Bar POS - SaaS Multi-Tenant

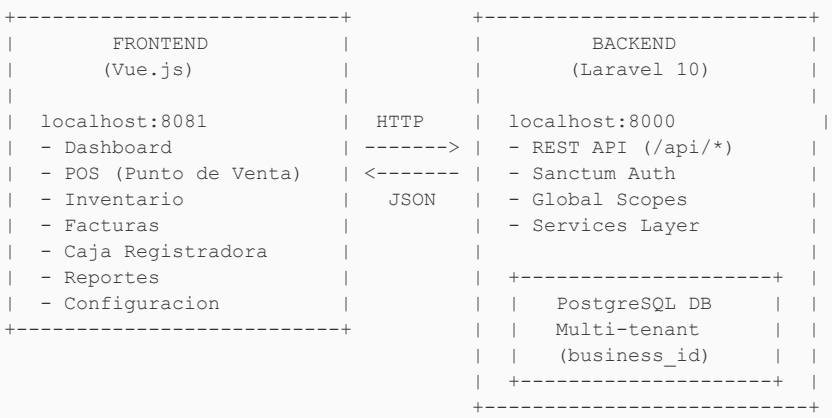
Documentacion Tecnica del Proyecto

Backend: Laravel 10 + PostgreSQL
Frontend: Vue.js
Arquitectura: REST API + Sanctum Auth

Version MVP v1.0
Febrero 2026

Tabla de Contenidos

1. Arquitectura General



Flujo de una Peticion

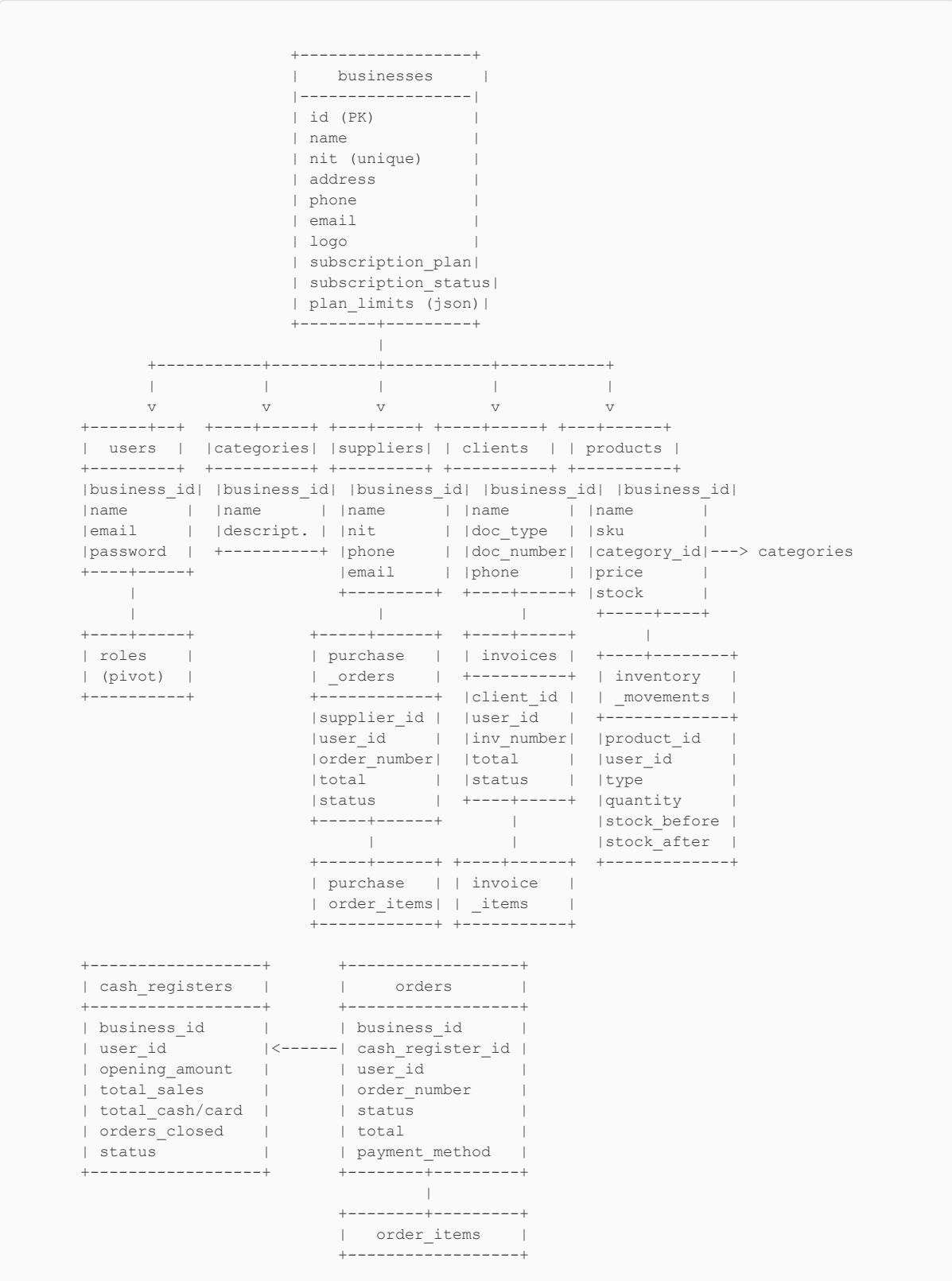
2. Stack Tecnologico

Componente	Tecnologia	Version
Backend Framework	Laravel	10.x
Lenguaje	PHP	8.1+

Base de Datos	PostgreSQL	15+
Autenticacion	Laravel Sanctum	Token-based
Frontend	Vue.js	3.x
reCAPTCHA	Google reCAPTCHA v3	-
Almacenamiento	Laravel Storage (local)	public disk

3. Modelo de Base de Datos

Diagrama de Entidades



```
                                | order_id          |
                                | product_id        |
                                | quantity           |
                                | unit_price         |
                                +-------------------+

        +-------------------+
        |   audit_logs      |
        +-------------------+
        | business_id       |
        | user_id           |
        | entity_type       |
        | entity_id         |
        | action            |
        | old_values (json) |
        | new_values (json) |
        | ip_address        |
        | user_agent        |
        +-------------------+
```

Detalle de Tablas

businesses

Columna	Tipo	Restriccion
id	bigint	PK, auto-increment
name	varchar(255)	NOT NULL
nit	varchar(255)	UNIQUE
address	varchar(255)	nullable
phone	varchar(255)	nullable
email	varchar(255)	nullable
logo	varchar(255)	nullable
subscription_plan	varchar	CHECK: basic|pro|enterprise
subscription_status	varchar	CHECK: active|suspended|cancelled
plan_limits	jsonb	nullable

users

Columna	Tipo	Restriccion
id	bigint	PK
business_id	bigint	FK -> businesses, nullable
name	varchar(255)	NOT NULL
email	varchar(255)	UNIQUE global
password	varchar(255)	hashed (bcrypt)

products

Columna	Tipo	Restriccion

id	bigint	PK
business_id	bigint	FK -> businesses
category_id	bigint	FK -> categories
name	varchar(255)	NOT NULL
sku	varchar(50)	UNIQUE (business_id, sku)
purchase_price	decimal(14,2)	NOT NULL
sale_price	decimal(14,2)	NOT NULL
stock	integer	default 0
min_stock	integer	default 0
is_active	boolean	default true

orders (POS)

Columna	Tipo	Restriccion
id	bigint	PK
business_id	bigint	FK -> businesses
user_id	bigint	FK -> users
cash_register_id	bigint	FK -> cash_registers, nullable
order_number	varchar	UNIQUE (business_id, order_number)
status	varchar	CHECK: open|closed|cancelled
total	decimal(14,2)	default 0
payment_method	varchar	nullable (cash|card|transfer|qr)

cash_registers

Columna	Tipo	Restriccion
id	bigint	PK
business_id	bigint	FK -> businesses
user_id	bigint	FK -> users
opening_amount	decimal(14,2)	default 0
closing_amount	decimal(14,2)	nullable
total_cash	decimal(14,2)	default 0
total_card	decimal(14,2)	default 0
total_transfer	decimal(14,2)	default 0
total_qr	decimal(14,2)	default 0
total_sales	decimal(14,2)	default 0
orders_closed	integer	default 0
status	varchar	CHECK: open|closed

audit_logs

Columna	Tipo	Restriccion
id	bigint	PK
business_id	bigint	FK -> businesses, INDEX
user_id	bigint	FK -> users, nullable
entity_type	varchar	INDEX (entity_type, entity_id)
entity_id	bigint	nullable
action	varchar	NOT NULL
old_values	jsonb	nullable
new_values	jsonb	nullable
ip_address	varchar(45)	nullable
user_agent	varchar	nullable
created_at	timestamp	default CURRENT_TIMESTAMP

Indexes de Rendimiento

Tabla	Index	Columnas
orders	orders_business_status_idx	(business_id, status)
orders	orders_business_status_opened_idx	(business_id, status, opened_at)
orders	orders_cash_register_id_idx	(cash_register_id)
invoices	invoices_business_status_created_idx	(business_id, status, created_at)
cash_registers	cash_registers_user_status_idx	(user_id, status)
products	products_business_active_idx	(business_id, is_active)
purchase_orders	purchase_orders_business_status_idx	(business_id, status)
inventory_movements	inventory_movements_business_product_idx	(business_id, product_id)
inventory_movements	inventory_movements_business_created_idx	(business_id, created_at)
audit_logs	audit_logs_business_created_idx	(business_id, created_at)
audit_logs	audit_logs_entity_idx	(entity_type, entity_id)

4. Modelos y Relaciones

Modelo	Tabla	Trait Multi-Tenant	Relaciones
Business	businesses	No (es el padre)	hasMany: users, categories, suppliers, clients, products, orders, invoices, purchaseOrders, inventoryMovements, cashRegisters
User	users	No (manual)	belongsTo: business | belongsToMany: roles | hasMany: orders, invoices, purchaseOrders, inventoryMovements
Category	categories	Si	belongsTo: business | hasMany: products
Supplier	suppliers	Si	belongsTo: business | hasMany: purchaseOrders
Client	clients	Si	belongsTo: business | hasMany: invoices
Product	products	Si	belongsTo: business, category | hasMany: inventoryMovements
Order	orders	Si	belongsTo: user, cashRegister, business | hasMany: items
Invoice	invoices	Si	belongsTo: client, user, business | hasMany: items
PurchaseOrder	purchase_orders	Si	belongsTo: supplier, user, business | hasMany: items
CashRegister	cash_registers	Si	belongsTo: user, business | hasMany: orders
InventoryMovement	inventory_movements	Si	belongsTo: product, user, business
AuditLog	audit_logs	Si	belongsTo: user, business

5. Endpoints de la API

Autenticacion

Metodo	Ruta	Controller	Auth
POST	/api/login	AuthController@login	Publica (reCAPTCHA)
POST	/api/logout	AuthController@logout	Sanctum
GET	/api/me	AuthController@me	Sanctum

Admin (role:admin)

Metodo	Ruta	Controller
GET	/api/users	UserController@index
GET	/api/audit-logs	AuditLogController@index

Catalogos (CRUD completo)

Metodo	Ruta	Controller
GET	/api/categories	CategoryController@index
POST	/api/categories	CategoryController@store
GET	/api/categories/{id}	CategoryController@show
PATCH	/api/categories/{id}	CategoryController@update
DEL	/api/categories/{id}	CategoryController@destroy
GET	/api/suppliers	SupplierController@index
POST	/api/suppliers	SupplierController@store
GET	/api/suppliers/{id}	SupplierController@show
PATCH	/api/suppliers/{id}	SupplierController@update
DEL	/api/suppliers/{id}	SupplierController@destroy
GET	/api/clients	ClientController@index
POST	/api/clients	ClientController@store
GET	/api/clients/{id}	ClientController@show
PATCH	/api/clients/{id}	ClientController@update
DEL	/api/clients/{id}	ClientController@destroy

Productos

Metodo	Ruta	Controller	Notas
GET	/api/products	ProductController@index	Filtros: search, category_id, low_stock, is_active
POST	/api/products	ProductController@store	Audit log
GET	/api/products/{id}	ProductController@show	
PATCH	/api/products/{id}	ProductController@update	Audit log (old/new)
DEL	/api/products/{id}	ProductController@destroy	Solo si stock == 0

POS - Ordenes

Metodo	Ruta	Controller	Notas
POST	/api/orders	OrderController@store	Requiere caja abierta
GET	/api/orders/open	OrderController@open	Lista ordenes abiertas
POST	/api/orders/{id}/add-item	OrderController@addItem	Descuenta stock
DEL	/api/orders/{id}/remove-item/{item}	OrderController@removeItem	Devuelve stock
POST	/api/orders/{id}/close	OrderController@close	Requiere payment_method
POST	/api/orders/{id}/cancel	OrderController@cancel	Devuelve todo el stock

Caja Registradora

Metodo	Ruta	Controller
POST	/api/cash-registers/open	CashRegisterController@open
GET	/api/cash-registers/current	CashRegisterController@current
POST	/api/cash-registers/{id}/close	CashRegisterController@close
GET	/api/cash-registers/{id}/report	CashRegisterController@report

Facturas, Compras, Inventario, Reportes, Config

Metodo	Ruta	Controller
GET	/api/invoices	InvoiceController@index
POST	/api/invoices	InvoiceController@store
PATCH	/api/invoices/{id}/cancel	InvoiceController@cancel
GET	/api/purchase-orders	PurchaseOrderController@index
POST	/api/purchase-orders	PurchaseOrderController@store
PATCH	/api/purchase-orders/{id}/receive	PurchaseOrderController@receive
GET	/api/inventory-movements	InventoryMovementController@index
POST	/api/inventory-movements	InventoryMovementController@store

GET	/api/reports/daily	ReportController@daily
GET	/api/dashboard	DashboardController@summary
GET	/api/business/settings	BusinessSettingsController@show
POST	/api/business/settings	BusinessSettingsController@update

6. Servicios (Logica de Negocio)

Servicio	Metodos	Responsabilidad
OrderService	createOrder, addItem, removeItem, closeOrder, cancelOrder	Flujo POS completo, stock, caja registradora
InvoiceService	createInvoice, cancelInvoice	Facturacion con IVA 19%, stock
InventoryService	increaseStock, decreaseStock, adjustStock	Control de inventario con trazabilidad
CashRegisterService	open, close, current, report	Gestion de caja registradora
AuditService	log	Registro centralizado de auditoria
RecaptchaService	verify	Validacion reCAPTCHA v3

7. Middleware y Seguridad

Middleware	Alias	Funcion
Authenticate	auth	Valida token Sanctum (Bearer)
RoleMiddleware	role	Verifica rol del usuario (admin, etc.)
CheckPlanLimits	plan.limits	Verifica suscripcion activa y limites del plan
HandleCors	(global)	Permite localhost:8080 y 8081
ThrottleRequests	throttle	Rate limiting en API

8. Multi-Tenancy

> **Patron:** Aislamiento por `business_id` usando Eloquent Global Scopes. Cada query se filtra automaticamente por el negocio del usuario autenticado.

Trait BelongsToBusiness

```php
trait BelongsToBusiness
{
    public static function bootBelongsToBusiness(): void
    {
        // Auto-filtrar por business_id del usuario autenticado
        static::addGlobalScope('business', function (Builder $builder) {
            if (auth()->check()) {
                $table = $builder->getModel()->getTable();
                $builder->where("{$table}.business_id", auth()->user()->business_id);
            }
        });

        // Auto-asignar business_id al crear
```

```
        static::creating(function ($model) {
            if (auth()->check() && empty($model->business_id)) {
                $model->business_id = auth()->user()->business_id;
            }
        });
    }
}
```

9. Guia: Como Crear un Endpoint Paso a Paso

> **Ejemplo:** Crearemos un endpoint `GET /api/promotions` para listar promociones del negocio actual, siguiendo los patrones del proyecto.

Paso 1: Crear la Migracion

```
php artisan make:migration create_promotions_table
```

Editar el archivo generado en `database/migrations/`:

```
public function up(): void
{
    Schema::create('promotions', function (Blueprint $table) {
        $table->id();
        $table->foreignId('business_id')->constrained()->cascadeOnDelete();
        $table->foreignId('product_id')->constrained()->cascadeOnDelete();
        $table->string('name');
        $table->decimal('discount_percent', 5, 2);
        $table->date('start_date');
        $table->date('end_date');
        $table->boolean('is_active')->default(true);
        $table->timestamps();

        // Index de rendimiento
        $table->index(['business_id', 'is_active']);
    });
}
```

Ejecutar:

```
php artisan migrate
```

Paso 2: Crear el Modelo

Crear `app/Models/Promotion.php`:

```
<?php

namespace App\Models;

use App\Models\Traits\BelongsToBusiness;  // <-- CLAVE para multi-tenancy
use Illuminate\Database\Eloquent\Model;

class Promotion extends Model
{
    use BelongsToBusiness;   // <-- Esto auto-filtra por business_id

    protected $fillable = [
        'business_id',       // <-- Siempre incluir
        'product_id',
```

```
        'name',
        'discount_percent',
        'start_date',
        'end_date',
        'is_active',
    ];

    protected $casts = [
        'discount_percent' => 'decimal:2',
        'start_date' => 'date',
        'end_date' => 'date',
        'is_active' => 'boolean',
    ];

    public function product()
    {
        return $this->belongsTo(Product::class);
    }
}
```

Paso 3: Crear el Controller

Crear `app/Http/Controllers/Api/PromotionController.php`:

```
<?php

namespace App\Http\Controllers\Api;

use App\Http\Controllers\Controller;
use App\Models\Promotion;
use App\Services\AuditService;
use Illuminate\Http\Request;

class PromotionController extends Controller
{
    protected AuditService $auditService;

    public function __construct(AuditService $auditService)
    {
        $this->auditService = $auditService;
    }

    /**
     * Listar promociones con filtros y paginacion.
     */
    public function index(Request $request)
    {
        // 1. Validar parametros de entrada
        $request->validate([
            'is_active' => 'nullable|boolean',
            'per_page'  => 'nullable|integer|min:1|max:100',
        ]);

        // 2. Construir query (Global Scope ya filtra por business_id)
        $query = Promotion::with('product');  // Evitar N+1

        if ($request->has('is_active')) {
            $query->where('is_active', $request->boolean('is_active'));
        }

        // 3. Paginar y retornar
        $perPage = $request->integer('per_page', 50);

        return response()->json(
            $query->orderByDesc('created_at')->paginate($perPage)
```

```
        );
    }

    /**
     * Crear nueva promocion.
     */
    public function store(Request $request)
    {
        // Validar con scope de business_id
        $validated = $request->validate([
            'product_id'       => 'required|exists:products,id,business_id,'
                                  . $request->user()->business_id,
            'name'             => 'required|string|max:255',
            'discount_percent' => 'required|numeric|min:0|max:100',
            'start_date'       => 'required|date',
            'end_date'         => 'required|date|after_or_equal:start_date',
        ]);

        $promotion = Promotion::create($validated);

        // Registrar auditoria
        $this->auditService->log(
            'Promotion',
            $promotion->id,
            'created',
            null,
            ['name' => $promotion->name]
        );

        return response()->json(
            $promotion->load('product'), 201
        );
    }
}
```

Paso 4: Registrar la Ruta

Editar `routes/api.php`:

```
// 1. Agregar el import arriba del archivo
use App\Http\Controllers\Api\PromotionController;

// 2. Agregar la ruta dentro del grupo auth:sanctum
Route::middleware(['auth:sanctum'])->group(function () {

    // ... rutas existentes ...

    // Promociones
    Route::apiResource('promotions', PromotionController::class)
        ->only(['index', 'store']);
});
```

Paso 5: Probar

```
# Verificar que la ruta existe
php artisan route:list --path=promotions

# Resultado esperado:
```

```
# GET|HEAD  api/promotions  PromotionController@index
# POST      api/promotions  PromotionController@store
```

Checklist para Todo Endpoint Nuevo

#	Paso	Archivo
1	Crear migracion con `business_id` FK + indexes	`database/migrations/`
2	Ejecutar `php artisan migrate`	-
3	Crear modelo con `use BelongsToBusiness` , fillable, casts, relaciones	`app/Models/`
4	Crear controller con validacion, paginacion, audit	`app/Http/Controllers/Api/`
5	Reglas `exists` siempre con `,business_id,{id}`	Controller
6	Registrar ruta en `routes/api.php` dentro de `auth:sanctum`	`routes/api.php`
7	Verificar con `php artisan route:list`	-
8	(Opcional) Crear Service si la logica es compleja	`app/Services/`
9	(Opcional) Agregar seeder para datos de prueba	`database/seeders/`

Reglas criticas:
- SIEMPRE usar `BelongsToBusiness` trait en modelos con `business_id`
- SIEMPRE incluir `'business_id'` en `$fillable`
- SIEMPRE validar `exists` con `,business_id,{user->business_id}` para evitar acceso cross-tenant
- SIEMPRE paginar resultados (max 100 por pagina)
- SIEMPRE usar `DB::transaction` + `lockForUpdate` en operaciones financieras
- Registrar acciones criticas con `AuditService::log()`