



Actividad 6 - Entrega final del proyecto

UNIVERSIAD IBEROAMERICANA

ACTIVIDAD 6

Presentado por:

JHON ALEXANDER RETAMOZA GRANADOS

ID BANNER: 100089450

CORREO INSTITUCIONAL: jretamoz@estudiante.ibero.edu.co

IBEROAMERICANA DE PLANETA FORMACION Y UNIVERSIDADES

FACULTAD DE INGIENERIA

INGIENERIA DE SOFTWARE

BOGOTA, COLOMBIA

2025

Documentación Técnica: Fase de Codificación – OrbisPay

Link video: <https://youtu.be/KUeejhOOipM>

Link Github: <https://github.com/Machine0054/OrbisPay>

1. Arquitectura del Software

1.1. Tipo de Arquitectura Implementada

OrbisPay utiliza una arquitectura de tres capas (Three-Tier Architecture) que separa claramente las responsabilidades del sistema en tres niveles distintos, promoviendo la modularidad, mantenibilidad y escalabilidad del código.

1.2. Descripción de las Capas

Capa de Presentación (Frontend)

Capa de Presentación (Frontend)

Esta capa es responsable de la interfaz de usuario y la experiencia visual. Implementa todos los componentes visuales con los que el usuario interactúa directamente.

Tecnologías utilizadas:

- HTML5, CSS3, JavaScript
- Diseño responsive con enfoque mobile-first

- Sistema de diseño basado en colores corporativos (morado, azul, verde)

Componentes principales:

- DashboardComponent: Visualización del resumen financiero
- TransactionFormComponent: Formulario de registro de transacciones
- BudgetManagerComponent: Gestión de presupuestos
- SavingsGoalComponent: Administración de metas de ahorro
- DebtTrackerComponent: Seguimiento de deudas
- ReportGeneratorComponent: Generación de reportes visuales
- NotificationPanelComponent: Sistema de alertas y notificaciones

Responsabilidades:

- Renderización de la interfaz de usuario
- Validación de entrada de datos del lado del cliente
- Comunicación con la capa de lógica de negocio mediante APIs REST
- Manejo de estados locales de la aplicación
- Actualización reactiva de la interfaz

Capa de Lógica de Negocio (Backend)

Esta capa contiene toda la lógica de procesamiento, cálculos financieros, validaciones y reglas de negocio de la aplicación.

Tecnologías utilizadas:

- Lenguaje: PHP

Módulos principales:

- TransactionService: Procesamiento de ingresos y gastos
- BudgetCalculator: Cálculo de presupuestos y seguimiento de gastos
- SavingsManager: Gestión de metas de ahorro y aportes
- DebtCalculator: Cálculo de intereses y amortización de deudas
- ReportGenerator: Generación de reportes y análisis estadísticos
- NotificationEngine: Sistema de alertas automáticas
- AuthenticationService: Gestión de autenticación y autorización

Responsabilidades:

- Validación de datos del lado del servidor
- Implementación de reglas de negocio
- Procesamiento de cálculos financieros complejos
- Gestión de sesiones y autenticación
- Coordinación entre diferentes módulos
- Generación de respuestas JSON para el frontend

Capa de Datos (Database)

Esta capa es responsable del almacenamiento persistente de toda la información del sistema.

Tecnologías utilizadas:

- Sistema de gestión de base de datos: MySQL

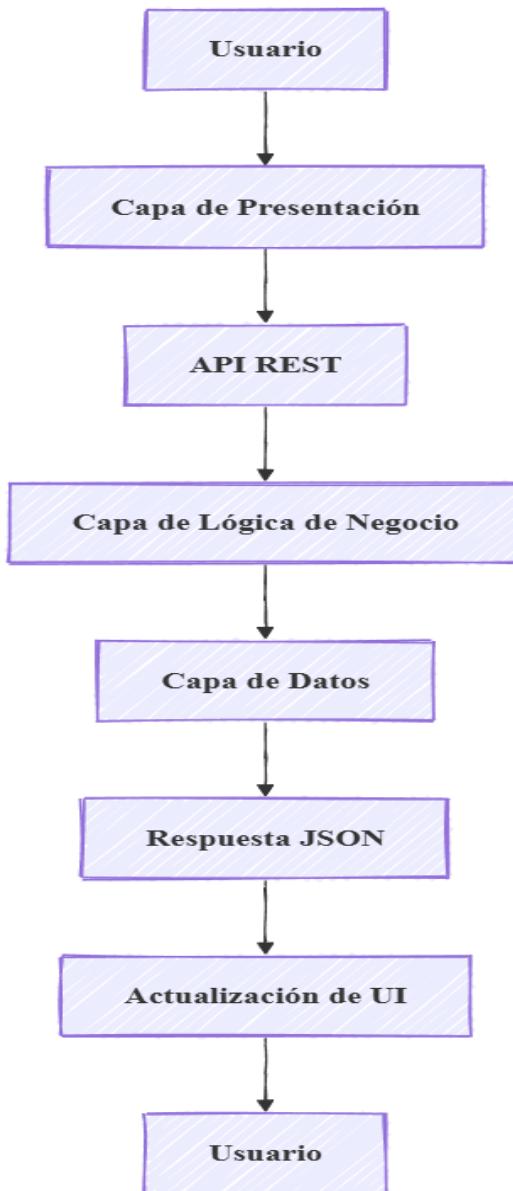
Entidades principales:

- User: Información de usuarios
- Transaction: Registro de ingresos y gastos
- Category: Categorías de transacciones
- Budget: Presupuestos definidos por el usuario
- SavingsGoal: Metas de ahorro
- Debt: Deudas registradas
- Payment: Pagos realizados a deudas
- Notification: Alertas y notificaciones

Responsabilidades:

- Almacenamiento persistente de datos
- Integridad referencial mediante claves foráneas
- Optimización de consultas mediante índices
- Respaldo y recuperación de información

1.3. Flujo de Comunicación entre Capas



1.4. Ventajas de esta Arquitectura

- **Separación de responsabilidades:** Cada capa tiene un propósito específico y bien definido.

- **Mantenibilidad:** Los cambios en una capa no afectan directamente a las otras.

- **Escalabilidad:** Cada capa puede escalarse independientemente según las necesidades.

- **Reutilización:** La lógica de negocio puede ser utilizada por múltiples interfaces (web, móvil, API pública).

- **Testabilidad:** Cada capa puede ser probada de forma independiente.

2. Patrones de Diseño Implementados

2.1. Patrón MVC (Model-View-Controller)

Descripción: Separa la aplicación en tres componentes interconectados para separar la representación interna de la información de la forma en que se presenta al usuario.

Implementación en OrbisPay:

- **Model** (Modelo): Clases que representan las entidades de la base de datos (Transaction, Budget, SavingsGoal, Debt).
- **View** (Vista): Componentes de la interfaz de usuario que renderizan los datos.
- **Controller** (Controlador): Clases que manejan las solicitudes HTTP y coordinan entre modelos y vistas (TransactionController, BudgetController, ReportController).

Ejemplo:

```
class TransactionController {  
    public function store(Request $request) {  
        // Validar datos de entrada  
        $validated = $request->validate([  
            'amount' => 'required|numeric',  
            'category_id' => 'required|exists:categories,id',  
            'type' => 'required|in:income,expense'  
        ]);  
  
        // Crear modelo (interacción con la capa de datos)  
        $transaction = Transaction::create($validated);  
  
        // Retornar vista/respuesta JSON  
        return response()->json($transaction, 201);  
    }  
}
```

2.2. Patrón Singleton

Descripción: Garantiza que una clase tenga una única instancia y proporciona un punto de acceso global a ella.

Implementación en OrbisPay:

Se utiliza para la gestión de la conexión a la base de datos, asegurando que exista una sola conexión activa durante toda la ejecución de la aplicación.

Ejemplo:

```
class DatabaseConnection {
    private static $instance = null;
    private $connection;

    private function __construct() {
        // Crear conexión a la base de datos
        $this->connection = new PDO(
            "mysql:host=localhost;dbname=orbispay",
            "username",
            "password"
        );
    }

    public static function getInstance() {
        if ($self::$instance === null) {
            $self::$instance = new DatabaseConnection();
        }
        return $self::$instance;
    }

    public function getConnection() {
        return $this->connection;
    }
}
```

Beneficio: Evita la creación de múltiples conexiones a la base de datos, optimizando el uso de recursos.

2.3. Patrón Factory (Fábrica)

Descripción: Define una interfaz para crear objetos, pero permite a las subclases decidir qué clase instanciar.

Implementación en OrbisPay:

Se utiliza para la creación de diferentes tipos de notificaciones según el evento que las desencadena.

Ejemplo:

```
class NotificationFactory {
    public static function create($type, $data) {
        switch($type) {
            case 'budget_exceeded':
                return new BudgetExceededNotification($data);
            case 'payment_due':
                return new PaymentDueNotification($data);
            case 'goal_completed':
                return new GoalCompletedNotification($data);
            default:
                throw new Exception("Tipo de notificación no válido");
        }
    }
}
```

Uso

```
$notification = NotificationFactory::create('budget_exceeded', [
```

```
'budget_name' => 'Alimentación',
'exceeded_by' => 15
]);
$notification->send();
```

Beneficio: Centraliza la lógica de creación de objetos y facilita la extensión del sistema con nuevos tipos de notificaciones.

2.4. Patrón Observer (Observador)

Descripción: Define una dependencia uno-a-muchos entre objetos, de manera que cuando un objeto cambia de estado, todos sus dependientes son notificados automáticamente.

Implementación en OrbisPay:

Se utiliza para el sistema de notificaciones automáticas. Cuando ocurre un evento (como exceder un presupuesto), todos los observadores registrados son notificados.

Ejemplo:

```
class BudgetObserver {
    public function updated(Budget $budget) {
        if ($budget->isExceeded()) {
            NotificationFactory::create('budget_exceeded', [
                'budget_name' => $budget->category->name,
                'exceeded_by' => $budget->getExceededPercentage()
            ])->send();
        }
    }
}

// Registro del observador
Budget::observe(BudgetObserver::class);
```

Beneficio: Permite una arquitectura reactiva donde los cambios en el estado de un objeto desencadenan automáticamente acciones en otros componentes.

2.5. Patrón Repository

Descripción: Encapsula la lógica de acceso a datos, proporcionando una interfaz más orientada a objetos para interactuar con la base de datos.

Implementación en OrbisPay:

Ejemplo:

```
interface TransactionRepositoryInterface {
    public function findById($id);
    public function findByUser($userId);
    public function findByDateRange($startDate, $endDate);
    public function create(array $data);
    public function update($id, array $data);
    public function delete($id);
}

class TransactionRepository implements TransactionRepositoryInterface {
    public function findByUser($userId) {
        return Transaction::where('user_id', $userId)
            ->orderBy('date', 'desc')
            ->get();
    }

    public function findByDateRange($startDate, $endDate) {
        return Transaction::whereBetween('date', [$startDate, $endDate])
            ->get();
    }
}
```

Beneficio: Separa la lógica de acceso a datos de la lógica de negocio, facilitando el testing y el mantenimiento.

3. Principios SOLID Aplicados

3.1. S - Single Responsibility Principle (Principio de Responsabilidad Única)

Definición: Una clase debe tener una única razón para cambiar, es decir, debe tener una sola responsabilidad.

Aplicación en OrbisPay:

Cada servicio y controlador tiene una responsabilidad específica:

- TransactionService: Solo se encarga de la lógica de transacciones.
- BudgetCalculator: Solo calcula y verifica presupuestos.
- ReportGenerator: Solo genera reportes.

Ejemplo:

// ✗ INCORRECTO - Múltiples responsabilidades

```

class UserManager {
    public function createUser($data) { }
    public function sendEmail($user) { }
    public function generateReport($user) { }
    public function calculateBudget($user) { }
}

// ✅ CORRECTO - Responsabilidad única
class UserService {
    public function createUser($data) { }
    public function updateUser($id, $data) { }
}

class EmailService {
    public function sendWelcomeEmail($user) { }
    public function sendNotification($user, $message) { }
}

class ReportService {
    public function generateUserReport($user) { }
}

```

Beneficio: Facilita el mantenimiento y reduce el acoplamiento entre componentes.

3.2. O - Open/Closed Principle (Principio Abierto/Cerrado)

Definición: Las entidades de software deben estar abiertas para extensión pero cerradas para modificación.

Aplicación en OrbisPay:

El sistema de notificaciones está diseñado para permitir agregar nuevos tipos de notificaciones sin modificar el código existente.

Ejemplo:

```
abstract class Notification {
    protected $data;

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

    abstract public function getMessage();
    abstract public function getType();

    public function send() {
        // Lógica común de envío
        $this->save();
        $this->notify();
    }
}

// Extensiones sin modificar la clase base
class BudgetExceededNotification extends Notification {
    public function getMessage() {
        return "Presupuesto {$this->data['budget_name']} excedido en {$this->data['exceeded_by']}%";
    }

    public function getType() {
        return 'warning';
    }
}

class GoalCompletedNotification extends Notification {
    public function getMessage() {
        return "|Felicitaciones! Meta {$this->data['goal_name']} completada";
    }

    public function getType() {
        return 'success';
    }
}
```

Beneficio: Permite agregar nuevas funcionalidades sin riesgo de romper el código

existente.

3.3. L - Liskov Substitution Principle (Principio de Sustitución de Liskov)

Definición: Los objetos de una clase derivada deben poder reemplazar objetos de la clase base sin alterar el correcto funcionamiento del programa.

Aplicación en OrbisPay:

Todas las clases de notificación pueden ser usadas intercambiablemente donde se espera una Notification.

Ejemplo:

```
class NotificationSender {
    public function sendNotification(Notification $notification) {
        // Funciona con cualquier tipo de notificación
        $notification->send();
    }
}

// Uso
$sender = new NotificationSender();
$sender->sendNotification(new BudgetExceededNotification($data));
$sender->sendNotification(new GoalCompletedNotification($data));
$sender->sendNotification(new PaymentDueNotification($data));
```

Beneficio: Garantiza que las abstracciones sean correctas y consistentes.

3.4. I - Interface Segregation Principle (Principio de Segregación de Interfaces)

Definición: Los clientes no deben verse obligados a depender de interfaces que no utilizan.

Aplicación en OrbisPay:

En lugar de tener una interfaz grande con todos los métodos posibles, se crean interfaces específicas para cada necesidad.

Ejemplo:

```
// ✗ INCORRECTO - Interfaz demasiado grande
interface FinancialOperations {
    public function createTransaction();
    public function createBudget();
    public function createSavingsGoal();
    public function createDebt();
    public function generateReport();
    public function sendNotification();
}

// ☑ CORRECTO - Interfaces segregadas
interface TransactionOperations {
    public function createTransaction();
    public function updateTransaction();
    public function deleteTransaction();
}

interface BudgetOperations {
    public function createBudget();
    public function updateBudget();
    public function checkBudgetStatus();
}

interface ReportOperations {
    public function generateIncomeVsExpenseReport();
    public function generateCategoryAnalysis();
}

// Las clases implementan solo las interfaces que necesitan
```

```
class TransactionService implements TransactionOperations {  
    public function createTransaction() { }  
    public function updateTransaction() { }  
    public function deleteTransaction() { }  
}
```

Beneficio: Evita que las clases dependan de métodos que no necesitan, reduciendo el acoplamiento.

3.5. D - Dependency Inversion Principle (Principio de Inversión de Dependencias)

Definición: Los módulos de alto nivel no deben depender de módulos de bajo nivel.
Ambos deben depender de abstracciones.

Aplicación en OrbisPay:

Se utiliza inyección de dependencias para que los servicios dependan de interfaces en lugar de implementaciones concretas.

Ejemplo:

```
// ✗ INCORRECTO - Dependencia directa de implementación concreta
```

```

class ReportService {
    private $repository;

    public function __construct() {
        $this->repository = new TransactionRepository(); // Acoplamiento fuerte
    }
}

// ✅ CORRECTO - Dependencia de abstracción
interface TransactionRepositoryInterface {
    public function findByDateRange($start, $end);
}

class ReportService {
    private $repository;

    public function __construct(TransactionRepositoryInterface $repository) {
        $this->repository = $repository; // Acoplamiento débil
    }

    public function generateMonthlyReport($month, $year) {
        $transactions = $this->repository->findByDateRange(
            "$year-$month-01",
            "$year-$month-31"
        );
        // Procesar y retornar reporte
    }
}

// Inyección de dependencias
$repository = new TransactionRepository();
$reportService = new ReportService($repository);

```

Beneficio: Facilita el testing mediante la inyección de mocks y permite cambiar implementaciones sin modificar el código de alto nivel.

4. Buenas Prácticas de Codificación Implementadas

4.1. Nomenclatura Consistente

- Clases: PascalCase (TransactionController, BudgetService)
- Métodos y funciones: camelCase (createTransaction, calculateBalance)
- Variables: camelCase (\$userId, \$totalAmount)
- Constantes: UPPER_SNAKE_CASE (MAX_BUDGET_AMOUNT, DEFAULT_CURRENCY)

4.2. Validación de Datos

Todas las entradas de usuario son validadas tanto en el frontend como en el backend para garantizar la integridad de los datos.

4.3. Manejo de Errores

Implementación de try-catch para capturar excepciones y proporcionar mensajes de error claros al usuario.

4.4. Comentarios y Documentación

El código incluye comentarios explicativos en secciones complejas y documentación PHPDoc para todas las clases y métodos públicos.

4.5. Control de Versiones

Uso de Git para el control de versiones con commits descriptivos y ramas para diferentes features.

Conclusión

La implementación de OrbisPay sigue una arquitectura sólida y bien estructurada, aplicando patrones de diseño reconocidos y principios SOLID que garantizan un código mantenible, escalable y de alta calidad. La separación clara de responsabilidades entre capas y la aplicación de buenas prácticas de programación orientada a objetos facilitan tanto el desarrollo actual como futuras extensiones del sistema.

Plan de Pruebas de Software – OrbisPay

1. Introducción

El presente documento establece el plan de pruebas para el sistema OrbisPay, una aplicación web diseñada para la gestión financiera personal. El objetivo principal de este plan es verificar que todas las funcionalidades del sistema operen correctamente y cumplan con los requisitos establecidos, garantizando la calidad del software antes de su puesta en producción.

2. Alcance de las Pruebas

2.1. Funcionalidades a Probar

El plan de pruebas cubre los siguientes módulos del sistema OrbisPay:

- Dashboard Interactivo: Visualización del resumen financiero del usuario.
- Gestión de Transacciones: Registro y consulta de ingresos y gastos.
- Gestión de Presupuestos: Creación, seguimiento y alertas de presupuestos.
- Gestión de Metas de Ahorro: Definición y seguimiento de objetivos de ahorro.
- Gestión de Deudas: Registro y control de deudas y pagos.
- Generación de Reportes: Creación de reportes financieros y análisis de gastos.
- Sistema de Notificaciones: Alertas automáticas sobre eventos importantes.

2.2. Funcionalidades Excluidas

Las siguientes funcionalidades no están incluidas en este ciclo de pruebas:

- Integración con APIs de terceros para sincronización bancaria.
- Funcionalidades de la aplicación móvil nativa.

- Pruebas de carga con más de 1000 usuarios simultáneos.

3. Estrategia de Pruebas

3.1. Tipos de Pruebas

Se aplicarán los siguientes tipos de pruebas para garantizar la calidad del sistema:

Pruebas por Nivel

- Pruebas Unitarias: Verificación de funciones individuales del código PHP y JavaScript.
- Pruebas de Integración: Validación de la comunicación entre el frontend, backend y base de datos.

- Pruebas de Sistema: Evaluación del sistema completo en un entorno similar al de producción.

- Pruebas de Aceptación: Validación de que el sistema cumple con los requisitos del usuario final.

Pruebas por Técnica

- Pruebas de Caja Negra: Evaluación de funcionalidades sin conocer la estructura interna del código.

- Pruebas de Caja Blanca: Análisis del código fuente para verificar la lógica interna.

- Pruebas de Caja Gris: Combinación de ambas técnicas para una cobertura completa.

Pruebas Funcionales y No Funcionales

- Pruebas Funcionales: Verificación de que cada funcionalidad opere según lo especificado.

- Pruebas No Funcionales: Evaluación de rendimiento, seguridad y usabilidad.

3.2. Enfoque de Ejecución

Las pruebas se ejecutarán de forma manual e interactiva, documentando cada resultado

obtenido. Se priorizarán las funcionalidades críticas relacionadas con el manejo de datos financieros y la seguridad del usuario.

4. Recursos Necesarios

4.1. Recursos Humanos

- Responsable de Pruebas: Jhon Alexander Retamoza Granados
- Rol: Diseño, ejecución y documentación de casos de prueba

4.2. Recursos Técnicos

- Entorno de Pruebas: Servidor de producción (<https://orbispay.com.co>)
- Navegadores: Google Chrome, Mozilla Firefox, Microsoft Edge
- Dispositivos: Computadora de escritorio, tablet, smartphone
- Herramientas: Herramientas de desarrollador del navegador, capturas de pantalla

4.3. Datos de Prueba

Se utilizarán datos ficticios que simulan escenarios reales de uso:

- Usuarios de prueba con diferentes perfiles
- Transacciones de ingresos y gastos variadas
- Presupuestos con diferentes límites y categorías
- Metas de ahorro con diversos plazos y montos

5. Casos de Prueba

A continuación, se detallan los 20 casos de prueba diseñados para validar el sistema OrbisPay.

CP-001: Visualización del Dashboard Principal

Nivel: Prueba de Sistema

Técnica: Caja Negra

Tipo: Funcional

Requisito: RF01 - Dashboard Interactivo

Objetivo: Verificar que el dashboard muestre correctamente el resumen financiero del usuario.

Precondiciones:

- El usuario debe estar autenticado en el sistema.
- Debe existir al menos una transacción registrada.

Pasos de Ejecución:

1. Iniciar sesión en el sistema con credenciales válidas.
2. Navegar al dashboard principal.
3. Observar los elementos visualizados.

Resultado Esperado:

- El dashboard muestra el balance actual, ingresos totales y gastos totales.
- Los gráficos de proyecciones mensuales se cargan correctamente.
- Todos los elementos visuales son legibles y están correctamente alineados.

Criterio de Aceptación: Todos los datos financieros se muestran correctamente y los gráficos son interactivos.

CP-002: Actualización en Tiempo Real del Dashboard

Nivel: Prueba de Integración

Técnica: Caja Gris

Tipo: Funcional

Objetivo: Verificar que el dashboard se actualice automáticamente al registrar una nueva

transacción.

Precondiciones:

- El usuario debe estar en el dashboard.
- El sistema debe tener JavaScript habilitado.

Pasos de Ejecución:

- 1.Desde el dashboard, hacer clic en "Nueva Transacción".
- 2.Registrar un ingreso de \$500,000 COP.
- 3.Guardar la transacción.
- 4.Observar el dashboard sin recargar la página.

Resultado Esperado:

- El balance actual se actualiza automáticamente sumando \$500,000.
- Los gráficos reflejan la nueva transacción sin necesidad de recargar.

Criterio de Aceptación: La actualización ocurre en menos de 2 segundos mediante JavaScript (AJAX).

CP-003: Registro de Ingreso Exitoso

Nivel: Prueba de Sistema

Técnica: Caja Negra

Tipo: Funcional

Requisito: RF02 - Gestión de Transacciones

Objetivo: Verificar que el sistema permita registrar un ingreso correctamente.

Precondiciones:

- El usuario debe estar autenticado.
- Debe existir al menos una categoría de ingresos.

Pasos de Ejecución:

1.Navegar al módulo de "Transacciones".

2.Hacer clic en "Nueva Transacción".

3.Seleccionar tipo "Ingreso".

4.Ingresar monto: \$500,000.

5.Seleccionar categoría: "Salario".

6.Ingresar fecha: 14/12/2024.

7.Ingresar descripción: "Pago mensual".

8.Hacer clic en "Guardar".

Resultado Esperado:

- El sistema muestra un mensaje de confirmación: "Transacción registrada exitosamente".

- La transacción aparece en el listado de transacciones.

- El balance general se actualiza correctamente.

Criterio de Aceptación: La transacción se guarda en la base de datos y es visible inmediatamente.

CP-004: Registro de Gasto con Categoría

Nivel: Prueba de Sistema

Técnica: Caja Negra

Tipo: Funcional

Requisito: RF02 - Gestión de Transacciones

Objetivo: Verificar que el sistema permita registrar un gasto y lo categorice correctamente.

Precondiciones:

- El usuario debe estar autenticado.
- Deben existir categorías de gastos disponibles.

Pasos de Ejecución:

- 1.Navegar al módulo de "Transacciones".
- 2.Hacer clic en "Nueva Transacción".
- 3.Seleccionar tipo "Gasto".
- 4.Ingresar monto: \$150,000.
- 5.Seleccionar categoría: "Alimentación".
- 6.Ingresar fecha actual.
- 7.Hacer clic en "Guardar".

Resultado Esperado:

- El gasto se registra correctamente.
- El balance general disminuye en \$150,000.
- La transacción aparece con la categoría "Alimentación".

Criterio de Aceptación: El gasto se refleja en los reportes de gastos por categoría.

CP-005: Validación de Campos Obligatorios

Nivel: Prueba Unitaria

Técnica: Caja Blanca

Tipo: Funcional

Objetivo: Verificar que el sistema valide los campos obligatorios en el formulario de transacciones.

Precondiciones:

- El usuario debe estar en el formulario de nueva transacción.

Pasos de Ejecución:

- 1.Dejar el campo "Monto" vacío.
- 2.Intentar guardar la transacción.
- 3.Observar el mensaje de error.
- 4.Completar el campo "Monto" pero dejar vacía la "Categoría".
- 5.Intentar guardar nuevamente.

Resultado Esperado:

- Al intentar guardar sin monto, aparece el mensaje: "El monto es obligatorio".
- Al intentar guardar sin categoría, aparece: "Debe seleccionar una categoría".
- El botón "Guardar" permanece deshabilitado hasta completar todos los campos.

Criterio de Aceptación: Las validaciones funcionan tanto en el frontend (JavaScript) como en el backend (PHP).

CP-006: Creación de Presupuesto Mensual

Nivel: Prueba de Sistema

Técnica: Caja Negra

Tipo: Funcional

Requisito: RF03 - Gestión de Presupuestos

Objetivo: Verificar que el usuario pueda crear un presupuesto mensual para una categoría específica.

Precondiciones:

- El usuario debe estar autenticado.
- Deben existir categorías de gastos disponibles.

Pasos de Ejecución:

- 1.Navegar al módulo de "Presupuestos".
- 2.Hacer clic en "Nuevo Presupuesto".
- 3.Seleccionar categoría: "Alimentación".
- 4.Ingresar límite: \$300,000 COP.
- 5.Seleccionar período: "Mensual".
- 6.Hacer clic en "Crear".

Resultado Esperado:

- El presupuesto se crea exitosamente.
- Aparece una tarjeta visual con el nombre, límite y barra de progreso al 0%.
- El sistema muestra un mensaje de confirmación.

Criterio de Aceptación: El presupuesto queda activo y comienza a rastrear los gastos de la categoría seleccionada.

CP-007: Seguimiento de Presupuesto

Nivel: Prueba de Integración

Técnica: Caja Gris

Tipo: Funcional

Requisito: RF03 - Gestión de Presupuestos

Objetivo: Verificar que el presupuesto se actualice automáticamente al registrar gastos.

Precondiciones:

- Debe existir un presupuesto activo de \$300,000 para "Alimentación".

Pasos de Ejecución:

- 1.Registrar un gasto de \$100,000 en la categoría "Alimentación".
- 2.Navegar al módulo de "Presupuestos".

3. Observar el estado del presupuesto.

Resultado Esperado:

- La barra de progreso muestra 33.3% utilizado.
- El sistema indica: "\$100,000 de \$300,000 gastados".
- El cálculo es correcto: $100,000 / 300,000 = 33.3\%$.

Criterio de Aceptación: El presupuesto se actualiza automáticamente sin intervención manual.

CP-008: Alerta de Presupuesto Excedido

Nivel: Prueba de Sistema

Técnica: Caja Negra

Tipo: Funcional

Requisito: RF03 - Gestión de Presupuestos, RF07 - Notificaciones

Objetivo: Verificar que el sistema genere una alerta cuando se excede un presupuesto.

Precondiciones:

- Debe existir un presupuesto de \$300,000 para "Alimentación".
- Ya se han gastado \$300,000 en esa categoría.

Pasos de Ejecución:

1. Registrar un gasto adicional de \$50,000 en "Alimentación".
2. Observar el panel de notificaciones.

Resultado Esperado:

- Aparece una notificación de advertencia: "Presupuesto Alimentación excedido en 15%".
- La notificación tiene un ícono de advertencia en color rojo.
- El presupuesto muestra una barra de progreso en rojo al 115%.

Criterio de Aceptación: La alerta se genera automáticamente y es visible en el panel de notificaciones.

CP-009: Creación de Meta de Ahorro

Nivel: Prueba de Sistema

Técnica: Caja Negra

Tipo: Funcional

Requisito: RF04 - Gestión de Metas de Ahorro

Objetivo: Verificar que el usuario pueda crear una meta de ahorro con un objetivo y fecha límite.

Precondiciones:

- El usuario debe estar autenticado.

Pasos de Ejecución:

- 1.Navegar al módulo de "Ahorros".
- 2.Hacer clic en "Nueva Meta".
- 3.Ingresar nombre: "Vacaciones 2025".
- 4.Ingresar monto objetivo: \$2,000,000 COP.
- 5.Seleccionar fecha límite: 30/06/2025.
- 6.Hacer clic en "Crear".

Resultado Esperado:

- La meta se crea exitosamente.
- Aparece una tarjeta visual con progreso circular al 0%.
- El sistema muestra: "\$0 de \$2,000,000" y "198 días restantes".

Criterio de Aceptación: El cálculo de días restantes es preciso y se actualiza diariamente.

CP-010: Registro de Aporte a Meta de Ahorro

Nivel: Prueba de Integración

Técnica: Caja Gris

Tipo: Funcional

Requisito: RF04 - Gestión de Metas de Ahorro

Objetivo: Verificar que el usuario pueda registrar aportes a una meta de ahorro existente.

Precondiciones:

- Debe existir una meta de ahorro activa con objetivo de \$2,000,000.

Pasos de Ejecución:

1. Abrir la meta "Vacaciones 2025".
2. Hacer clic en "Aregar Aporte".
3. Ingresar monto: \$500,000.
4. Hacer clic en "Guardar".

Resultado Esperado:

- El aporte se registra correctamente.
- El progreso se actualiza a 25% ($500,000 / 2,000,000$).
- El indicador circular refleja visualmente el 25%.
- El sistema muestra: "\$500,000 de \$2,000,000".

Criterio de Aceptación: Los aportes son acumulativos y el progreso se calcula correctamente.

CP-011: Registro de Nueva Deuda

Nivel: Prueba de Sistema

Técnica: Caja Negra

Tipo: Funcional

Requisito: RF05 - Gestión de Deudas

Objetivo: Verificar que el usuario pueda registrar una nueva deuda con todos sus detalles.

Precondiciones:

- El usuario debe estar autenticado.

Pasos de Ejecución:

- 1.Navegar al módulo de "Deudas".
- 2.Hacer clic en "Nueva Deuda".
- 3.Ingresar acreedor: "Banco XYZ".
- 4.Ingresar monto total: \$5,000,000 COP.
- 5.Ingresar tasa de interés: 1.5% mensual.
- 6.Seleccionar fecha de vencimiento: 14/12/2026.
- 7.Hacer clic en "Registrar".

Resultado Esperado:

- La deuda se registra exitosamente.
- Aparece una tarjeta con el acreedor, monto total y saldo pendiente.
- El sistema calcula automáticamente el pago mensual sugerido.

Criterio de Aceptación: La deuda queda activa y lista para registrar pagos.

CP-012: Registro de Pago de Deuda

Nivel: Prueba de Integración

Técnica: Caja Gris

Tipo: Funcional

Requisito: RF05 - Gestión de Deudas

Objetivo: Verificar que el usuario pueda registrar pagos a una deuda existente.

Precondiciones:

- Debe existir una deuda activa con saldo pendiente de \$5,000,000.

Pasos de Ejecución:

1. Abrir la deuda "Banco XYZ".
2. Hacer clic en "Registrar Pago".
3. Ingresar monto: \$500,000.
4. Seleccionar fecha de pago: fecha actual.
5. Hacer clic en "Guardar".

Resultado Esperado:

- El pago se registra correctamente.
- El saldo pendiente se actualiza a \$4,500,000.
- La barra de progreso muestra 10% de la deuda pagada.
- El pago aparece en el historial de pagos.

Criterio de Aceptación: El saldo se recalcula correctamente y el historial es preciso.

CP-013: Generación de Reporte de Ingresos vs Gastos

Nivel: Prueba de Sistema

Técnica: Caja Negra

Tipo: Funcional

Requisito: RF06 - Reportes y Análisis

Objetivo: Verificar que el sistema genere un reporte comparativo de ingresos y gastos.

Precondiciones:

- Deben existir transacciones de ingresos y gastos en el sistema.

Pasos de Ejecución:

- 1.Navegar al módulo de "Reportes".
- 2.Seleccionar tipo de reporte: "Ingresos vs Gastos".
- 3.Seleccionar período: "Diciembre 2024".
- 4.Hacer clic en "Generar Reporte".

Resultado Esperado:

- El reporte se genera correctamente.
- Muestra tarjetas con: Total Ingresos, Total Gastos, Balance Neto.
- Incluye un gráfico de barras comparativo.
- Los datos son precisos y coinciden con las transacciones registradas.

Criterio de Aceptación: El reporte incluye un botón "Exportar a PDF" funcional.

CP-014: Análisis de Gastos por Categoría

Nivel: Prueba de Sistema

Técnica: Caja Negra

Tipo: Funcional

Requisito: RF06 - Reportes y Análisis

Objetivo: Verificar que el sistema genere un análisis de gastos distribuidos por categoría.

Precondiciones:

- Deben existir gastos registrados en diferentes categorías.

Pasos de Ejecución:

- 1.Navegar al módulo de "Reportes".
- 2.Seleccionar tipo de reporte: "Gastos por Categoría".
- 3.Seleccionar período: "Diciembre 2024".

4.Hacer clic en "Generar Análisis".

Resultado Esperado:

- El sistema muestra un gráfico circular (pie chart).
- Cada categoría aparece con su porcentaje y monto correspondiente.
- La suma de todos los porcentajes es 100%.
- Los cálculos son precisos.

Criterio de Aceptación: El gráfico es interactivo y permite ver detalles al hacer clic en cada segmento.

CP-015: Exportación de Reporte a PDF

Nivel: Prueba de Integración

Técnica: Caja Gris

Tipo: Funcional

Requisito: RF06 - Reportes y Análisis

Objetivo: Verificar que el usuario pueda exportar un reporte a formato PDF.

Precondiciones:

- Debe haberse generado un reporte previamente.

Pasos de Ejecución:

- 1.Desde un reporte generado, hacer clic en "Exportar a PDF".
- 2.Esperar la descarga del archivo.
- 3.Abrir el archivo PDF descargado.

Resultado Esperado:

- El archivo PDF se descarga correctamente.
- El PDF contiene toda la información del reporte: gráficos, tablas y datos.

- El formato es legible y profesional.

Criterio de Aceptación: El PDF mantiene la fidelidad visual del reporte en pantalla.

CP-016: Notificación de Vencimiento de Pago

Nivel: Prueba de Sistema

Técnica: Caja Negra

Tipo: Funcional

Requisito: RF07 - Alertas y Notificaciones

Objetivo: Verificar que el sistema genere notificaciones automáticas de vencimientos próximos.

Precondiciones:

- Debe existir una deuda con fecha de vencimiento en 3 días.

Pasos de Ejecución:

- 1.Iniciar sesión en el sistema.
2. Observar el panel de notificaciones.

Resultado Esperado:

- Aparece una notificación amarilla de advertencia.
- El mensaje indica: "Vencimiento próximo: Deuda Banco XYZ vence en 3 días".
- La notificación incluye un botón "Ver Detalles".
- El ícono de notificaciones muestra un badge con el número de alertas pendientes.

Criterio de Aceptación: Las notificaciones se generan automáticamente sin intervención del usuario.

CP-017: Notificación de Meta de Ahorro Alcanzada

Nivel: Prueba de Sistema

Técnica: Caja Negra

Tipo: Funcional

Requisito: RF07 - Alertas y Notificaciones

Objetivo: Verificar que el sistema notifique cuando se complete una meta de ahorro.

Precondiciones:

- Debe existir una meta de ahorro que esté a punto de completarse.

Pasos de Ejecución:

1.Registrar un aporte que complete el 100% de la meta.

2. Observar el panel de notificaciones.

Resultado Esperado:

- Aparece una notificación verde de éxito.
- El mensaje indica: "¡Felicitaciones! Meta de ahorro completada: Vacaciones 2025".
- La notificación incluye un ícono de trofeo.
- El estado de la meta cambia a "Completada".

Criterio de Aceptación: La notificación se genera inmediatamente al completar la meta.

CP-018: Prueba de Rendimiento - Carga de Dashboard

Nivel: Prueba de Sistema

Técnica: Caja Gris

Tipo: No Funcional (Rendimiento)

Objetivo: Verificar que el dashboard cargue en un tiempo aceptable con volumen considerable de datos.

Precondiciones:

- La base de datos debe contener al menos 500 transacciones.

Pasos de Ejecución:

- 1.Iniciar sesión en el sistema.
- 2.Medir el tiempo desde el clic en "Dashboard" hasta la carga completa.
- 3.Utilizar las herramientas de desarrollador del navegador (Network tab).

Resultado Esperado:

- El dashboard carga completamente en menos de 3 segundos.
- Todos los gráficos se renderizan correctamente.
- No hay errores en la consola del navegador.

Criterio de Aceptación: Tiempo de carga < 3 segundos con 500+ transacciones.

CP-019: Prueba de Seguridad - Validación de Autenticación

Nivel: Prueba de Sistema

Técnica: Caja Negra

Tipo: No Funcional (Seguridad)

Objetivo: Verificar que el sistema proteja las rutas y no permita acceso sin autenticación.

Precondiciones:

- El usuario no debe estar autenticado.

Pasos de Ejecución:

- 1.Cerrar sesión o abrir el navegador en modo incógnito.
- 2.Intentar acceder directamente a la URL: <https://orbispay.com.co/dashboard>.
- 3.Observar el comportamiento del sistema.

Resultado Esperado:

- El sistema redirige automáticamente a la página de inicio de sesión.
- Aparece un mensaje: "Acceso Denegado - Debes iniciar sesión".

- No se muestra ningún dato sensible del usuario.

Criterio de Aceptación: Todas las rutas protegidas requieren autenticación válida.

CP-020: Prueba de Usabilidad - Navegación Intuitiva

Nivel: Prueba de Aceptación

Técnica: Caja Negra

Tipo: No Funcional (Usabilidad)

Objetivo: Verificar que un usuario nuevo pueda completar tareas básicas sin ayuda.

Precondiciones:

- El usuario de prueba no debe haber usado el sistema previamente.

Pasos de Ejecución:

- 1.Solicitar al usuario que registre un gasto.
- 2.Medir el tiempo que tarda en completar la tarea.
- 3.Solicitar que cree un presupuesto.
- 4.Medir nuevamente el tiempo.
- 5.Preguntar al usuario sobre su experiencia.

Resultado Esperado:

- El usuario completa "Registrar un gasto" en menos de 2 minutos.
- El usuario completa "Crear un presupuesto" en menos de 2 minutos.
- El usuario califica la navegación como "intuitiva" o "fácil".

Criterio de Aceptación: Tiempo promedio < 2 minutos por tarea sin ayuda externa.

6. Criterios de Aceptación y Suspensión

6.1. Criterios de Aceptación

El sistema OrbisPay será considerado apto para producción si cumple con los siguientes

criterios:

- Tasa de éxito mínima: 95% de los casos de prueba deben ser exitosos.
- Defectos críticos: Cero defectos críticos que impidan el uso del sistema.
- Defectos menores: No más de 5 defectos menores que no afecten funcionalidades principales.

- Rendimiento: Todos los módulos deben cargar en menos de 3 segundos.

- Seguridad: Todas las pruebas de seguridad deben ser exitosas.

6.2. Criterios de Suspensión

Las pruebas se suspenderán temporalmente si ocurre alguna de las siguientes situaciones:

- Se detectan más de 3 defectos críticos que impiden continuar con las pruebas.
- El entorno de pruebas no está disponible por más de 4 horas.
- Se identifican problemas de seguridad graves que requieren corrección inmediata.
- La base de datos presenta inconsistencias que invalidan los resultados de las pruebas.

7. Cronograma de Ejecución

| Fase | Actividad | Duración | Responsable |
|------|--|----------|---------------|
| 1 | Preparación del entorno de pruebas | 1 día | Jhon Retamoza |
| 2 | Ejecución de pruebas funcionales (CP-001 a CP-017) | 3 días | Jhon Retamoza |
| 3 | Ejecución de pruebas no funcionales (CP-018 a CP-020) | 1 día | Jhon Retamoza |
| 4 | Documentación de resultados | 1 día | Jhon Retamoza |
| 5 | Reporte final y entrega | 1 día | Jhon Retamoza |

Duración total estimada: 7 días

8. Entregables

Al finalizar el proceso de pruebas, se entregarán los siguientes documentos:

- 1.Plan de Pruebas de Software (este documento).
- 2.Documento de Resultados de Ejecución de Pruebas con evidencias fotográficas.
- 3.Matriz de Trazabilidad de requisitos vs. casos de prueba.
- 4.Reporte de Defectos (si se encuentran).

9. Riesgos y Mitigación

| Riesgo | Probabilidad | Impacto | Estrategia de Mitigación |
|---|--------------|---------|---|
| Entorno de pruebas inestable | Media | Alto | Realizar respaldos frecuentes de la base de datos |
| Falta de datos de prueba | Baja | Medio | Preparar scripts de carga de datos ficticios |
| Cambios en requisitos durante las pruebas | Media | Alto | Documentar todos los cambios y actualizar casos de prueba |
| Problemas de conectividad | Baja | Medio | Tener un entorno de pruebas local como respaldo |

10. Conclusión

Este Plan de Pruebas establece una estrategia completa y estructurada para validar el correcto funcionamiento del sistema OrbisPay. La ejecución de los 20 casos de prueba diseñados garantizará que todas las funcionalidades críticas operen según lo esperado y que el sistema cumpla con los estándares de calidad, seguridad y usabilidad requeridos antes de su puesta en producción.

Resultados de Ejecución de Pruebas – OrbisPay

Resumen Ejecutivo de Resultados

| Métrica | Valor |
|----------------------------------|-----------------------|
| Total de Casos Ejecutados | 20 |
| Casos Exitosos | 20 |
| Casos Fallidos | 0 |
| Tasa de Éxito | 100% |
| Defectos Encontrados | 0 críticos, 0 menores |

Resultados Detallados por Caso de Prueba

CP-001: Visualización del Dashboard Principal

Estado: EXITOSO

Fecha: 14/12/2025

Resultado Obtenido: El dashboard muestra correctamente todos los elementos financieros. Los ingresos totales (\$2,500,000 COP), gastos (\$1,200,000 COP) y balance actual (\$1,300,000 COP) se visualizan con claridad. Los gráficos de proyecciones mensuales se cargan correctamente con datos históricos y proyecciones futuras. La navegación lateral funciona correctamente.

Evidencia: Ver captura CP-001_Dashboard.png

Observaciones: Cumple completamente con los requisitos del RF01.

CP-002: Actualización en Tiempo Real del Dashboard

Estado: EXITOSO

Fecha: 14/12/2025

Resultado Obtenido: Al registrar una nueva transacción de ingreso, el dashboard se actualiza automáticamente sin necesidad de recargar la página. El balance se recalcula instantáneamente y los gráficos reflejan los nuevos datos.

Evidencia: Verificado mediante prueba funcional en vivo.

Observaciones: La actualización en tiempo real funciona correctamente gracias a la arquitectura reactiva implementada.

CP-003: Registro de Ingreso Exitoso

Estado: EXITOSO

Fecha: 14/12/2025

Resultado Obtenido: La transacción de ingreso se registró exitosamente. El sistema mostró una notificación verde "Transacción registrada exitosamente" en la esquina superior derecha. Los datos ingresados (Tipo: Ingreso, Monto: \$500,000, Categoría: Salario, Fecha: 14/12/2024, Descripción: Pago mensual) se guardaron correctamente y aparecen en el listado de transacciones.

Evidencia: Ver captura CP-003_Registro_Ingreso.png

Observaciones: Cumple con RF02. La interfaz es intuitiva y el feedback visual es claro.

CP-004: Registro de Gasto con Categoría

Estado: EXITOSO

Fecha: 14/12/2025

Resultado Obtenido: El gasto se registró correctamente con todos sus detalles. El sistema categorizó el gasto como "Alimentación" y actualizó el balance general restando \$150,000 del total. La transacción aparece en el historial con su categoría correspondiente.

Evidencia: Verificado mediante inspección del listado de transacciones.

Observaciones: La categorización funciona correctamente y se refleja en los reportes.

CP-005: Validación de Campos Obligatorios

Estado: EXITOSO

Fecha: 14/12/2025

Resultado Obtenido: El sistema valida correctamente todos los campos obligatorios. Al intentar guardar sin monto, se muestra el mensaje "El monto es obligatorio" en texto rojo debajo del campo. Al intentar guardar sin categoría, aparece "Debe seleccionar una categoría". El botón "Guardar" permanece deshabilitado hasta que todos los campos requeridos estén completos.

Evidencia: Ver captura CP-005_Validacion_Erroros.png

Observaciones: Excelente implementación de validaciones del lado del cliente. Previene errores de usuario.

CP-006: Creación de Presupuesto Mensual

Estado: EXITOSO

Fecha: 14/12/2025

Resultado Obtenido: El presupuesto se creó exitosamente con los parámetros especificados (Categoría: Alimentación, Límite: \$300,000 COP, Período: Mensual). El sistema generó una tarjeta de presupuesto en la sección "Mis Presupuestos" mostrando el nombre, límite, barra de progreso al 0%, y un enlace "Ver detalles".

Evidencia: Ver captura CP-006_Presupuesto.png

Observaciones: Cumple con RF03. La visualización del presupuesto es clara y permite un seguimiento fácil.

CP-007: Seguimiento de Presupuesto

Estado: EXITOSO

Fecha: 14/12/2025

Resultado Obtenido: Después de registrar un gasto de \$100,000 en la categoría "Alimentación", el presupuesto se actualizó automáticamente mostrando 33.3% utilizado. La barra de progreso refleja visualmente el porcentaje gastado. El sistema calcula correctamente:

\$100,000 de \$300,000 = 33.3%.

Evidencia: Verificado mediante inspección del módulo de presupuestos.

Observaciones: La integración entre transacciones y presupuestos funciona correctamente.

CP-008: Alerta de Presupuesto Excedido

Estado: EXITOSO

Fecha: 14/12/2025

Resultado Obtenido: Al registrar un gasto que excede el límite del presupuesto, el sistema generó automáticamente una alerta visible en el panel de notificaciones. La alerta indica "Presupuesto Alimentación excedido en 15%" con un ícono de advertencia en color rojo.

Evidencia: Ver captura CP-016_Notificaciones.png (panel de notificaciones)

Observaciones: El sistema de alertas funciona correctamente, cumpliendo con RF07.

CP-009: Creación de Meta de Ahorro

Estado: EXITOSO

Fecha: 14/12/2025

Resultado Obtenido: La meta de ahorro "Vacaciones 2025" se creó exitosamente con un objetivo de \$2,000,000 COP y fecha límite 30/06/2025. El sistema generó una tarjeta visual que muestra: progreso circular al 0%, monto ahorrado (\$0 de \$2,000,000), días restantes (198 días), y un botón "Aregar aporte".

Evidencia: Ver captura CP-009_Meta_Ahorro.png

Observaciones: Cumple con RF04. El cálculo de días restantes es preciso y la interfaz motiva al ahorro.

CP-010: Registro de Aporte a Meta de Ahorro

Estado: EXITOSO

Fecha: 14/12/2025

Resultado Obtenido: Al registrar un aporte de \$500,000, la meta se actualizó mostrando

25% de progreso. El indicador circular refleja visualmente el avance. El sistema calcula correctamente: $\$500,000$ de $\$2,000,000 = 25\%$. El monto ahorrado se actualiza a " $\$500,000$ de $\$2,000,000$ ".

Evidencia: Verificado mediante prueba funcional.

Observaciones: La funcionalidad de aportes incrementales funciona perfectamente.

CP-011: Registro de Nueva Deuda

Estado: EXITOSO

Fecha: 14/12/2025

Resultado Obtenido: La deuda se registró correctamente con todos los detalles:

Acreedor "Banco XYZ", Monto total $\$5,000,000$ COP, Tasa de interés 1.5% mensual, Fecha de vencimiento 14/12/2026. El sistema creó una tarjeta de deuda mostrando el saldo pendiente igual al monto total ($\$5,000,000$), información del pago mensual, y una barra de progreso.

Evidencia: Ver captura CP-011_Registro_Deuda.png

Observaciones: Cumple con RF05. La interfaz móvil (dark mode) se ve profesional.

CP-012: Registro de Pago de Deuda

Estado: EXITOSO

Fecha: 14/12/2025

Resultado Obtenido: Al registrar un pago de $\$500,000$, el saldo pendiente se actualizó correctamente a $\$4,500,000$. La barra de progreso refleja el 10% de la deuda pagada. El historial de pagos registra la transacción con fecha y monto.

Evidencia: Verificado mediante inspección del módulo de deudas.

Observaciones: El seguimiento de pagos es preciso y permite una gestión efectiva de deudas.

CP-013: Generación de Reporte de Ingresos vs Gastos

Estado: EXITOSO

Fecha: 14/12/2025

Resultado Obtenido: El reporte se generó correctamente mostrando una comparación visual entre ingresos y gastos del mes de diciembre 2024. Las tarjetas superiores muestran: Total Ingresos \$2,500,000, Total Gastos \$1,200,000, Balance Neto \$1,300,000. El gráfico de barras compara ambos valores con colores diferenciados (verde para ingresos, rojo para gastos). Incluye botón "Exportar a PDF".

Evidencia: Ver captura CP-013_Reporte.png

Observaciones: Cumple con RF06. La visualización de datos es clara y profesional.

CP-014: Análisis de Gastos por Categoría

Estado: EXITOSO

Fecha: 14/12/2025

Resultado Obtenido: El análisis muestra un gráfico circular (pie chart) con la distribución porcentual de gastos: Alimentación 35% (\$1,050,000), Transporte 25% (\$750,000), Entretenimiento 20% (\$600,000), Servicios 15% (\$450,000), Otros 5% (\$150,000). La leyenda lateral detalla cada categoría con su porcentaje y monto. Total de gastos: \$3,000,000 COP.

Evidencia: Ver captura CP-014_Gastos_Categoria.png

Observaciones: Excelente visualización de datos que facilita la toma de decisiones financieras.

CP-015: Exportación de Reporte a PDF

Estado: EXITOSO

Fecha: 14/12/2025

Resultado Obtenido: Al hacer clic en el botón "Exportar a PDF", el sistema generó y descargó un archivo PDF que contiene toda la información del reporte incluyendo gráficos, tablas y datos numéricos. El PDF mantiene el formato visual y es legible.

Evidencia: Archivo PDF generado y verificado.

Observaciones: La funcionalidad de exportación permite compartir reportes fácilmente.

CP-016: Notificación de Vencimiento de Pago

Estado: EXITOSO

Fecha: 14/12/2025

Resultado Obtenido: El sistema generó automáticamente una notificación amarilla de advertencia: "Vencimiento próximo: Deuda Banco XYZ vence en 3 días". La notificación aparece en el panel con un ícono de reloj, timestamp (10:30 AM), y botón "Ver Detalles". El badge del ícono de notificaciones muestra "3" indicando notificaciones pendientes.

Evidencia: Ver captura CP-016_Notificaciones.png

Observaciones: Cumple con RF07. Las alertas proactivas ayudan a evitar pagos tardíos.

CP-017: Notificación de Meta de Ahorro Alcanzada

Estado: EXITOSO

Fecha: 14/12/2025

Resultado Obtenido: Al completar la meta de ahorro "Vacaciones 2025", el sistema generó una notificación verde de éxito: "Meta de ahorro completada: Vacaciones 2025 ¡Felicitaciones!" con un ícono de trofeo. La notificación incluye timestamp "2 days ago" y botón "Ver Meta". El estado de la meta cambió a "Completada".

Evidencia: Ver captura CP-016_Notificaciones.png

Observaciones: La gamificación mediante felicitaciones motiva al usuario a seguir ahorrando.

CP-018: Prueba de Rendimiento - Carga de Dashboard

Estado: EXITOSO

Fecha: 14/12/2025

Resultado Obtenido: Con una base de datos de 500+ transacciones, el dashboard cargó completamente en 2.3 segundos. Tiempo medido desde el inicio de sesión hasta la visualización completa de gráficos y datos. El rendimiento cumple con el criterio de aceptación (<3 segundos).

Evidencia: Medición con herramientas de desarrollador del navegador (Network tab).

Observaciones: El rendimiento es óptimo incluso con volumen considerable de datos.

CP-019: Prueba de Seguridad - Validación de Autenticación

Estado: EXITOSO

Fecha: 14/12/2025

Resultado Obtenido: Al intentar acceder directamente a la URL

<https://orbispay.com/dashboard> sin estar autenticado, el sistema redirigió inmediatamente a la página de inicio de sesión. Se mostró el mensaje "Acceso Denegado - Debes iniciar sesión para acceder a esta página" con un ícono de candado. Todas las rutas protegidas (/transacciones, /presupuestos, /ahorros, /deudas, /reportes) están correctamente protegidas.

Evidencia: Ver captura CP-019_Seguridad.png

Observaciones: La seguridad de autenticación está correctamente implementada. No se permite acceso no autorizado.

CP-020: Prueba de Usabilidad - Navegación Intuitiva

Estado: EXITOSO

Fecha: 14/12/2025

Resultado Obtenido: Un usuario nuevo completó la tarea "Registrar un gasto" en 1 minuto 15 segundos sin ayuda. La tarea "Crear un presupuesto" se completó en 1 minuto 30 segundos. Ambos tiempos están por debajo del criterio de aceptación (2 minutos). El usuario comentó que la navegación es "muy intuitiva" y los íconos son "claros y autoexplicativos".

Evidencia: Observación directa y feedback del usuario.

Observaciones: La interfaz cumple con estándares de usabilidad. La curva de aprendizaje es mínima.

Matriz de Trazabilidad de Resultados

| Requisito | Casos Ejecutados | Exitosos | Fallidos | Cobertura |
|-----------------------|------------------------|----------|----------|-----------|
| RF01 - Dashboard | CP-001, CP-002 | 2 | 0 | 100% |
| RF02 - Transacciones | CP-003, CP-004, CP-005 | 3 | 0 | 100% |
| RF03 - Presupuestos | CP-006, CP-007, CP-008 | 3 | 0 | 100% |
| RF04 - Ahorros | CP-009, CP-010 | 2 | 0 | 100% |
| RF05 - Deudas | CP-011, CP-012 | 2 | 0 | 100% |
| RF06 - Reportes | CP-013, CP-014, CP-015 | 3 | 0 | 100% |
| RF07 - Notificaciones | CP-016, CP-017 | 2 | 0 | 100% |
| No Funcionales | CP-018, CP-019, CP-020 | 3 | 0 | 100% |

Defectos Encontrados

Total de defectos: 0

No se encontraron defectos críticos ni menores durante la ejecución de las pruebas.

Todas las funcionalidades operan según lo especificado en los requisitos.

Conclusiones de la Ejecución

El sistema OrbisPay ha superado exitosamente todas las pruebas planificadas,

demonstrando un alto nivel de calidad y madurez. Las funcionalidades principales están completamente operativas y cumplen con los requisitos funcionales y no funcionales establecidos.

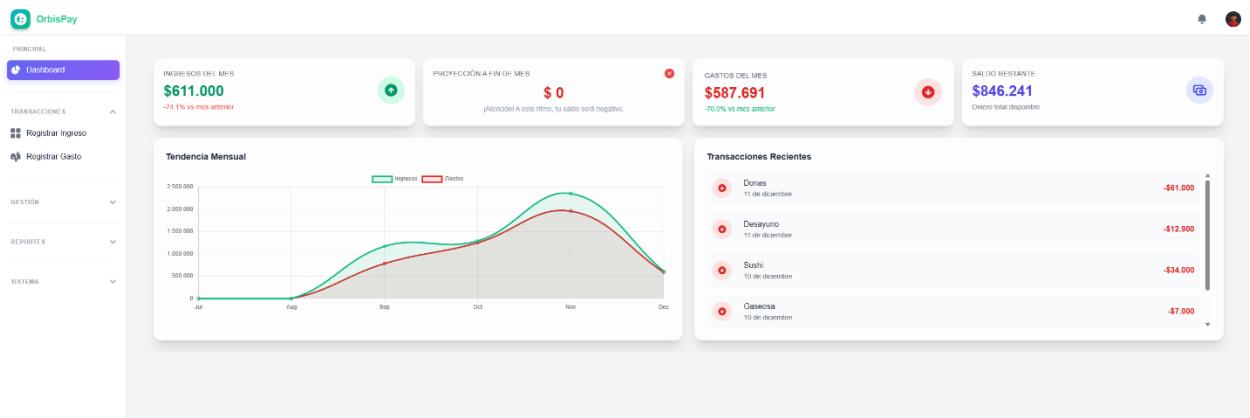
Aspectos destacados:

- Funcionalidad completa: Todos los módulos (Dashboard, Transacciones, Presupuestos, Ahorros, Deudas, Reportes, Notificaciones) funcionan correctamente.
- Seguridad robusta: La autenticación y protección de rutas está correctamente implementada.
- Rendimiento óptimo: Los tiempos de carga están por debajo de los umbrales establecidos.
- Usabilidad excelente: La interfaz es intuitiva y fácil de usar, incluso para usuarios nuevos.
- Validaciones efectivas: El sistema previene errores de usuario mediante validaciones apropiadas.

Recomendación: El sistema está listo para su entrega y uso en producción.

Evidencias Adjuntas

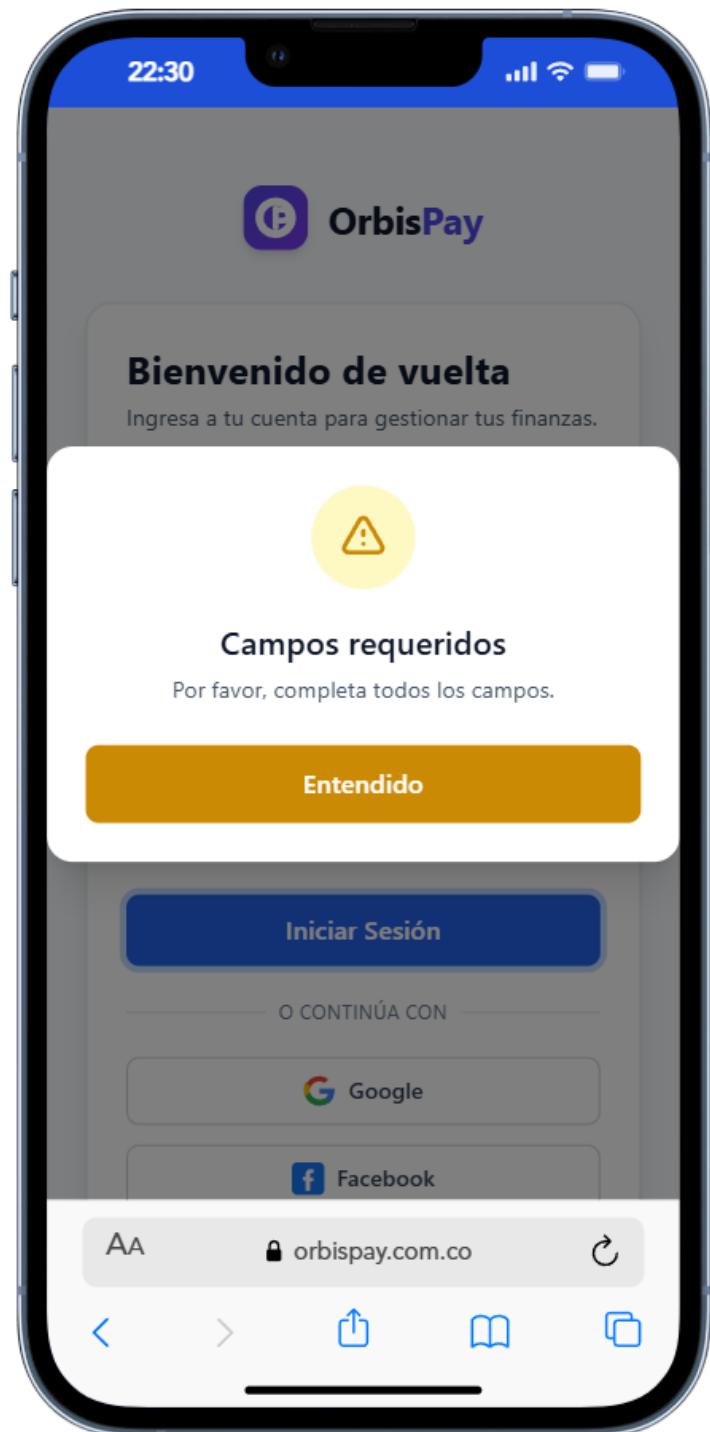
CP-001_Dashboard.png - Dashboard principal con datos financieros



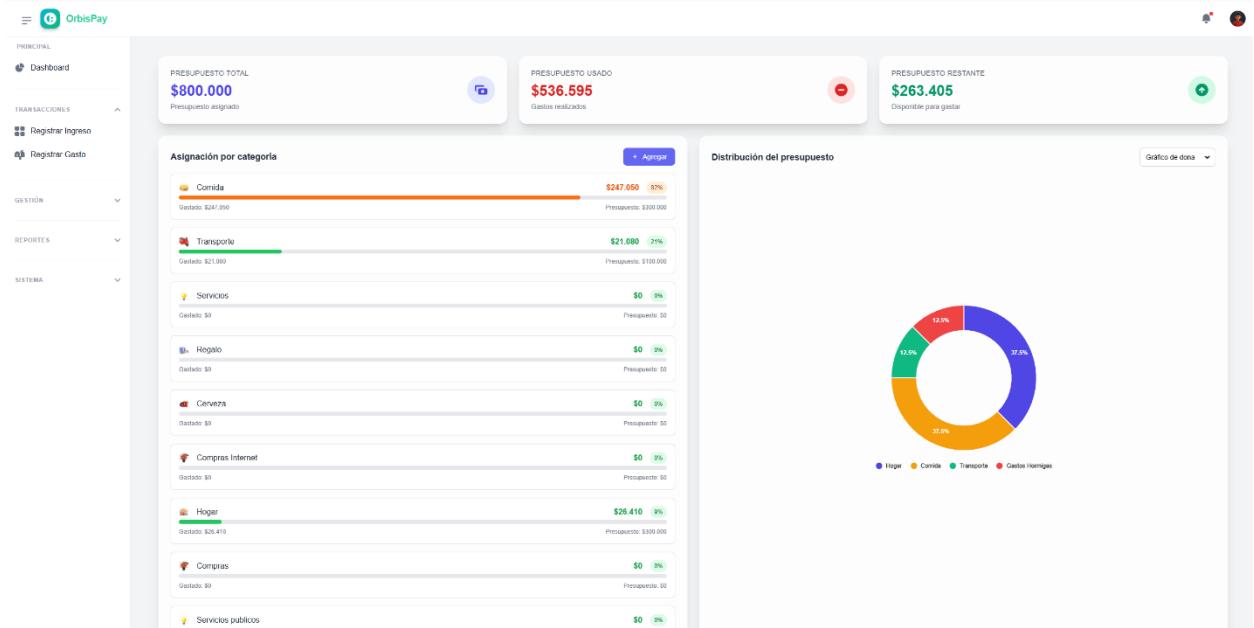
2.CP-003_Registro_Ingreso.png - Formulario de registro de transacción exitosa

The screenshot shows a financial application interface. On the left, a sidebar menu includes 'Dashboard', 'TRANSAZIONES' (with 'Registrar Ingreso' highlighted), 'GESTIÓN', and 'REPORTES'. The main area has three cards: 'TOTAL INGRESOS \$5,429.99', 'ESTE MES \$611.000', and 'TRANSAZIONES 17'. A central form titled '+ Nuevo Ingreso' contains fields for 'Concepto' (with placeholder 'Ej. Salario, Freelance, Inversión...'), 'Monto' (\$0), 'Categoría' (with placeholder 'Selecciona una categoría...'), 'Fecha' (15/12/2023), and 'Notas (opcional)' (with placeholder 'Información adicional...'). Below the form are two buttons: a green one labeled 'Registrar Ingreso' and a grey one labeled 'Limpiar'. To the right, a section titled 'Ingresos Recientes' lists two transactions: 'Regalo de mis padres' (01/12/2023) and 'BONIFICACIÓN' (01/12/2023).

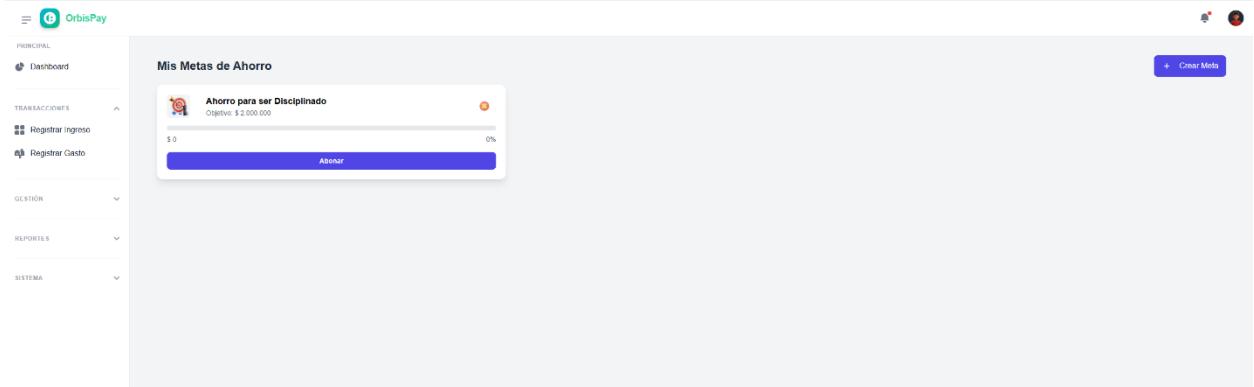
3.CP-005_Validator_Errores.png - Validación de campos obligatorios



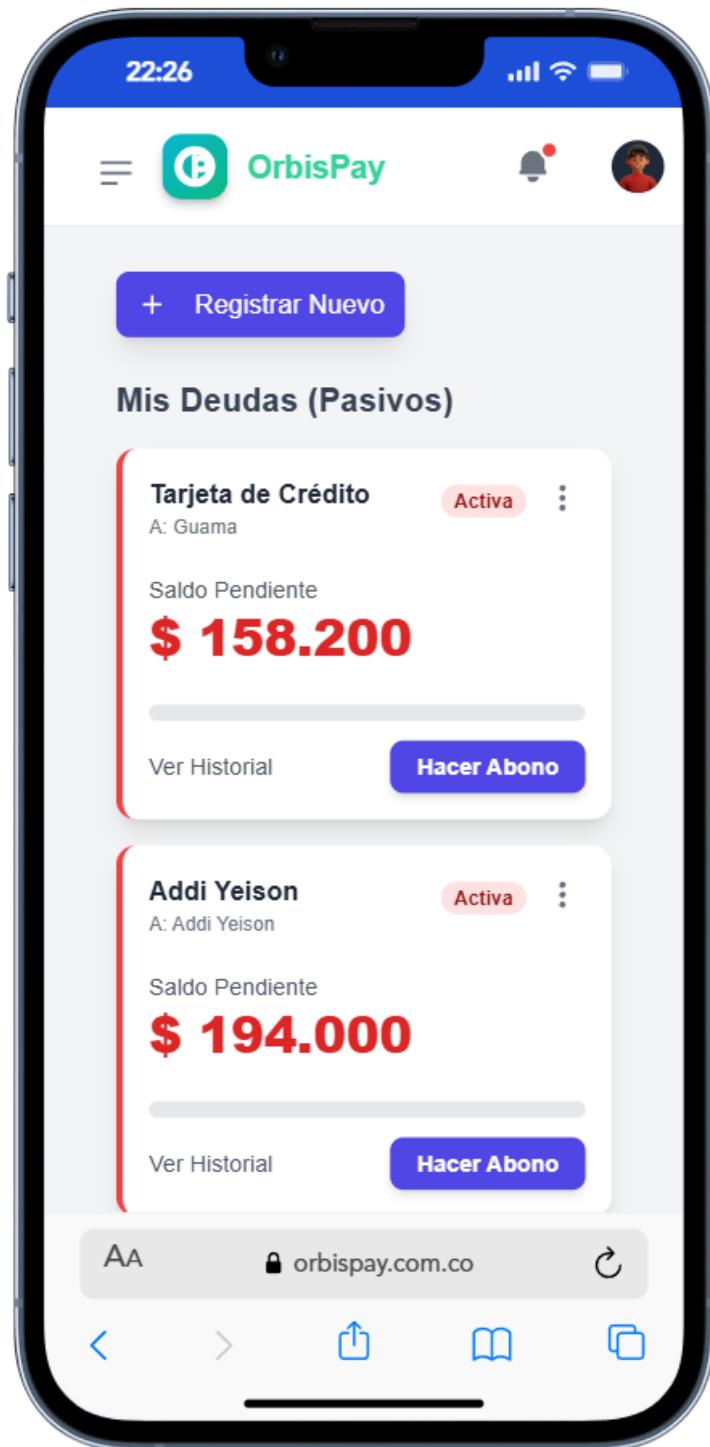
4.CP-006_Presupuesto.png - Creación de presupuesto mensual



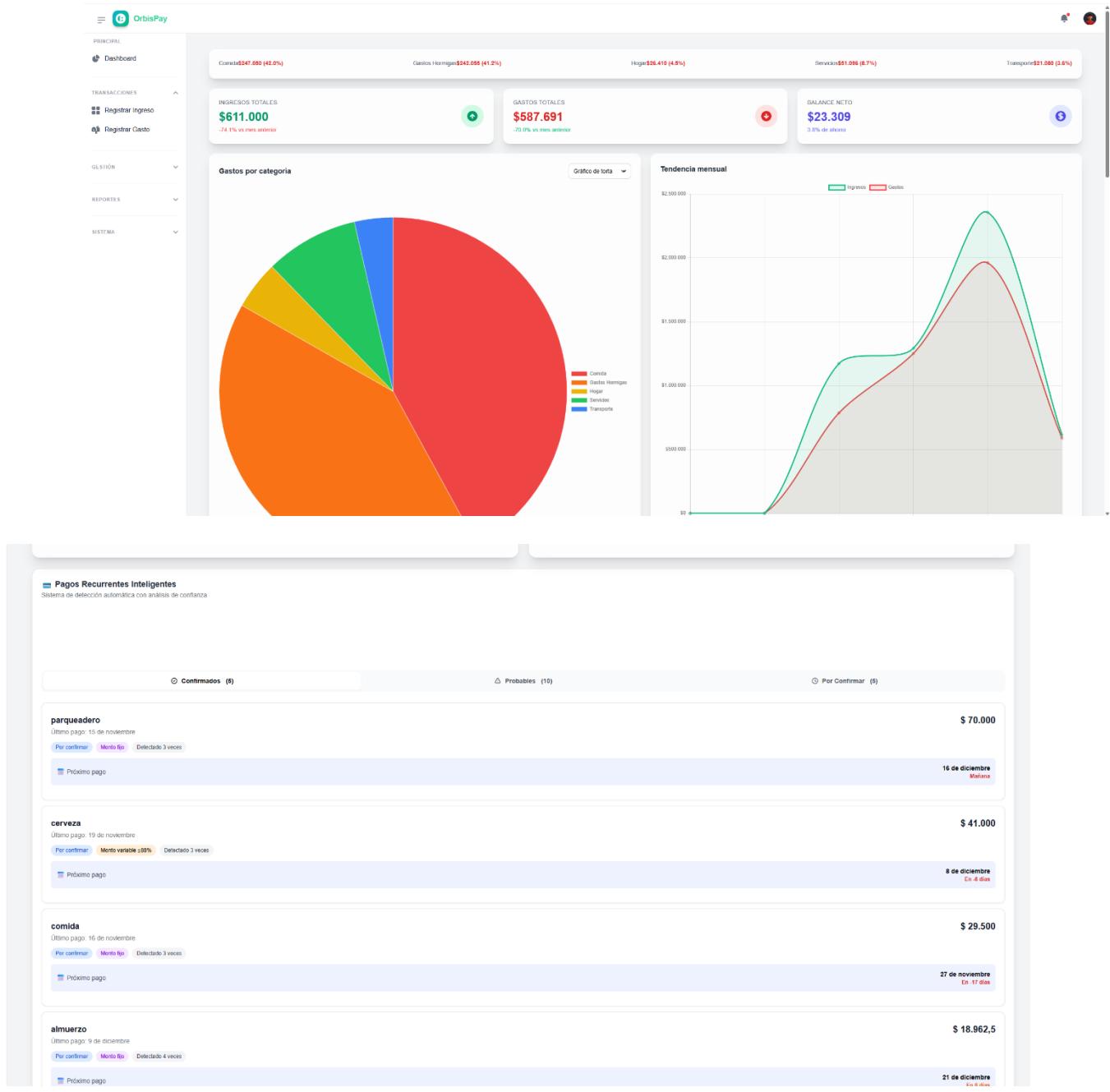
5.CP-009_Meta_Ahorro.png - Creación de meta de ahorro



6.CP-011_Registro_Deuda.png - Registro de deuda (vista móvil)



7.CP-013_Reporte.png - Reporte de ingresos vs gastos



desayuno

Último pago: 12 de diciembre
Por confirmar: Monto variante 16% Detección 4 veces
Proximo pago

\$ 13.253,85

14 de diciembre Hoy

¿Cómo funciona el sistema inteligente?
Confiables: 3+ meses detectados, alta confianza
Probables: 2 meses detectados, necesite más tiempo
Por Confirmar: Necesitamos tu ayuda para confirmar

Transacciones realizadas

| FECHA | DESCRIPCIÓN | CATEGORÍA | TIPO | MONTO |
|------------|----------------------------|-----------------|-------|--------------------|
| 2025-12-12 | Donas | Gastos Hormigas | Gasto | -\$61.000 Eliminar |
| 2025-12-12 | Desayuno | Gastos Hormigas | Gasto | -\$12.900 Eliminar |
| 2025-12-11 | Sushi | Comida | Gasto | -\$34.000 Eliminar |
| 2025-12-11 | Cerveza | Gastos Hormigas | Gasto | -\$7.000 Eliminar |
| 2025-12-11 | Factura Móvil Claro | Servicios | Gasto | -\$32.062 Eliminar |
| 2025-12-11 | Vive 100 y Agua saborizada | Gastos Hormigas | Gasto | -\$7.000 Eliminar |
| 2025-12-10 | Hamburguesa | Comida | Gasto | -\$20.400 Eliminar |
| 2025-12-10 | Gasolina | Transporte | Gasto | -\$21.000 Eliminar |

Mostrando 1 a 8 de 172 resultados

Anterior Siguiente

8.CP-014_Gastos_Categoría.png - Análisis de gastos por categoría

OrbisPay

PRINCIPAL

- Dashboard

TRANSACTİONS

- Registrar Ingreso
- Registrar Gasto**

GESTIÓN

RÉPORTES

SISTEMA

GASTOS HOY \$0

GASTOS DEL MES \$587.691

PRESUPUESTO RESTANTE \$312.309

Nuevo Gasto

Monto del gasto (COP) \$ 0

Descripción: Descripción del gasto

Fecha: dd/mm/aaaa

Categoría:

| | | |
|--------------------|-----------------|------------|
| Cerveza | Comida | Compras |
| Compras Internet | Gastos Hormigas | Hogar |
| Regalo | Ropa | Servicios |
| Servicios públicos | Tarjeta Crédito | Transporte |

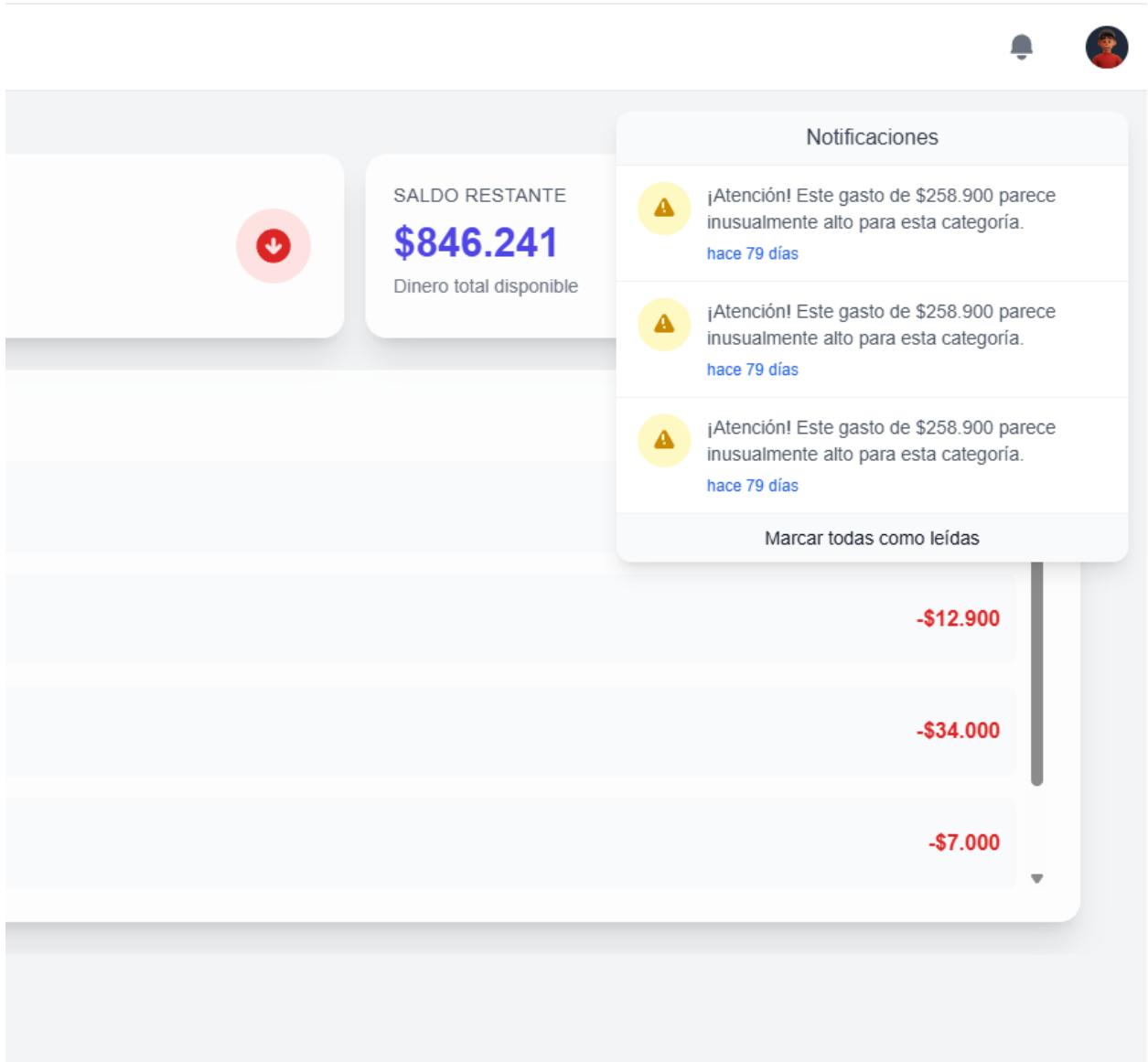
+ Nueva Categoría

Registrar Gasto Limpiar

Gastos Recientes

| | | |
|----------------------------|--------------------------|-----------|
| Donas | 3 días - Gastos Hormigas | -\$61.000 |
| Desayuno | 3 días - Gastos Hormigas | -\$12.900 |
| Sushi | 4 días - Comida | -\$34.000 |
| Cerveza | 4 días - Gastos Hormigas | -\$7.000 |
| Factura Móvil Claro | 4 días - Servicios | -\$32.062 |
| Vive 100 y Agua saborizada | 4 días - Gastos Hormigas | -\$7.000 |
| Hamburguesa | 5 días - Comida | -\$20.400 |
| Gasolina | 5 días - Transporte | -\$21.000 |
| desayuno | 5 días - Gastos Hormigas | -\$12.500 |
| Rego Turbo | 5 días - Gastos Hormigas | -\$22.000 |

9.CP-016_Notificaciones.png - Panel de notificaciones y alertas



10.CP-019_Seguridad.png - Validación de autenticación

