

# Technical Architecture & Development Guidelines

## Overview

This is an AI-powered Shopify alternative built with PayloadCMS, Supabase, and deployed on Vercel. Each merchant receives their own isolated deployment with dedicated database and hosting infrastructure.

## Architecture

### Multi-Instance Deployment Model

- **Each website = separate Supabase project + Vercel deployment**
- **Physical isolation** between merchants (no shared databases)
- **Independent scaling** and configuration per merchant
- **Complete data privacy** through infrastructure separation

### Core Stack

- **Frontend/Backend:** Next.js with PayloadCMS
- **Database:** Supabase (PostgreSQL)
- **Hosting:** Vercel
- **Payments:** Stripe, Przelewy24, Autopay
- **Content Management:** PayloadCMS Collections & Globals

## Database Migration Strategy

### Append-Only Migration Principle

✔ **SAFE - Always Use:**

```
sql
```

*-- Add new tables*

```
CREATE TABLE product_variants (  
  id UUID PRIMARY KEY,  
  product_id UUID REFERENCES products(id),  
  name TEXT NOT NULL,  
  price DECIMAL(10,2)  
);
```

*-- Add new columns with defaults*

```
ALTER TABLE products ADD COLUMN has_variants BOOLEAN DEFAULT false;  
ALTER TABLE products ADD COLUMN created_by UUID REFERENCES users(id);
```

*-- Add indexes*

```
CREATE INDEX idx_products_category ON products(category_id);
```

## ✗ DANGEROUS - Never Use:

sql

*-- Destructive changes*

```
ALTER TABLE products DROP COLUMN old_field;  
ALTER TABLE products RENAME COLUMN old_name TO new_name;  
DROP TABLE deprecated_table;
```

*-- Schema restructuring without migration path*

```
ALTER TABLE products ALTER COLUMN price TYPE INTEGER;
```

## Multi-Step Schema Changes

For complex changes, use a 3-step approach:

### Step 1: Add New Structure

sql

```
ALTER TABLE products ADD COLUMN title TEXT;  
UPDATE products SET title = product_name WHERE title IS NULL;
```

### Step 2: Update Application Code

typescript

```
// Handle both old and new data  
const productTitle = product.title || product.product_name;
```

### Step 3: Remove Old Structure (After 2-3 Months)

```
sql
```

```
ALTER TABLE products DROP COLUMN product_name;
```

## Migration Deployment Process

1. **Test migration** on development database
2. **Deploy to staging** environment first
3. **Gradual rollout** to production sites
4. **Monitor** for issues before proceeding
5. **Keep rollback plan** ready

## Data Modeling Best Practices

### Country/Region Handling

**Current Issue:** Enum-based countries require code changes to add new regions (e.g., adding 'sa' for Saudi Arabia).

#### Recommended Migration:

```
typescript
```

```
// Option 1: Move to ISO codes with validation
```

```
type Country = {  
  code: string; // "US", "PL", "DE", "SA" (ISO 3166-1 alpha-2)  
  name: string;  
  currency: string;  
}
```

```
// Option 2: Maintain superset enum with migration template
```

```
// Standard enum-extend migration snippet:
```

```
// ALTER TYPE country_enum ADD VALUE 'sa';
```

```
// UPDATE countries SET supported_regions = array_append(supported_regions, 'sa');
```

## Currency Management

```
typescript
```

```
// Store prices as integers (cents) with currency metadata
```

```
type Price = {  
  amount: number; // 1299 = $12.99  
  currency: string; // "USD", "EUR", "PLN"  
}
```

# AI Agent Safety Guidelines

## What Agents CAN Modify

### Content & Media:

- Product descriptions, prices, titles
- Page content and blog posts
- Images, logos, banners
- SEO metadata

### Store Configuration (via Payload Globals):

- Theme colors, fonts, spacing
- Navigation structure
- Shipping rates and payment methods
- Email templates
- Discount rules and coupons

**Safe Zone Definition:** If it's stored as data in PayloadCMS collections/globals, agents can modify it.

## What Agents CANNOT Modify

### Database Schema:

- Migration files
- Table structures
- Column definitions
- Indexes and constraints

### Core Application Code:

- API endpoints
- Business logic
- Authentication systems
- Payment processing

### Infrastructure:

- Environment variables
- Build/deployment scripts
- Server configurations

# AI Agent Implementation Pattern

typescript

```
// Safe agent modifications
const updateSiteSettings = async (changes: SiteSettingsUpdate) => {
  return await payload.updateGlobal({
    slug: 'site-settings',
    data: {
      primaryColor: changes.primaryColor,
      heroTitle: changes.heroTitle,
      // ... other safe configuration
    }
  });
};

// Preview system for agent changes
const previewChanges = async (merchantId: string, changes: any) => {
  // Create preview environment
  // Apply changes
  // Return preview URL for merchant approval
};
```

## Error Handling & Observability

### Observability Standards

#### Correlation IDs & Structured Logging:

typescript

*// Standard log schema - always include these fields*

```
const logEvent = (message: string, data: {  
  correlationId: string;  
  tenantId: string;  
  orderId?: string;  
  paywall?: 'cash' | 'stripe' | 'autopay' | 'p24';  
  stage?: 'create' | 'payment' | 'fulfill' | 'complete';  
  level: 'info' | 'warn' | 'error';  
  [key: string]: any;  
}) => {  
  logger[data.level](message, data);  
};
```

*// Usage example*

```
const correlationId = `order_${Date.now()}_${Math.random().toString(36)}`;  
logEvent('Order creation started', {  
  correlationId,  
  tenantId: merchantId,  
  paywall: 'stripe',  
  stage: 'create',  
  level: 'info'  
});
```

### Recommended Monitoring:

- **Sentry:** Tag by `tenantId` for merchant-specific error tracking
- **Correlation tracking:** Link all order events by `correlationId`
- **Payment method tracking:** Monitor success rates per `paywall` type

### Error Response Patterns

typescript

```

// Surface specific server errors to forms
try {
  await createOrder(orderData);
} catch (error) {
  if (error.statusCode === 400) {
    // Translate server errors to user-friendly messages
    const errorMap = {
      'COURIER_NOT_FOUND': 'Selected shipping method is not available',
      'SHIPPING_COST_MISSING': 'Unable to calculate shipping cost',
      'INVALID_POSTAL_CODE': 'Please check your postal code format',
      'PAYMENT_METHOD_UNAVAILABLE': 'Payment method temporarily unavailable'
    };

    return {
      error: errorMap[error.code] || error.message || 'Please check your information and try again'
    };
  }

  return { error: 'Something went wrong. Please try again.' };
}

```

## Webhook Idempotency

typescript

```

const processWebhook = async (webhookData: WebhookPayload) => {
  const existingEvent = await payload.find({
    collection: 'webhook-events',
    where: {
      externalId: { equals: webhookData.id }
    }
  });

  if (existingEvent.docs.length > 0) {
    return; // Already processed
  }

  // Process webhook and store event
  await processPayment(webhookData);
  await payload.create({
    collection: 'webhook-events',
    data: { externalId: webhookData.id, processed: true }
  });
};

```

# Payment Integration

## Supported Payment Methods (Exact Values)

typescript

```
type PaywallType = 'cash' | 'stripe' | 'autopay' | 'p24';

// Required environment variables per provider
const paymentConfig = {
  cash: {}, // No keys required - sets order status to "processing"
  stripe: {
    secretKey: process.env.STRIPE_SECRET_KEY,
    webhookSecret: process.env.STRIPE_WEBHOOK_SECRET
  },
  autopay: {
    secretKey: process.env.AUTOPAY_SECRET_KEY
  },
  p24: {
    secretKey: process.env.P24_SECRET_KEY
  }
};
```

## Order Management & Race Conditions

### Idempotent Order Creation

typescript



```
const createOrder = async (orderData: OrderData) => {  
  // Use database transaction for atomicity  
  return await db.transaction(async (tx) => {  
    // Check for existing order with same session/cart ID  
    const existing = await tx.query.orders.findFirst({  
      where: eq(orders.sessionId, orderData.sessionId)  
    });  
  
    if (existing) {  
      return existing; // Return existing order  
    }  
  
    // Create new order with stock reservation  
    const order = await tx.insert(orders).values(orderData);  
    await reserveStock(tx, orderData.items);  
  
    return order;  
  });  
};
```

## Performance & Caching

### Cache Strategy & Tag Management

typescript

```
// Revalidation tag mapping
const cacheTagMap = {
  'products': ['products', 'product-categories', 'homepage'],
  'pages': ['pages', 'navigation'],
  'shopSettings': ['layout', 'theme', 'homepage'],
  'header': ['navigation', 'layout'],
  'footer': ['layout'],
  'paywalls': ['checkout', 'payment-methods']
};

// Cache with proper revalidation
const getProducts = cache(
  async () => {
    return await payload.find({ collection: 'products' });
  },
  ['products'],
  { tags: ['products'], revalidate: 3600 }
);

// Revalidate on content changes using Payload hooks
const afterProductUpdate = ({ doc }) => {
  cacheTagMap.products.forEach(tag => revalidateTag(tag));
};
```

## Security Considerations

### Security & Compliance

#### PII Handling:

typescript

```
// Never log secrets or sensitive data
const sanitizeForLogging = (data: any) => ({
  ...data,
  email: data.email ? maskEmail(data.email) : undefined,
  phone: data.phone ? maskPhone(data.phone) : undefined,
  // Remove any keys containing 'secret', 'key', 'token', 'password'
  ...Object.fromEntries(
    Object.entries(data).filter(([key]) =>
      ![ 'secret', 'key', 'token', 'password' ].some(word =>
        key.toLowerCase().includes(word)
      )
    )
  )
});

// Pre-commit/CI checks for secrets
// Add to .github/workflows or pre-commit hooks:
// - Check for hardcoded API keys
// - Scan for console.log with sensitive patterns
// - Validate environment variable names
```

## Email Configuration:

```
bash

# Production SMTP requirements
SMTP_HOST=smtp.provider.com
SMTP_PORT=587
SMTP_USER=your-app@domain.com
SMTP_PASS=app-specific-password
SMTP_FROM=noreply@yourdomain.com
```

## Multi-Instance Deployment

### Bootstrap Runbook (New Merchant)

1. **Provision Supabase project** → Copy connection string
2. **Set environment variables** → Add to Vercel/hosting platform
3. **Run initial migration** → `npm run db:migrate`
4. **Optional seeding** → `REQUIRES_SEEDING=true npm run seed`
5. **Deploy to Vercel** → Connect GitHub repo
6. **Run smoke checks** → Verify health endpoints and key flows

## Decommission Checklist

- ☐ Export merchant data (orders, products, customers)
- ☐ Revoke all API keys and webhooks
- ☐ Delete Supabase project
- ☐ Remove Vercel deployment
- ☐ Archive code repository
- ☐ Document handoff if migrating to another platform

## Health Check Endpoints

typescript

*// Add to your API routes*

```
export const healthChecks = {  
  basic: () => GET('/api/health'), // 200 OK  
  database: () => GET('/api/health/db'), // Test DB connection  
  payload: () => GET('/api/payload/health'), // PayloadCMS status  
  payments: () => GET('/api/health/payments'), // Test payment provider connectivity  
};
```

*// Post-deployment verification*

```
const verifyDeployment = async () => {  
  // 1. Basic site loads  
  await fetch(`${process.env.NEXT_PUBLIC_SERVER_URL}/`);  
  
  // 2. PayloadCMS accessible  
  await fetch(`${process.env.NEXT_PUBLIC_SERVER_URL}/api/payload/health`);  
  
  // 3. Key collections readable  
  await payload.find({ collection: 'products', limit: 1 });  
  
  // 4. Payment methods configured  
  const paywalls = await payload.findGlobal({ slug: 'paywalls' });  
  assert(paywalls.enabledPaywalls?.length > 0, 'No payment methods configured');  
  
  // 5. Order creation dry-run  
  await validateOrderFlow({ dryRun: true });  
};
```

## Deployment Guidelines

### Environment Variables

bash

*# Required for all environments*

`DATABASE_URI=` *# Supabase connection string*  
`PAYLOAD_SECRET=` *# Required for PayloadCMS seeding/auth*  
`NEXT_PUBLIC_SERVER_URL=` *# Origin only, no path (e.g., https://mystore.com)*  
`REQUIRES_SEEDING=true` *# Enable seeding (dev/staging only)*

*# Payment providers (exact casing required)*

`STRIPE_SECRET_KEY=`  
`STRIPE_WEBHOOK_SECRET=`  
`AUTOPAY_SECRET_KEY=`  
`P24_SECRET_KEY=`

*# Production recommended*

`SENTRY_DSN=` *# Error tracking*  
`SMTP_HOST=` *# Transactional emails*  
`SMTP_USER=`  
`SMTP_PASS=`

## Provider Limitations

⚠ **Important:** Never use `CREATE EXTENSION` in migrations on hosted databases (Supabase, RDS, etc.). Most providers restrict extension creation. Use built-in functions only.

## Database Seeding

### Prerequisites:

- Media URLs must exist before seeding products
- Paywalls global must be present with required payment methods
- Couriers must be enabled and configured
- **Never seed in production** unless initial bootstrap

typescript

```

// Gate seeding with environment check
if (process.env.REQUIRES_SEEDING === 'true') {
  await seedDatabase();
}

// Resilient seeding with prerequisites check
const seedProducts = async () => {
  // Verify prerequisites first
  const paywalls = await payload.findGlobal({ slug: 'paywalls' });
  if (!paywalls || !paywalls.enabledPaywalls?.length) {
    throw new Error('Paywalls must be configured before seeding products');
  }

  const products = productData.map(p => ({
    ...p,
    heroImage: p.heroImage || null, // Handle missing media gracefully
  }));

  await payload.create({ collection: 'products', data: products });

  // Smoke test after seeding
  const productCount = await payload.count({ collection: 'products' });
  console.log(`✅ Seeded ${productCount.totalDocs} products successfully`);
};

```

## Monitoring & Alerting

### Key Metrics to Track

- Order completion rates
- Payment processing errors
- Site performance (Core Web Vitals)
- Database query performance
- Webhook processing delays

### Recommended Tools

- **Error Tracking:** Sentry
- **Performance:** Vercel Analytics
- **Uptime:** Vercel monitoring
- **Database:** Supabase built-in monitoring

## Development Workflow

# Development Workflow

## Migration Safety Rules

```
sql

-- ✅ SAFE: Standard enum extension template
ALTER TYPE country_enum ADD VALUE 'sa';
UPDATE supported_countries SET regions = array_append(regions, 'sa');

-- ⚠️ DANGEROUS: Mark these scripts clearly
-- File: scripts/reset-dev-db.js (DEVELOPMENT ONLY)
-- Add warning comments and environment checks
if (process.env.NODE_ENV === 'production') {
  throw new Error('❌ DANGER: Cannot run reset script in production');
}
```

## Fast Fix Runbooks

### Order Insert Fails with Enum Error:

```
sql

-- Quick fix: Extend enum
ALTER TYPE payment_method_enum ADD VALUE 'new_method';
-- Long-term: Migrate to text fields with validation
```

### Malformed Redirect URL:

```
bash

# Check environment variable format
echo $NEXT_PUBLIC_SERVER_URL
# Should be: https://mystore.com (origin only, no trailing slash or path)
```

### Seeding Fails "Hero > Media Required":

```
typescript

// Add fallback in seed data
const seedData = {
  ...productData,
  heroImage: productData.heroImage || null, // Allow missing media
  media: productData.media?.filter(Boolean) || [] // Filter out missing items
};
```

## Code Review Checklist

- ☐ Migration is append-only
- ☐ No sensitive data in logs
- ☐ Error handling is user-friendly
- ☐ Webhook processing is idempotent
- ☐ Cache invalidation is correct
- ☐ AI agent changes are restricted to safe zones