Información del Proyecto para Pruebas de Software

Restaurante Bambú - Sistema de Pedidos en Línea

INFORMACIÓN OBLIGATORIA

1. Descripción General del Proyecto

¿Qué hace? Sistema web de pedidos en línea para el Restaurante Bambú que permite a los clientes explorar el menú, realizar pedidos con pagos en línea, y a los administradores gestionar productos, categorías y pedidos.

Propósito: Digitalizar el proceso de pedidos del restaurante, ofreciendo una experiencia moderna y cómoda para los clientes, con gestión centralizada para el negocio.

Público Objetivo:

- **Clientes:** Usuarios que desean hacer pedidos de comida en línea
- **Administradores:** Personal del restaurante que gestiona menú y pedidos

Tipo: Aplicación web responsiva (Next.js)

Entregable: MVP funcional completado en 13 días (30 Oct - 12 Nov 2024)

---

2. Alcance de la Versión a Probar

Módulos Incluidos:

Para Clientes:

- ✅ **Autenticación:** Registro, login (credenciales/Google OAuth)
- ✅ **Menú público:** Catálogo con búsqueda y filtros por categoría
- ✅ **Carrito de compras:** Agregar/editar/eliminar productos
- ✅ **Checkout:** Proceso de pago con Stripe
- ✅ **Perfil:** Gestión de información personal y dirección
- ✅ **Historial de pedidos:** Ver pedidos realizados
- ✅ **Contacto:** Integración WhatsApp

Para Administradores:

- ✅ **Gestión de productos:** CRUD completo con imágenes
- ✅ **Gestión de categorías:** CRUD completo
- ✅ **Gestión de usuarios:** Ver y editar permisos admin
- ✅ **Gestión de pedidos:** Ver todos los pedidos, búsqueda
- ✅ **Upload de imágenes:** Subida a AWS S3

Exclusiones (NO probar):

- ❌ Sistema de notificaciones por email
- ❌ Calificaciones y reseñas de productos
- ❌ Programa de lealtad/puntos
- ❌ Cupones de descuento
- ❌ Multi-idioma
- ❌ Reportes analíticos avanzados

---

3. Requisitos Funcionales

ID	Módulo	Funcionalidad	Criterios de Aceptación	Prioridad
RF-01	Autenticación	Registro de usuario	Email único, contraseña hasheada, redirección a login	Alta
RF-02	Autenticación	Inicio de sesión	Login con credenciales o Google OAuth, sesión persistente	Alta
RF-03	Perfil	Gestión de perfil	Actualizar nombre, dirección, teléfono, foto	Media
RF-04	Usuarios	Roles de usuario	Distinguir entre usuario regular y admin	Alta
RF-05	Categorías	Crear categoría	Admin puede crear categorías para organizar menú	Alta
RF-06	Categorías	Editar/eliminar categoría	Admin puede modificar o eliminar categorías	Media
RF-07	Productos	Crear producto	Admin agrega productos con nombre, precio, imagen, categoría	Alta
RF-08	Productos	Editar producto	Admin modifica información de productos existentes	Alta
RF-09	Productos	Eliminar producto	Admin puede eliminar productos del menú	Media
RF-10	Productos	Gestionar extras	Admin define tamaños y extras con precios adicionales	Media
RF-11	Upload	Subir imágenes	Admin sube fotos que se almacenan en AWS S3	Alta
RF-12	Carrito	Agregar al carrito	Usuarios agregan productos seleccionando tamaño/extras	Alta
RF-13	Carrito	Modificar cantidad	Incrementar o reducir cantidad de productos	Alta

ID	Módulo	Funcionalidad	Criterios de Aceptación	Prioridad
RF-14	Carrito	Eliminar del carrito	Remover productos del carrito	Alta
RF-15	Carrito	Calcular subtotal	Sistema calcula automáticamente precio total	Alta
RF-16	Checkout	Proceso de pago	Usuario proporciona dirección y procede a pago	Alta
RF-17	Checkout	Integración Stripe	Sistema genera sesión de pago segura	Alta
RF-18	Checkout	Confirmación de pago	Webhook confirma pago exitoso	Alta
RF-19	Pedidos	Creación de pedido	Pedido se registra tras confirmación de pago	Alta
RF-20	Pedidos	Historial (usuario)	Usuarios ven sus pedidos anteriores	Media
RF-21	Pedidos	Detalle de pedido	Ver productos, precios, dirección, estado	Media
RF-22	Pedidos	Listado (admin)	Admin ve todos los pedidos del sistema	Alta
RF-23	Pedidos	Actualizar estado	Admin cambia estado del pedido	Media
RF-24	Pedidos	Buscar pedidos	Admin busca pedidos por email de cliente	Media

4. Requisitos No Funcionales / SLAs

Usabilidad

ID	Requerimiento	Descripción	Métrica
RNF-01	Interfaz intuitiva	Usuario puede navegar y completar pedido sin ayuda	Tasa de éxito >90%
RNF-02	Diseño responsive	Funciona en móvil, tablet y escritorio	Compatibilidad 100%
RNF-03	Feedback visual	Confirmación inmediata de acciones con toast	<500ms respuesta
RNF-04	Consistencia visual	Estilo coherente con TailwindCSS	100% páginas

Rendimiento

ID	Requerimiento	Descripción	Umbral

ID	Requerimiento	Descripción	Umbral
RNF-05	Tiempo de carga	Páginas principales cargan rápido	<2 segundos
RNF-06	Optimización imágenes	Next.js optimiza imágenes automáticamente	Compresión WebP
RNF-07	Server Side Rendering	Páginas de menú usan SSR para SEO	100% páginas públicas

Seguridad

ID	Requerimiento	Descripción	Implementación
RSEG-01	Protección contraseñas	Contraseñas hasheadas	bcrypt (10 rounds)
RSEG-02	Autenticación segura	NextAuth con tokens seguros	httpOnly cookies
RSEG-03	Autorización por rol	Rutas admin verifican permisos	Middleware en APIs
RSEG-04	Validación de datos	Inputs validados en servidor	Validación mongoose
RSEG-05	Protección API keys	Claves en variables de entorno	.env nunca committeado
RSEG-06	Webhooks verificados	Webhooks verifican firma Stripe	Firma secreta

Fiabilidad

ID	Requerimiento	Descripción
RFIA-01	Manejo de errores	Sistema captura errores sin exponer info sensible
RFIA-02	Transacciones atómicas	Pedidos se crean completamente o no se crean
RFIA-03	Validación de pagos	Solo crear pedidos tras confirmación webhook
RFIA-04	Disponibilidad BD	Manejo apropiado si MongoDB no disponible

---

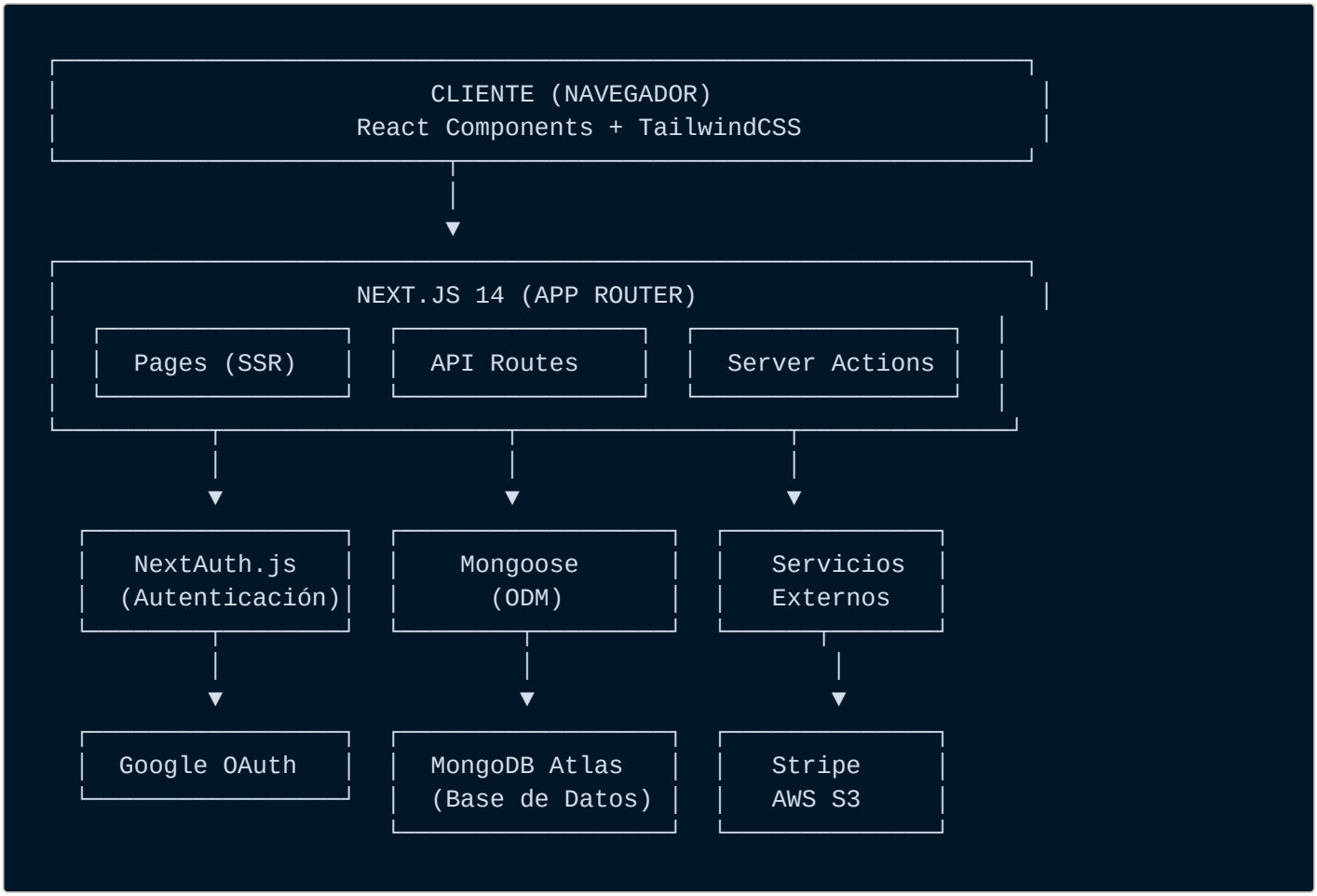
5. Arquitectura Técnica y Stack

Stack Tecnológico

Capa	Tecnología	Versión	Justificación
Frontend	Next.js	14.0.0	SSR, API Routes, optimización automática, SEO
	React	18	Componentes reutilizables, estado reactivo
	TailwindCSS	3.x	Desarrollo rápido UI, responsive
Backend	Next.js API Routes	14.0.0	Backend integrado, serverless-ready
	Node.js	v22.19.0	Runtime JavaScript
Base de Datos	MongoDB	6.2.0	NoSQL flexible, escalable
	Mongoose	7.6.3	ODM, validación schemas

Capa	Tecnología	Versión	Justificación
Autenticación	NextAuth.js	4.24.4	Multi-provider, sesiones seguras
Pagos	Stripe	14.3.0	Plataforma confiable para pagos
Storage	AWS S3	SDK 3.438.0	Almacenamiento escalable imágenes
Seguridad	bcryptjs	3.0.2	Hashing contraseñas
Otros	react-hot-toast	2.4.1	Notificaciones UI

Arquitectura de Alto Nivel



Estructura de Directorios

```
restaurante-bambu/
├── src/
│   ├── app/                    # App Router (Next.js 14)
│   │   ├── api/                # API Routes (Backend)
│   │   │   ├── auth/           # NextAuth endpoints
│   │   │   ├── register/       # POST /api/register
│   │   │   ├── profile/        # GET/PUT /api/profile
│   │   │   ├── categories/     # CRUD /api/categories
│   │   │   ├── menu-items/     # CRUD /api/menu-items
```

```
│   │   │   ├── users/           # GET/PUT /api/users
│   │   │   ├── checkout/        # POST /api/checkout
│   │   │   ├── orders/          # GET /api/orders
│   │   │   ├── webhook/         # POST /api/webhook (Stripe)
│   │   │   └── upload/          # POST /api/upload (S3)
│   │   ├── login/               # Página login
│   │   ├── register/            # Página registro
│   │   ├── menu/                # Menú público
│   │   ├── cart/                # Carrito
│   │   ├── orders/              # Historial pedidos
│   │   ├── profile/             # Perfil usuario
│   │   ├── menu-items/          # Gestión productos (admin)
│   │   ├── categories/          # Gestión categorías (admin)
│   │   └── users/               # Gestión usuarios (admin)
│   ├── components/              # Componentes React reutilizables
│   │   ├── layout/              # Header, Hero, Footer
│   │   ├── menu/                # Componentes de menú
│   │   └── icons/               # Iconos SVG
│   ├── models/                  # Modelos Mongoose
│   │   ├── User.js
│   │   ├── UserInfo.js
│   │   ├── MenuItem.js
│   │   ├── Category.js
│   │   └── Order.js
│   └── libs/                    # Utilidades
│       └── mongoConnect.js
├── public/                      # Archivos estáticos
├── scripts/                     # Scripts de testing
│   ├── smoke-tests.mjs          # Pruebas funcionales
│   └── mock-loader.mjs          # Mocks para testing
└── .env                         # Variables de entorno

Componentes: 25+
APIs: 10 endpoints
Modelos: 5 colecciones MongoDB
```

---

6. Endpoints / API Contract

Autenticación

Método	Endpoint	Descripción	Request Body	Response	Auth
POST	`/api/register`	Registro de usuario	`{email, password, name}`	`{success: true}`	No
POST	`/api/auth/[...nextauth]`	Login NextAuth	Credentials / OAuth	Session	No

Perfil y Usuarios

Método	Endpoint	Descripción	Request Body	Response	Auth
GET	`/api/profile`	Obtener info perfil	-	`{email, streetAddress, city, phone, admin}`	Sí
PUT	`/api/profile`	Actualizar perfil	`{name, image, streetAddress, city, phone}`	`{success: true}`	Sí
GET	`/api/users`	Listar usuarios	-	`[{_id, name, email, admin}]`	Admin
PUT	`/api/users`	Actualizar usuario	`{_id, admin}`	`{success: true}`	Admin

Categorías

Método	Endpoint	Descripción	Request Body	Response	Auth
GET	`/api/categories`	Listar categorías	-	`[{_id, name}]`	No
POST	`/api/categories`	Crear categoría	`{name}`	`{_id, name}`	Admin
PUT	`/api/categories`	Actualizar categoría	`{_id, name}`	`{success: true}`	Admin
DELETE	`/api/categories`	Eliminar categoría	`{_id}`	`{success: true}`	Admin

Productos del Menú

Método	Endpoint	Descripción	Request Body	Response	Auth
GET	`/api/menu-items`	Listar productos	-	`[{_id, name, description, basePrice, image, category, sizes, extraIngredientPrices}]`	No
POST	`/api/menu-items`	Crear producto	`{name, description, basePrice, image, category, sizes, extraIngredientPrices}`	`{_id, ...}`	Admin
PUT	`/api/menu-items`	Actualizar producto	`{_id, name, description, basePrice, image, category, sizes, extraIngredientPrices}`	`{success: true}`	Admin
DELETE	`/api/menu-items`	Eliminar producto	`{_id}`	`{success: true}`	Admin

Checkout y Pedidos

Método	Endpoint	Descripción	Request Body	Response	Auth
POST	`/api/checkout`	Crear sesión pago	`{cartProducts, address: {phone, streetAddress, city}}`	`{url: stripeCheckoutUrl}`	Sí
POST	`/api/webhook`	Webhook Stripe	Stripe Event	-	No (firma verificada)
GET	`/api/orders`	Listar pedidos	`?email=user@example.com` (opcional)	`[{_id, userEmail, cartProducts, paid, createdAt, streetAddress, city, phone}]`	Sí

Upload

Método	Endpoint	Descripción	Request Body	Response	Auth
POST	`/api/upload`	Subir imagen a S3	`FormData {file}`	`{url: s3ImageUrl}`	Sí

7. Esquema de Base de Datos (MongoDB)

Colección: `users`

```
{
  _id: ObjectId,
  name: String,
  email: String (unique, required),
  password: String (hashed),
  image: String (URL),
  createdAt: Date,
  updatedAt: Date
}
```

Índices:

- `email`: unique
- `_id`: primary key

Restricciones:

- `email`: formato email válido, único
- `password`: hasheado con bcrypt (10 rounds)

Colección: `userinfos`

```
{
  _id: ObjectId,
  email: String (required),
  streetAddress: String,
  postalCode: String,
  city: String,
  country: String,
  phone: String,
  admin: Boolean (default: false),
  createdAt: Date,
  updatedAt: Date
}
```

Relaciones:

- `email` relaciona con `users.email`

Colección: `categories`

```
{
  _id: ObjectId,
  name: String (required),
  createdAt: Date,
  updatedAt: Date
}
```

Colección: `menuitems`

```
{
  _id: ObjectId,
  name: String,
  description: String,
  basePrice: Number,
  image: String (S3 URL),
  category: ObjectId (ref: categories),
  sizes: [{
    name: String,
    price: Number
  }],
  extraIngredientPrices: [{
    name: String,
    price: Number
  }],
  createdAt: Date,
```

```
  updatedAt: Date
}
```

Relaciones:

- `category` referencia a `categories._id`

Cálculo de precio:

```
precioTotal = basePrice + size.price + sum(extras.price)
```

Colección: `orders`

```
{
  _id: ObjectId,
  userEmail: String,
  phone: String,
  streetAddress: String,
  postalCode: String,
  city: String,
  country: String,
  cartProducts: Object (array de productos con cantidades),
  paid: Boolean (default: false),
  createdAt: Date,
  updatedAt: Date
}
```

Relaciones:

- `userEmail` relaciona con `users.email`

---

8. Accesos y Artefactos

Repositorio

- **URL:** Local (`/home/davidmanuel/Pruebas/ing-software/restaurante-bambu`)
- **Branch principal:** `main`
- **Total commits:** 20
- **Período desarrollo:** 30 Oct - 12 Nov 2024

Entorno de Desarrollo

- **URL:** `http://localhost:3000`
- **Node version:** v22.19.0
- **Package manager:** npm / pnpm

Comandos de Instalación

```
# Clonar repositorio
cd /home/davidmanuel/Pruebas/ing-software/restaurante-bambu

# Instalar dependencias
npm install

# Configurar variables de entorno
cp .env.example .env
# Editar .env con credenciales reales

# Ejecutar en desarrollo
npm run dev

# Ejecutar pruebas funcionales
npm run smoke
```

Variables de Entorno Requeridas

```
MONGO_URL="mongodb+srv://user:pass@cluster.mongodb.net/restaurante-bambu"
NEXTAUTH_URL="http://localhost:3000"
SECRET="nextauth-secret-key-here"
GOOGLE_CLIENT_ID="google-oauth-client-id"
GOOGLE_CLIENT_SECRET="google-oauth-client-secret"
MY_AWS_ACCESS_KEY="aws-access-key"
MY_AWS_SECRET_KEY="aws-secret-key"
STRIPE_SK="sk_test_..."
STRIPE_PK="pk_test_..."
```

⚠️ **Nota de seguridad:** Las credenciales reales están en `.env` (no committeado). Para testing usar credenciales de staging/test.

---

9. Casos de Prueba Existentes

Estado Actual de Pruebas

Según `resumen_ejecución_pruebas.md`:

Categoría	Total	Pasaron	Fallaron	Pendientes
Autenticación	2	2	0	0
Dashboard	3	3	0	0
Contenidos	3	3	0	0

Categoría	Total	Pasaron	Fallaron	Pendientes
Usuarios	2	2	0	0
Carrito/Checkout	3	3	0	0
Subidas	1	0	0	1
Usabilidad	4	0	0	4
Seguridad	3	3	0	0
TOTAL	**21**	**16**	**0**	**5**

Tasa de Éxito: 76%

Scripts de Prueba

- **Ubicación:** `/scripts/smoke-tests.mjs`
- **Mocks:** `/scripts/mock-loader.mjs`
- **Comando:** `npm run smoke`

Defectos Conocidos

- Pendiente: Pruebas de upload de imágenes a S3 (requiere credenciales AWS)
- Pendiente: Pruebas de usabilidad (4 casos)

INFORMACIÓN RECOMENDADA

10. Wireframes / Mockups

Páginas Principales

1. **Home (`/`)**
    - Hero con llamado a acción
    - Preview de productos destacados
    - Sección "Nuestra Historia"
    - Botón WhatsApp flotante
2. **Menú (`/menu`)**
    - Catálogo de productos con imágenes
    - Búsqueda en tiempo real
    - Filtros por categoría
    - Cards responsive
3. **Carrito (`/cart`)**
    - Lista de productos seleccionados

- Controles de cantidad
- Cálculo de subtotal
- Botón "Proceder al pago"

4. **Perfil (`/profile`)**
   - Formulario de información personal
   - Upload de foto de perfil
   - Dirección de entrega predeterminada
5. **Panel Admin (`/menu-items`, `/categories`, `/users`, `/orders`)**
   - Tablas con CRUD
   - Formularios de edición
   - Búsqueda y filtros

---

11. Casos de Uso y Diagramas de Secuencia

Caso de Uso 1: Realizar un Pedido

Actores: Cliente, Sistema, Stripe

Flujo Principal:

1. Cliente navega por el menú
2. Selecciona producto y configura tamaño/extras
3. Agrega al carrito
4. Revisa carrito y procede al checkout
5. Proporciona dirección de entrega
6. Sistema crea sesión de Stripe
7. Cliente es redirigido a Stripe Checkout
8. Cliente completa pago con tarjeta
9. Stripe envía webhook de confirmación
10. Sistema crea pedido en BD con `paid: true`
11. Cliente ve confirmación del pedido

Diagrama de Secuencia:

```
Cliente → [Menu] → Selecciona Producto
Cliente → [Cart] → Agrega al Carrito (localStorage)
Cliente → [Cart] → "Proceder al Pago"
Cliente → [API /checkout] → {cartProducts, address}
API → Stripe API → Crear Checkout Session
Stripe → Cliente → Redirección a checkout.stripe.com
Cliente → Stripe → Completa pago
Stripe → [API /webhook] → payment_intent.succeeded
API /webhook → MongoDB → Crea Order {paid: true}
MongoDB → API → Orden creada
Cliente → [/orders/{id}] → Ve confirmación
```

Caso de Uso 2: Gestionar Productos (Admin)

Actores: Administrador, Sistema, AWS S3

Flujo Principal:

1. Admin inicia sesión
2. Navega a `/menu-items`
3. Clic en "Crear Nuevo Producto"
4. Sube imagen → API envía a S3
5. Completa formulario (nombre, precio, categoría, extras)
6. Guarda producto
7. Sistema valida y guarda en MongoDB
8. Producto aparece en menú público

---

12. Datos de Prueba

Usuarios de Prueba

```
// Admin
{
  "email": "admin@restaurantebambu.com",
  "password": "Admin123!",
  "name": "Administrador Test",
  "admin": true
}

// Cliente
{
  "email": "cliente@example.com",
  "password": "Cliente123!",
  "name": "Cliente Test",
  "admin": false
}
```

Categorías de Prueba

```
[
  { "name": "Platos Principales" },
  { "name": "Bebidas" },
  { "name": "Postres" },
  { "name": "Entradas" }
]
```

Productos de Prueba

```
{
  "name": "Arroz Chaufa",
  "description": "Arroz frito estilo chino con pollo y verduras",
  "basePrice": 25,
  "category": "Platos Principales",
  "sizes": [
    { "name": "Pequeño", "price": 0 },
    { "name": "Mediano", "price": 5 },
    { "name": "Grande", "price": 10 }
  ],
  "extraIngredientPrices": [
    { "name": "Extra carne", "price": 8 },
    { "name": "Extra verduras", "price": 4 }
  ]
}
```

Datos de Checkout

```
{
  "cartProducts": [
    {
      "name": "Arroz Chaufa",
      "basePrice": 25,
      "size": { "name": "Mediano", "price": 5 },
      "extras": [{ "name": "Extra carne", "price": 8 }]
    }
  ],
  "address": {
    "phone": "+591 62294912",
    "streetAddress": "Av. Busch #123",
    "city": "Santa Cruz"
  }
}
```

---

13. Cronograma de Releases

Información del Proyecto

- **Período de Desarrollo:** 30 Oct 2024 - 12 Nov 2024
- **Duración:** 13 días
- **Modalidad:** Sprint único
- **Estado:** MVP completado

Cronología

Semana 1 (30 Oct - 5 Nov):

- Setup inicial y configuración
- Modelos de datos (Mongoose schemas)
- Autenticación (NextAuth + bcrypt)
- CRUD productos y categorías
- Panel admin básico

Semana 2 (6 Nov - 12 Nov):

- Carrito de compras con Context API
- Integración Stripe y checkout
- Sistema de pedidos y webhook
- Perfil de usuario y gestión admin
- Testing y refinamiento

Próximos Pasos (Post-MVP)

- [ ] Notificaciones por email
- [ ] Sistema de calificaciones
- [ ] Reportes analíticos
- [ ] Cupones de descuento

---

14. Herramientas y Preferencia para Automatización

Stack de Testing Recomendado

Tipo de Prueba	Herramienta Recomendada	Justificación
E2E (Frontend)	Playwright o Cypress	Soporte Next.js, testing de flujos completos
API Testing	Jest + Supertest	Integración con Next.js, mocking fácil
Unit Testing	Jest + React Testing Library	Estándar para React, fast feedback
Load Testing	Artillery o k6	APIs RESTful, webhooks
Security Testing	OWASP ZAP	Escaneo automático de vulnerabilidades

Preferencias del Proyecto

- **Lenguaje:** JavaScript/TypeScript (ya usado en el proyecto)
- **Framework de testing:** Jest (ya configurado en Next.js)
- **Testing E2E:** Recomendado Playwright por velocidad
- **CI/CD:** GitHub Actions (sugerido para futura integración)

Estructura de Tests Sugerida

```
tests/
├── e2e/
│   ├── auth.spec.js
│   ├── menu.spec.js
│   ├── cart.spec.js
│   └── checkout.spec.js
├── integration/
│   ├── api/
│   │   ├── auth.test.js
│   │   ├── menu-items.test.js
│   │   └── orders.test.js
│   └── database/
│       └── models.test.js
├── unit/
│   └── components/
│       ├── Header.test.js
│       └── MenuItem.test.js
└── load/
    └── checkout-load.yml
```

---

15. Entorno Objetivo / Matrices de Compatibilidad

Navegadores Soportados

Navegador	Versiones	Prioridad
Chrome	Últimas 2 versiones	Alta
Firefox	Últimas 2 versiones	Alta
Safari	Últimas 2 versiones	Media
Edge	Últimas 2 versiones	Media
Mobile Safari (iOS)	iOS 14+	Alta
Chrome Mobile (Android)	Android 10+	Alta

Dispositivos y Resoluciones

Dispositivo	Resolución	Breakpoint Tailwind
Mobile	320px - 640px	`sm:`
Tablet	640px - 1024px	`md:`, `lg:`
Desktop	1024px+	`xl:`, `2xl:`

Sistema Operativo

- **Desarrollo:** Linux (Ubuntu/similar)

- **Producción:** Compatible con cualquier OS (Next.js deployable a Vercel, AWS, etc.)

---

16. Requisitos de Monitoreo y Logging

Logging Actual

- **Nivel:** Console.log en desarrollo
- **Libraries:** Ninguna específica (usar console nativo)

Recomendaciones para Producción

- **APM:** New Relic, Datadog, o Vercel Analytics
- **Error Tracking:** Sentry
- **Logs estructurados:** Winston o Pino
- **Métricas:**
  - Tiempo de respuesta API
  - Tasa de conversión checkout
  - Errores de webhook Stripe

INFORMACIÓN OPCIONAL

---

17. Métricas de Uso (Estimadas)

- **Usuarios activos objetivo:** 100-500 clientes/mes
- **Pedidos esperados:** 10-50 pedidos/día
- **Productos en catálogo:** 30-100 ítems
- **Páginas más visitadas:** `/menu`, `/cart`, `/checkout`

---

18. Políticas de Backup (Recomendadas)

MongoDB

- **Backup automático:** MongoDB Atlas (si se usa)
- **Frecuencia:** Diaria
- **Retención:** 7 días

Imágenes S3

- **Versionado:** Habilitado
- **Lifecycle policy:** Mantener versiones 30 días

PLANTILLAS Y FORMATOS

Plantilla de Caso de Prueba

Campo	Descripción
ID	CP-XXX
Título	Nombre descriptivo del caso
Módulo	Autenticación / Menú / Carrito / etc.
Tipo	Funcional / No-Funcional
Método	Caja Negra / Caja Blanca / Caja Gris
Prioridad	Alta / Media / Baja
Precondiciones	Estado inicial requerido
Pasos	1. Paso uno 2. Paso dos 3. Paso tres
Datos de Prueba	Inputs específicos
Resultado Esperado	Output esperado
Criterios de Aceptación	Condiciones de éxito
Estado	Pendiente / Pasó / Falló
Observaciones	Notas adicionales

Plantilla de Reporte de Defecto

Campo	Descripción
ID	BUG-XXX
Título	Resumen del bug
Severidad	Crítica / Alta / Media / Baja
Prioridad	Alta / Media / Baja
Entorno	Desarrollo / Staging / Producción
Pasos para Reproducir	1. Paso uno 2. Paso dos
Resultado Actual	Qué ocurre ahora
Resultado Esperado	Qué debería ocurrir
Logs/Stacktrace	Error messages
Evidencias	Screenshots, videos
Asignado a	Desarrollador responsable

Campo	Descripción
Estado	Abierto / En Progreso / Cerrado

REQUISITOS PARA PRUEBAS NO FUNCIONALES

Rendimiento

Perfil de Carga

Escenario	Usuarios Concurrentes	Duración	TPS Objetivo
Carga Normal	50	10 min	5-10
Pico de Tráfico	200	5 min	20-30
Estrés	500	2 min	50+

Métricas a Medir

- **Tiempo de respuesta promedio:** <500ms (APIs)
- **Tiempo de carga página:** <2s (SSR)
- **Percentil 95:** <1s
- **Tasa de error:** <1%

Seguridad

Amenazas a Validar

Amenaza	Prueba	Herramienta
Inyección SQL/NoSQL	Inputs maliciosos	Manual + OWASP ZAP
XSS	Scripts en inputs	Manual + OWASP ZAP
CSRF	Requests no autorizados	Postman
Autenticación débil	Brute force password	Hydra (test env)
Exposición de datos	Revisar responses API	Manual
Storage inseguro	Revisar cookies/localStorage	DevTools

Checklist de Seguridad

- [ ] Contraseñas hasheadas (bcrypt)
- [ ] Sesiones con httpOnly cookies
- [ ] Variables sensibles en .env
- [ ] Validación de inputs en servidor

- [ ] Rate limiting en APIs críticas
- [ ] HTTPS en producción
- [ ] Verificación de webhooks Stripe

Usabilidad

Criterios WCAG (Nivel AA objetivo)

- **Contraste:** Mínimo 4.5:1 para texto
- **Navegación por teclado:** Tab index correcto
- **Alt text:** Imágenes con descripción
- **Labels:** Formularios con labels asociados

Herramientas de Testing

- **Lighthouse:** Auditoría automática
- **axe DevTools:** Accesibilidad
- **Manual:** Testing con lectores de pantalla

ENTREGABLES QUE PRODUCIRÉ

Con la información proporcionada en este documento, puedo generar:

1. Plan de Pruebas Completo

- Alcance y objetivos
- Estrategia (manual/automatizado)
- Criterios de entrada/salida
- Matriz de riesgos
- Configuración de entorno
- Cronograma de ejecución
- Recursos necesarios

2. Matriz de Cobertura

Requisito	Casos de Prueba	Cobertura
RF-01	CP-001, CP-002	100%
RF-02	CP-003, CP-004, CP-005	100%
...	...	...

3. Casos de Prueba Detallados

- Formato Excel/CSV con todas las columnas especificadas
- Mínimo 50+ casos de prueba cubriendo:

    - Autenticación (8 casos)
    - Gestión de productos (12 casos)
    - Carrito y checkout (15 casos)
    - Pedidos (10 casos)
    - Permisos admin (8 casos)

4. Scripts de Automatización

Ejemplos en Playwright/Cypress:

- Login flow
- Agregar producto al carrito
- Proceso completo de checkout
- CRUD de categorías (admin)

5. Plan de Pruebas No Funcionales

- Escenarios de carga (Artillery/k6 config)
- Checklist de seguridad (OWASP Top 10)
- Pruebas de usabilidad (Lighthouse, axe)
- Pruebas de compatibilidad (BrowserStack matrix)

6. Reporte de Ejecución

- Resumen ejecutivo
- Resultados por módulo
- Defectos encontrados
- Métricas de calidad
- Recomendaciones de mitigación
- Prioridades de corrección

CHECKLIST DE INFORMACIÓN PROPORCIONADA

- [x] README + descripción del proyecto
- [x] Alcance de la versión
- [x] Requisitos funcionales (24 RF)
- [x] Requisitos no funcionales / SLAs
- [x] Diagrama de arquitectura
- [x] Esquema de API (10 endpoints documentados)
- [x] DB schema / DDL (5 colecciones)
- [x] Accesos a repo (local)
- [x] Casos de prueba existentes (21 casos, 76% éxito)
- [x] Wireframes / mockups (descripción)
- [x] Cronograma de releases (13 días, completado)
- [x] Preferencias de herramientas de automatización

PRÓXIMOS PASOS

Puedo comenzar a generar:

1. **Plantillas vacías** (Excel/CSV) para que completes con casos adicionales
2. **Plan de pruebas formal** con toda la información recopilada
3. **Scripts de automatización** para los flujos críticos
4. **Matriz de cobertura** enlazando requisitos con casos de prueba

---

Documento generado el: 19 de Noviembre de 2025
Proyecto: Restaurante Bambú - Sistema de Pedidos en Línea
Versión: 1.0 (MVP)
Estado: Listo para diseño de pruebas