco conjuntos de datos. Usa `streamDownload()` con BOM `\xEF\xBB\xBF` y separador punto y coma para compatibilidad con Excel en español.

5 Versión imprimible

Plantilla propia con la identidad de la institución y reglas `@media print` que evitan cortar filas entre páginas.

Ficheros que participan

Rutas de fichero

```
app\Http\Controllers\ReporteController.php
resources\views\reportes\imprimir.blade.php
resources\views\reportes\index.blade.php
routes\web.php (grupo reportes)
```

Tablas de datos

Rutas expuestas

Verbo	URI	Nombre de ruta	Acción	Middleware
GET	/panel/reportes	reportes.index	ReporteController::index()	—
GET	/panel/reportes/exportar	reportes.exportar	ReporteController::exportar()	—
GET	/panel/reportes/imprimir	reportes.imprimir	ReporteController::imprimir()	—

Nota técnica

La exportación no genera XLSX real sino CSV. El BOM inicial es lo que hace que Excel interprete correctamente los acentos; sin él, aparecen caracteres corruptos. El separador punto y coma es el esperado por Excel en configuraciones regionales de español.

Módulos del dominio de sistema

6.23 Configuración

Ajustes internos de la institución: datos propios, usuarios y roles.

Proceso interno

- 1 Mi Institución**
Permite editar los datos de la propia empresa. El controlador restringe la consulta al identificador del usuario autenticado.
- 2 Usuarios y Roles**
Alta de personal con rol y activación o desactivación de cuentas.
- 3 Restricción de creación**
Los métodos `create()` y `store()` de empresas responden 403: sólo el Super Admin da de alta instituciones.

Ficheros que participan

Rutas de fichero

```
app\Http\Controllers\Configuracion\ConfiguracionController.php
app\Models\User.php
database\migrations\0001_01_01_000002_create_users_table.php
resources\views\configuracion\index.blade.php
routes\web.php (grupo configuracion)
```

Modelo User	Detalle
Fichero	app\Models\User.php
Tabla	(convención)
Campos asignables	empresa_id, name, email, password, rol, telefono, avatar, activo
Relaciones	empresa() — belongsTo(Empresa)
Métodos propios	esSuperadmin(), iniciales()

Tablas de datos

Tabla `users` · migración `0001_01_01_000002_create_users_table.php`

Columna	Tipo	Restricciones / relación
id	bigint UNSIGNED	PK autoincremental
empresa_id	bigint UNSIGNED (FK)	nullable, → empresas, set null
name	string	—

Columna	Tipo	Restricciones / relación
email	string	único
email_verified_at	timestamp	nullable
password	string	—
rol	string	default admin
telefono	string	nullable
avatar	string	nullable
activo	boolean	default true
created_at / updated_at	timestamp	gestionados por Eloquent

Rutas expuestas

Verbo	URI	Nombre de ruta	Acción	Middleware
GET	/panel/configuracion	configuracion.index	ConfiguracionController::index()	—
GET	/panel/configuracion/empresas/crear	configuracion.empresas.create	EmpresaController::create()	—
POST	/panel/configuracion/empresas	configuracion.empresas.store	EmpresaController::store()	—
GET	/panel/configuracion/empresas/{empresa}/editar	configuracion.empresas.edit	EmpresaController::edit()	—
PUT	/panel/configuracion/empresas/{empresa}	configuracion.empresas.update	EmpresaController::update()	—
PATCH	/panel/configuracion/empresas/{empresa}/toggle	configuracion.empresas.toggle	EmpresaController::toggle()	—
GET	/panel/configuracion/empresas/{empresa}/modulos	configuracion.empresas.modulos	EmpresaController::modulos()	—
PUT	/panel/configuracion/empresas/{empresa}/modulos	configuracion.empresas.modulos.update	EmpresaController::updateModulos()	—
POST	/panel/configuracion/usuarios	configuracion.usuarios.store	UsuarioController::store()	—
PATCH	/panel/configuracion/usuarios/{usuario}/toggle	configuracion.usuarios.toggle	UsuarioController::toggle()	—

Nota técnica

Este módulo tuvo una fuga de datos entre instituciones en versiones tempranas: el listado de empresas no filtraba por el identificador del usuario. La corrección fue acotar la consulta con `where('id', auth()->user()->empresa_id)`.

6.24 Mantenimiento del tenant

Versión acotada del mantenimiento, disponible para el administrador de cada institución.

Proceso interno

1 Autorización

Triple verificación: que no sea superadmin, que el rol sea `admin` y que tenga institución asignada.

2 Directorio aislado

Las copias se guardan en `storage\app\backups\empresa_{id}\`, de modo que una institución nunca ve los respaldos de otra.

3 Copia propia

Vuelca 21 tablas operativas filtrando por `empresa_id`. No incluye la estructura de la base.

4 Restauración verificada

Además de la firma del sistema, comprueba que la cabecera contenga `-- empresa_id: {id}` coincidente. Un archivo de otra institución es rechazado.

5 Reseteo por grupos

Cuatro grupos seleccionables; conserva empresa, usuarios, plan y módulos.

Ficheros que participan

Rutas de fichero

```
app\Http\Controllers\MantenimientoController.php
resources\views\mantenimiento\index.blade.php
routes\web.php (grupo mantenimiento)
```

Tablas de datos

Rutas expuestas

Verbo	URI	Nombre de ruta	Acción	Middleware
GET	/panel/mantenimiento	mantenimiento.index	MantenimientoController::index()	—
POST	/panel/mantenimiento/backup	mantenimiento.backup	MantenimientoController::backup()	—
GET	/panel/mantenimiento/backup/{archivo}/descargar	mantenimiento.descargar	MantenimientoController::descargar()	—
DELETE	/panel/mantenimiento/backup/{archivo}	mantenimiento.eliminar	MantenimientoController::eliminar()	—
POST	/panel/mantenimiento/restaurar	mantenimiento.restaurar	MantenimientoController::restaurar()	—
POST	/panel/mantenimiento/reset	mantenimiento.reset	MantenimientoController::reset()	—

Nota técnica

Existen **dos clases con el mismo nombre**: `Admin\MantenimientoController` y `MantenimientoController`. En `routes\web.php` la primera se importa con alias `AdminMantenimientoController` para evitar la colisión de nombres de PHP.

7. Procesos transversales

7.1 Construcción del dashboard

El dashboard del tenant calcula ocho indicadores y cuatro series para gráficos en una sola pasada del controlador.

1 Redirección por rol

Si el usuario es superadmin, se le envía a `admin.dashboard`.

2 Cálculo de indicadores

Ocho consultas de agregación acotadas por institución: estudiantes, docentes, programas, ingresos del mes, aulas, clases de hoy, por cobrar y pagos pendientes.

3 Series para gráficos

Cuatro conjuntos: matrículas por mes, estudiantes por programa, recaudación mensual y distribución por estado.

4 Datos de demostración

Si un conjunto viene vacío, se sustituye por valores de ejemplo para que el gráfico no aparezca en blanco en una institución recién creada.

5 Renderizado

Las series se serializan a JSON y se inyectan en la configuración de Chart.js. Cada lienzo vive en un contenedor de altura fija con `maintainAspectRatio: false` para que el diseño sea adaptable.

7.2 Generación de copias de seguridad

Algoritmo compartido por el mantenimiento de plataforma y el del tenant:

Pseudocódigo del volcado

```
cabecera() → firma, institución y fecha
SET FOREIGN_KEY_CHECKS = 0;

para cada tabla del alcance:
  si es copia completa:
    DROP TABLE IF EXISTS `tabla`;
    SHOW CREATE TABLE `tabla` → aplanado a una línea
  si es copia por institución:
    DELETE FROM `tabla` WHERE empresa_id = X;

    SELECT * por lotes de 200 filas
    cada valor escapado con $pdo->quote()
    INSERT INTO `tabla` (cols) VALUES (...), (...);
```

```
SET FOREIGN_KEY_CHECKS = 1;
```

Casos especiales del volcado

La tabla `empresas` se filtra por `id` y no por `empresa_id`. La tabla `calificaciones` se filtra por subconsulta sobre `evaluaciones`. Las tablas `sessions`, `cache` y `cache_locks` se excluyen siempre por ser transitorias.

7.3 Compilación y caché de vistas

Blade compila cada plantilla a PHP plano en `storage\framework\views\`. Hay una particularidad del motor que conviene conocer.

La regla del carácter previo a una directiva

La expresión regular de Blade usa `\B@`, lo que significa que una directiva pegada a un carácter de palabra **no se compila**. Escribir `...con esos filtros@endif` deja el `@endif` como texto literal y produce un error de sintaxis en el archivo compilado. Esta protección existe para no romper las direcciones de correo escritas en las plantillas. **Deja siempre un espacio antes de cualquier directiva.**

8. Instalación y puesta en marcha

1 Requisitos previos

PHP 8.2 o superior con las extensiones pdo_mysql, mbstring, openssl, fileinfo y ctype. MySQL 8 o MariaDB 10.4+. Composer 2.

2 Instalar dependencias

```
composer install
```

3 Configurar el entorno

Copiar `.env.example` a `.env` y ajustar la conexión: base `suite_saas_educacion_integral`, usuario `root`, puerto `3306`.

4 Generar la clave

```
php artisan key:generate
```

5 Crear la base de datos

```
CREATE DATABASE suite_saas_educacion_integral CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
```

6 Ejecutar las migraciones

```
php artisan migrate
```

7 Cargar los catálogos y datos demo

```
php artisan db:seed
```

8 Levantar el servidor

```
php artisan serve --port=8062
```

Comandos de mantenimiento habituales

Comando	Cuándo usarlo
<code>php artisan view:clear</code>	Después de editar vistas si los cambios no se reflejan.
<code>php artisan config:clear</code>	Después de modificar <code>.env</code> o los archivos de configuración.
<code>php artisan route:list</code>	Para inspeccionar las 163 rutas registradas y sus middleware.
<code>php artisan migrate:fresh --seed</code>	Reconstruir la base desde cero. Destruye todos los datos.
<code>php artisan db:seed --class=ModuloSeeder</code>	Recargar sólo el catálogo de módulos.
<code>php artisan optimize</code>	Cachear configuración y rutas para producción.

Seeders disponibles

Seeder	Qué carga
TipoNegocioSeeder	Los 9 tipos de negocio educativo.
PlanSeeder	Los 3 planes comerciales con sus módulos.
ModuloSeeder	El catálogo de 22 módulos con grupo, icono y orden.
EmpresaDemoSeeder	Instituciones de demostración con su usuario administrador.
EstudianteSeeder / DocenteSeeder / ProgramaSeeder	Datos académicos de ejemplo.
AulaSeeder / HorarioSeeder / PagoSeeder	Ambientes, programación y cobranza de ejemplo.
DatabaseSeeder	Orquestador: invoca a todos los anteriores en el orden correcto.

Orden de ejecución de los seeders

El orden importa por las claves foráneas: primero tipos de negocio y planes, después módulos, luego empresas, y sólo entonces los datos académicos. [DatabaseSeeder](#) ya respeta esta secuencia.

9. Cómo agregar un módulo nuevo

El sistema sigue una receta repetible. Estos son los ocho pasos, en orden, para incorporar un módulo manteniendo la coherencia arquitectónica.

1 Migración

Crear la tabla en `database\migrations\`. **Obligatorio:** incluir `empresa_id` con `constrained('empresas')->cascadeOnDelete()` y un índice sobre `[empresa_id, estado]`.

2 Modelo

Crear la clase en `app\Models\` con `$table`, `$fillable`, `$casts`, las constantes de estado, las relaciones y el scope `scopeDeMiEmpresa()`.

3 Controlador

Crear en `app\Http\Controllers\` con los métodos CRUD y el método protegido `autorizar()` que valide la pertenencia institucional.

4 Rutas

Añadir un grupo en `routes\web.php` con `prefix()`, `name()` y `middleware('modulo:slug')`. Debe quedar **antes** de la ruta genérica `modulo.show`, que actúa como comodín.

5 Vistas

Crear la carpeta en `resources\views\` con al menos `index.blade.php`, `form.blade.php` y `show.blade.php`, extendiendo `layouts\app.blade.php`.

6 Catálogo de módulos

Añadir la entrada en `ModuloSeeder.php` con nombre, slug, icono, grupo y orden. Si la base ya existe, insertar también con SQL en `modulos` y `empresa_modulo`.

7 Menú lateral

Dos ediciones en `partials\sidebar.blade.php`: una línea en el `match()` que resuelve la URL, y otra en la expresión que determina el estado activo.

8 Icono

Si usas un icono nuevo, añadir su trazado SVG al diccionario de `partials\icon.blade.php`. Sin esa entrada, se dibuja el icono `cube` por defecto.

Los tres olvidos más frecuentes

1. Definir la ruta después de `modulo.show`: el comodín la captura y nunca se alcanza el controlador. **2.** Olvidar la segunda edición del sidebar: el elemento no se marca como activo al navegarlo. **3.** Sembrar el módulo sólo en el seeder cuando la base ya está creada: el módulo no aparece hasta ejecutar el SQL de alta.

10. Anexos

A. Índice completo de tablas

Tabla	Columnas	Migración que la crea
articulos	13	0001_01_01_000026_create_inventario_tables.php
asistencias	8	0001_01_01_000015_create_asistencias_table.php
aulas	10	0001_01_01_000009_create_aulas_table.php
cache	3	0001_01_01_000004_create_cache_table.php
cache_locks	3	0001_01_01_000004_create_cache_table.php
cajas	13	0001_01_01_000023_create_cajas_table.php
calificaciones	6	0001_01_01_000017_create_calificaciones_table.php
certificados	11	0001_01_01_000019_create_certificados_table.php
comunicados	12	0001_01_01_000021_create_comunicados_table.php
docentes	17	0001_01_01_000007_create_docentes_table.php
empleados	18	0001_01_01_000025_create_empleados_table.php
empresa_modulo	5	0001_01_01_000003_create_modulos_table.php
empresas	13	0001_01_01_000001_create_empresas_table.php
estudiantes	21	0001_01_01_000005_create_estudiantes_table.php
evaluaciones	12	0001_01_01_000016_create_evaluaciones_table.php
horarios	12	0001_01_01_000010_create_horarios_table.php
libros	14	0001_01_01_000024_create_biblioteca_tables.php
matriculas	13	0001_01_01_000018_create_matriculas_table.php
modulos	9	0001_01_01_000003_create_modulos_table.php
movimientos_caja	10	0001_01_01_000023_create_cajas_table.php
movimientos_inventario	10	0001_01_01_000026_create_inventario_tables.php
pagos	15	0001_01_01_000008_create_pagos_table.php
password_reset_tokens	3	0001_01_01_000002_create_users_table.php
planes	13	0001_01_01_000011_create_planes_table.php
prestamos	11	0001_01_01_000024_create_biblioteca_tables.php
programas	14	0001_01_01_000006_create_programas_table.php
prospectos	14	0001_01_01_000020_create_prospectos_table.php

Tabla	Columnas	Migración que la crea
recursos	13	0001_01_01_000022_create_recursos_table.php
sessions	6	0001_01_01_000002_create_users_table.php
suscripciones	9	0001_01_01_000013_create_suscripciones_table.php
tipos_negocio	7	0001_01_01_000000_create_tipos_negocio_table.php
users	11	0001_01_01_000002_create_users_table.php

B. Índice de modelos

Modelo	Tabla	Campos	Relaciones	Scopes
Articulo	articulos	11	2	DeMiEmpresa, Buscar
Asistencia	asistencias	6	3	DeMiEmpresa
Aula	aulas	8	2	DeMiEmpresa, Buscar
Caja	cajas	11	3	DeMiEmpresa
Calificacion	calificaciones	4	2	—
Certificado	certificados	9	2	DeMiEmpresa, Buscar
Comunicado	comunicados	10	2	DeMiEmpresa, Buscar
Docente	docentes	15	1	DeMiEmpresa, Buscar, Estado
Empleado	empleados	16	1	DeMiEmpresa, Buscar
Empresa	—	16	5	—
Estudiante	estudiantes	19	1	DeMiEmpresa, Buscar, Estado
Evaluacion	evaluaciones	10	4	DeMiEmpresa, Buscar
Horario	horarios	10	4	DeMiEmpresa
Libro	libros	12	2	DeMiEmpresa, Buscar
Matricula	matriculas	11	3	DeMiEmpresa, Buscar
Modulo	—	7	1	—
MovimientoCaja	movimientos_caja	8	2	—
MovimientoInventario	movimientos_inventario	8	2	—
Pago	pagos	13	2	DeMiEmpresa, Estado, Buscar
Plan	planes	11	2	—
Prestamo	prestamos	9	4	DeMiEmpresa
Programa	programas	12	1	DeMiEmpresa, Buscar, Tipo
Prospecto	prospectos	12	2	DeMiEmpresa, Buscar
Recurso	recursos	11	3	DeMiEmpresa, Buscar
Suscripcion	suscripciones	7	2	Vigente
TipoNegocio	tipos_negocio	5	1	—
User	—	8	1	—

C. Índice de controladores

Controlador	Métodos	Modelos que usa	Líneas
Admin\AdminDashboardController	1	Empresa, Plan, Suscripcion, TipoNegocio, User	59
Admin\EmpresaController	13	Empresa, Modulo, Plan, Suscripcion, TipoNegocio, User	291
Admin\MantenimientoController	6	Empresa	351
Admin\PlanController	6	Modulo, Plan	78
Admin\SuscripcionController	2	Suscripcion	40
AdmisionController	8	Estudiante, Matricula, Pago, Programa	242
AsistenciaController	3	Asistencia, Estudiante, Horario	92
AulaController	6	Aula	77
BibliotecaController	9	Docente, Estudiante, Libro, Prestamo	188
CajaController	5	Caja, MovimientoCaja	114
CampusController	7	Docente, Programa, Recurso	122
CertificadoController	6	Certificado, Estudiante	93
ComunicadoController	5	Comunicado, Docente, Estudiante, Programa	96
Configuracion\ConfiguracionController	1	Empresa, Modulo, TipoNegocio, User	32
Configuracion\EmpresaController	8	Empresa, Modulo, TipoNegocio	95
Configuracion\UsuarioController	3	Empresa, User	49
Controller	0	—	9
DashboardController	2	Aula, Docente, Estudiante, Horario, Pago, Programa	123
DocenteController	7	Docente	96
EstudianteController	7	Estudiante	99
EvaluacionController	9	Calificacion, Docente, Estudiante, Evaluacion, Programa	133
HorarioController	6	Aula, Docente, Horario, Programa	99
InventarioController	8	Articulo, MovimientoInventario	155
LoginController	3	—	70
MantenimientoController	6	—	414
PagoController	7	Estudiante, Pago	131
ProgramaController	7	Programa	91
ProspectoController	9	Estudiante, Programa, Prospecto	148
ReporteController	3	Estudiante, Evaluacion, Matricula, Pago	291

Controlador	Métodos	Modelos que usa	Líneas
RrhhController	7	Empleado	104

D. Índice de vistas por carpeta

Carpeta	N.º	Archivos
(raíz)	1	dashboard.blade.php
admin	1	dashboard.blade.php
admin.empresas	6	form.blade.php, index.blade.php, modulos.blade.php, show.blade.php, suscripcion.blade.php, ticket.blade.php
admin.layouts	1	app.blade.php
admin.mantenimiento	1	index.blade.php
admin.partials	1	sidebar.blade.php
admin.planes	2	form.blade.php, index.blade.php
admin.suscripciones	1	index.blade.php
admisiones	4	create.blade.php, edit.blade.php, index.blade.php, show.blade.php
asistencia	2	index.blade.php, tomar.blade.php
aulas	2	form.blade.php, index.blade.php
auth	1	login.blade.php
biblioteca	3	index.blade.php, libro.blade.php, prestamos.blade.php
caja	2	index.blade.php, show.blade.php
campus	3	form.blade.php, index.blade.php, show.blade.php
certificados	3	create.blade.php, index.blade.php, show.blade.php
comunicaciones	3	form.blade.php, index.blade.php, show.blade.php
configuracion	1	index.blade.php
configuracion.empresas	3	form.blade.php, index.blade.php, modulos.blade.php
configuracion.usuarios	1	index.blade.php
crm	3	form.blade.php, index.blade.php, show.blade.php
docentes	3	form.blade.php, index.blade.php, show.blade.php
estudiantes	3	form.blade.php, index.blade.php, show.blade.php
evaluaciones	4	calificar.blade.php, form.blade.php, index.blade.php, show.blade.php
horarios	2	form.blade.php, index.blade.php
inventario	3	form.blade.php, index.blade.php, show.blade.php
layouts	1	app.blade.php
mantenimiento	1	index.blade.php

Carpeta	N.º	Archivos
modules	1	placeholder.blade.php
pagos	3	form.blade.php, index.blade.php, show.blade.php
partials	2	icon.blade.php, sidebar.blade.php
programas	3	form.blade.php, index.blade.php, show.blade.php
reportes	2	imprimir.blade.php, index.blade.php
rrhh	3	form.blade.php, index.blade.php, show.blade.php

E. Índice completo de rutas

El sistema expone **163 rutas nombradas**.

Verbo	URI	Nombre	Acción
GET	/login	login	LoginController::show()
POST	/login	login.attempt	LoginController::login()
POST	/logout	logout	LoginController::logout()
GET	/admin	admin.dashboard	AdminDashboardController::index()
GET	/admin/empresas/crear	admin.empresas.create	AdminEmpresaController::create()
POST	/admin/empresas	admin.empresas.store	AdminEmpresaController::store()
GET	/admin/empresas/{empresa}	admin.empresas.show	AdminEmpresaController::show()
GET	/admin/empresas/{empresa}/editar	admin.empresas.edit	AdminEmpresaController::edit()
PUT	/admin/empresas/{empresa}	admin.empresas.update	AdminEmpresaController::update()
PATCH	/admin/empresas/{empresa}/toggle	admin.empresas.toggle	AdminEmpresaController::toggle()
DELETE	/admin/empresas/{empresa}	admin.empresas.destroy	AdminEmpresaController::destroy()
GET	/admin/empresas/{empresa}/modulos	admin.empresas.modulos	AdminEmpresaController::modulos()
PUT	/admin/empresas/{empresa}/modulos	admin.empresas.modulos.update	AdminEmpresaController::updateModulos()
GET	/admin/empresas/{empresa}/suscripcion	admin.empresas.suscripcion	AdminEmpresaController::suscripcionForm()
POST	/admin/empresas/{empresa}/suscripcion	admin.empresas.suscripcion.store	AdminEmpresaController::suscripcionStore()
GET	/admin/empresas/{empresa}/suscripcion/{suscripcion}/ticket	admin.empresas.suscripcion.ticket	AdminEmpresaController::ticket()
GET	/admin/planes/crear	admin.planes.create	PlanController::create()
POST	/admin/planes	admin.planes.store	PlanController::store()
GET	/admin/planes/{plan}/editar	admin.planes.edit	PlanController::edit()
PUT	/admin/planes/{plan}	admin.planes.update	PlanController::update()
DELETE	/admin/planes/{plan}	admin.planes.destroy	PlanController::destroy()
PATCH	/admin/suscripciones/{suscripcion}/estado	admin.suscripciones.estado	SuscripcionController::cambiarEstado()
POST	/admin/mantenimiento/backup	admin.mantenimiento.backup	AdminMantenimientoController::backup()

Verbo	URI	Nombre	Acción
GET	/admin/mantenimiento/backup/{archivo}/descargar	admin.mantenimiento.descargar	AdminMantenimientoController::descargar()
DELETE	/admin/mantenimiento/backup/{archivo}	admin.mantenimiento.eliminar	AdminMantenimientoController::eliminar()
POST	/admin/mantenimiento/restaurar	admin.mantenimiento.restaurar	AdminMantenimientoController::restaurar()
POST	/admin/mantenimiento/reset	admin.mantenimiento.reset	AdminMantenimientoController::reset()
GET	/dashboard	dashboard	DashboardController::index()
GET	/panel/configuracion	configuracion.index	ConfiguracionController::index()
GET	/panel/configuracion/empresas/crear	configuracion.empresas.create	EmpresaController::create()
POST	/panel/configuracion/empresas	configuracion.empresas.store	EmpresaController::store()
GET	/panel/configuracion/empresas/{empresa}/editar	configuracion.empresas.edit	EmpresaController::edit()
PUT	/panel/configuracion/empresas/{empresa}	configuracion.empresas.update	EmpresaController::update()
PATCH	/panel/configuracion/empresas/{empresa}/toggle	configuracion.empresas.toggle	EmpresaController::toggle()
GET	/panel/configuracion/empresas/{empresa}/modulos	configuracion.empresas.modulos	EmpresaController::modulos()
PUT	/panel/configuracion/empresas/{empresa}/modulos	configuracion.empresas.modulos.update	EmpresaController::updateModulos()
POST	/panel/configuracion/usuarios	configuracion.usuarios.store	UsuarioController::store()
PATCH	/panel/configuracion/usuarios/{usuario}/toggle	configuracion.usuarios.toggle	UsuarioController::toggle()
GET	/panel/estudiantes	estudiantes.index	EstudianteController::index()
GET	/panel/estudiantes/crear	estudiantes.create	EstudianteController::create()
POST	/panel/estudiantes	estudiantes.store	EstudianteController::store()
GET	/panel/estudiantes/{estudiante}	estudiantes.show	EstudianteController::show()
GET	/panel/estudiantes/{estudiante}/editar	estudiantes.edit	EstudianteController::edit()
PUT	/panel/estudiantes/{estudiante}	estudiantes.update	EstudianteController::update()
DELETE	/panel/estudiantes/{estudiante}	estudiantes.destroy	EstudianteController::destroy()
GET	/panel/programas	programas.index	ProgramaController::index()

Verbo	URI	Nombre	Acción
GET	/panel/programas/crear	programas.create	ProgramaController::create()
POST	/panel/programas	programas.store	ProgramaController::store()
GET	/panel/programas/{programa}	programas.show	ProgramaController::show()
GET	/panel/programas/{programa}/editar	programas.edit	ProgramaController::edit()
PUT	/panel/programas/{programa}	programas.update	ProgramaController::update()
DELETE	/panel/programas/{programa}	programas.destroy	ProgramaController::destroy()
GET	/panel/docentes	docentes.index	DocenteController::index()
GET	/panel/docentes/crear	docentes.create	DocenteController::create()
POST	/panel/docentes	docentes.store	DocenteController::store()
GET	/panel/docentes/{docente}	docentes.show	DocenteController::show()
GET	/panel/docentes/{docente}/editar	docentes.edit	DocenteController::edit()
PUT	/panel/docentes/{docente}	docentes.update	DocenteController::update()
DELETE	/panel/docentes/{docente}	docentes.destroy	DocenteController::destroy()
GET	/panel/pagos	pagos.index	PagoController::index()
GET	/panel/pagos/crear	pagos.create	PagoController::create()
POST	/panel/pagos	pagos.store	PagoController::store()
GET	/panel/pagos/{pago}	pagos.show	PagoController::show()
POST	/panel/pagos/{pago}/pagar	pagos.pagar	PagoController::registrarPago()
PATCH	/panel/pagos/{pago}/anular	pagos.anular	PagoController::anular()
DELETE	/panel/pagos/{pago}	pagos.destroy	PagoController::destroy()
GET	/panel/aulas	aulas.index	AulaController::index()
GET	/panel/aulas/crear	aulas.create	AulaController::create()
POST	/panel/aulas	aulas.store	AulaController::store()
GET	/panel/aulas/{aula}/editar	aulas.edit	AulaController::edit()
PUT	/panel/aulas/{aula}	aulas.update	AulaController::update()
DELETE	/panel/aulas/{aula}	aulas.destroy	AulaController::destroy()
GET	/panel/horarios	horarios.index	HorarioController::index()
GET	/panel/horarios/crear	horarios.create	HorarioController::create()

Verbo	URI	Nombre	Acción
POST	/panel/horarios	horarios.store	HorarioController::store()
GET	/panel/horarios/{horario}/editar	horarios.edit	HorarioController::edit()
PUT	/panel/horarios/{horario}	horarios.update	HorarioController::update()
DELETE	/panel/horarios/{horario}	horarios.destroy	HorarioController::destroy()
GET	/panel/inventario	inventario.index	InventarioController::index()
GET	/panel/inventario/nuevo	inventario.create	InventarioController::create()
POST	/panel/inventario	inventario.store	InventarioController::store()
GET	/panel/inventario/{articulo}	inventario.show	InventarioController::show()
GET	/panel/inventario/{articulo}/editar	inventario.edit	InventarioController::edit()
PUT	/panel/inventario/{articulo}	inventario.update	InventarioController::update()
POST	/panel/inventario/{articulo}/movimiento	inventario.movimiento	InventarioController::movimiento()
DELETE	/panel/inventario/{articulo}	inventario.destroy	InventarioController::destroy()
GET	/panel/rrhh	rrhh.index	RrhhController::index()
GET	/panel/rrhh/nuevo	rrhh.create	RrhhController::create()
POST	/panel/rrhh	rrhh.store	RrhhController::store()
GET	/panel/rrhh/{empleado}	rrhh.show	RrhhController::show()
GET	/panel/rrhh/{empleado}/editar	rrhh.edit	RrhhController::edit()
PUT	/panel/rrhh/{empleado}	rrhh.update	RrhhController::update()
DELETE	/panel/rrhh/{empleado}	rrhh.destroy	RrhhController::destroy()
GET	/panel/biblioteca	biblioteca.index	BibliotecaController::index()
GET	/panel/biblioteca/libros/nuevo	biblioteca.libros.create	BibliotecaController::createLibro()
POST	/panel/biblioteca/libros	biblioteca.libros.store	BibliotecaController::storeLibro()
GET	/panel/biblioteca/libros/{libro}/editar	biblioteca.libros.edit	BibliotecaController::editLibro()
PUT	/panel/biblioteca/libros/{libro}	biblioteca.libros.update	BibliotecaController::updateLibro()
DELETE	/panel/biblioteca/libros/{libro}	biblioteca.libros.destroy	BibliotecaController::destroyLibro()
GET	/panel/biblioteca/prestamos	biblioteca.prestamos	BibliotecaController::prestamos()

Verbo	URI	Nombre	Acción
POST	/panel/biblioteca/prestamos	biblioteca.prestar	BibliotecaController::prestar()
PATCH	/panel/biblioteca/prestamos/{prestamo}/devolver	biblioteca.devolver	BibliotecaController::devolver()
GET	/panel/caja	caja.index	CajaController::index()
POST	/panel/caja/abrir	caja.abrir	CajaController::abrir()
GET	/panel/caja/{caja}	caja.show	CajaController::show()
POST	/panel/caja/{caja}/movimiento	caja.movimiento	CajaController::movimiento()
POST	/panel/caja/{caja}/cerrar	caja.cerrar	CajaController::cerrar()
GET	/panel/campus	campus.index	CampusController::index()
GET	/panel/campus/nuevo	campus.create	CampusController::create()
POST	/panel/campus	campus.store	CampusController::store()
GET	/panel/campus/{recurso}	campus.show	CampusController::show()
GET	/panel/campus/{recurso}/editar	campus.edit	CampusController::edit()
PUT	/panel/campus/{recurso}	campus.update	CampusController::update()
DELETE	/panel/campus/{recurso}	campus.destroy	CampusController::destroy()
GET	/panel/comunicaciones	comunicaciones.index	ComunicadoController::index()
GET	/panel/comunicaciones/nuevo	comunicaciones.create	ComunicadoController::create()
POST	/panel/comunicaciones	comunicaciones.store	ComunicadoController::store()
GET	/panel/comunicaciones/{comunicado}	comunicaciones.show	ComunicadoController::show()
DELETE	/panel/comunicaciones/{comunicado}	comunicaciones.destroy	ComunicadoController::destroy()
GET	/panel/crm	crm.index	ProspectoController::index()
GET	/panel/crm/nuevo	crm.create	ProspectoController::create()
POST	/panel/crm	crm.store	ProspectoController::store()
GET	/panel/crm/{prospecto}	crm.show	ProspectoController::show()
GET	/panel/crm/{prospecto}/editar	crm.edit	ProspectoController::edit()
PUT	/panel/crm/{prospecto}	crm.update	ProspectoController::update()
PATCH	/panel/crm/{prospecto}/estado	crm.estado	ProspectoController::cambiarEstado()
POST	/panel/crm/{prospecto}/convertir	crm.convertir	ProspectoController::convertir()

Verbo	URI	Nombre	Acción
DELETE	/panel/crm/{prospecto}	crm.destroy	ProspectoController::destroy()
GET	/panel/certificados	certificados.index	CertificadoController::index()
GET	/panel/certificados/emitir	certificados.create	CertificadoController::create()
POST	/panel/certificados	certificados.store	CertificadoController::store()
GET	/panel/certificados/{certificado}	certificados.show	CertificadoController::show()
PATCH	/panel/certificados/{certificado}/anular	certificados.anular	CertificadoController::anular()
DELETE	/panel/certificados/{certificado}	certificados.destroy	CertificadoController::destroy()
GET	/panel/reportes	reportes.index	ReporteController::index()
GET	/panel/reportes/exportar	reportes.exportar	ReporteController::exportar()
GET	/panel/reportes/imprimir	reportes.imprimir	ReporteController::imprimir()
GET	/panel/admisiones	admisiones.index	AdmisionController::index()
GET	/panel/admisiones/matricular	admisiones.create	AdmisionController::create()
POST	/panel/admisiones	admisiones.store	AdmisionController::store()
GET	/panel/admisiones/{matricula}	admisiones.show	AdmisionController::show()
GET	/panel/admisiones/{matricula}/editar	admisiones.edit	AdmisionController::edit()
PUT	/panel/admisiones/{matricula}	admisiones.update	AdmisionController::update()
PATCH	/panel/admisiones/{matricula}/anular	admisiones.anular	AdmisionController::anular()
PATCH	/panel/admisiones/{matricula}/reactivar	admisiones.reactivar	AdmisionController::reactivar()
GET	/panel/evaluaciones	evaluaciones.index	EvaluacionController::index()
GET	/panel/evaluaciones/crear	evaluaciones.create	EvaluacionController::create()
POST	/panel/evaluaciones	evaluaciones.store	EvaluacionController::store()
GET	/panel/evaluaciones/{evaluacion}	evaluaciones.show	EvaluacionController::show()
GET	/panel/evaluaciones/{evaluacion}/editar	evaluaciones.edit	EvaluacionController::edit()
PUT	/panel/evaluaciones/{evaluacion}	evaluaciones.update	EvaluacionController::update()
GET	/panel/evaluaciones/{evaluacion}/calificar	evaluaciones.calificar	EvaluacionController::calificar()

Verbo	URI	Nombre	Acción
POST	/panel/evaluaciones/{evaluacion}/calificar	evaluaciones.calificar.store	EvaluacionController::guardarCalificaciones()
DELETE	/panel/evaluaciones/{evaluacion}	evaluaciones.destroy	EvaluacionController::destroy()
GET	/panel/asistencia	asistencia.index	AsistenciaController::index()
GET	/panel/asistencia/clase/{horario}	asistencia.tomar	AsistenciaController::tomar()
POST	/panel/asistencia/clase/{horario}	asistencia.store	AsistenciaController::store()
GET	/panel/mantenimiento	mantenimiento.index	MantenimientoController::index()
POST	/panel/mantenimiento/backup	mantenimiento.backup	MantenimientoController::backup()
GET	/panel/mantenimiento/backup/{archivo}/descargar	mantenimiento.descargar	MantenimientoController::descargar()
DELETE	/panel/mantenimiento/backup/{archivo}	mantenimiento.eliminar	MantenimientoController::eliminar()
POST	/panel/mantenimiento/restaurar	mantenimiento.restaurar	MantenimientoController::restaurar()
POST	/panel/mantenimiento/reset	mantenimiento.reset	MantenimientoController::reset()

Sobre este documento

El contenido de los anexos y de las fichas de módulo fue **extraído automáticamente del código fuente** mediante un analizador que recorre las migraciones, los modelos, los controladores, las vistas y el fichero de rutas. Si modificas el sistema, vuelve a generar este manual para mantenerlo sincronizado.