"PLATAFORMA DE GESTIÓN INTEGRAL PARA EL CONTROL DE CLIENTES, SERVICIOS Y FINANZAS EN UNA ELECTROMECÁNICA"

Prueba de Caja Negra

Integrantes:
Benavides Xavier, Yacelga Iván, Suasnavas Gabriel, Jami Klever

Fecha: 2025-02-17

Contenido

Tabla 1: Inicio de Sesión 4

Tabla 1: Requisito Funcional Nº1 Registrar Cliente....5

Codigo 6

PASS....6

NO PASS 7

Tabla 2: Requisito Funcional Nº 2 Registrar Transacción....7

CODIGO 8

PASS....9

NO PASS 9

NO PASS 9

Tabla 3: Requisito Funcional Nº 3 Generador de reportes 10

CODIGO 11

PASS....11

Tabla 4: Requisito Funcional Nº 4 Consultar Historial 12

CODIGO 12

PASS....13

Tabla 5: Requisito Funcional Nº 5 Gestión de Permisos 13

CODIGO 14

PASS....14

NO PASS 15

Historia de Revisión

Fecha	Versión	Descripción	Autores
23/ENERO/2025	1	Version 1.0	Benavides Xavier Yacelga Iván Suasnavas Gabriel Jami Klever
28/ENERO/2025	2	Version 2.0	Benavides Xavier Yacelga Iván Suasnavas Gabriel Jami Klever
11/FEBRERO/2025	2	Version 3.0	Benavides Xavier Yacelga Iván Suasnavas Gabriel Jami Klever

Tabla 1: Inicio de Sesión

Variable	*Clase de Equivalencia*	*Estado*	*Representante*
usuario	*EC1: usuario == "admin"*	*Válido*	*"admin"*
	EC2: usuario != "admin"	*No válido*	*"user123"*
password	*EC1: password == "admin123"*	*Válido*	*"admin123"*
	EC2: password != "admin123"	*No válido*	*"pass456"*

CODIGO

```c
int autenticarUsuario() {
    char usuario[20], password[20];
    printf("Ingrese usuario: ");
    scanf("%s", usuario);
    printf("Ingrese contrasena: ");
    scanf("%s", password);

    if (strcmp(usuario, "admin") == 0 && strcmp(password, "admin123") == 0) {
        printf("\nAutenticacion exitosa.\n");
        return 1;
    }
    return 0;
}
```

PASS

```
Ingrese usuario: admin
Ingrese contrasena: admin123

Autenticacion exitosa.

MENU PRINCIPAL
1. Registrar cliente
2. Registrar transaccion financiera
3. Generar reportes
4. Consultar historial de transacciones
5. Gestionar permisos
6. Salir
Seleccione una opcion:
```

NO PASS

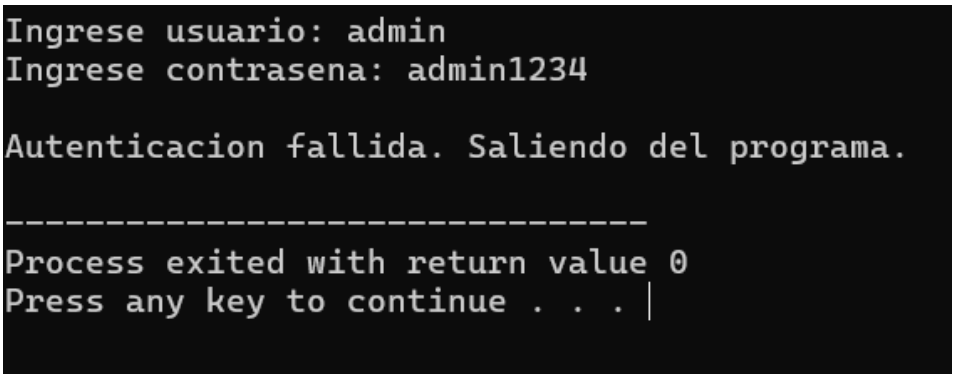

Tabla 1: Requisito Funcional Nº1 Registrar Cliente

Historia de usuario	
Número: RQ01	**Usuario**: Erick Jimenez
Nombre de la Historia: Registro de clientes	
Prioridad: Alta	
Programador responsable: Alexander Benavides	
Descripción: Crear un formulario en el sistema donde se puedan ingresar los datos básicos de los clientes (nombre, teléfono, etc.).	
Validación: Verificar que los datos del cliente se guarden correctamente.	

Variable	*Clase de Equivalencia*	*Estado*	*Representante*
numClientes	*EC1: numClientes < 100*	*Válido*	*50*
	EC2: numClientes >= 100	*No válido*	*100*
nombre	*EC1: strlen(nombre) > 0*	*Válido*	*"JuanPerez"*

	EC2: strlen(nombre) == 0	***No válido***	***""***
telefono	***EC1: strlen(telefono) > 0***	***Válido***	***"0987654321"***
	EC2: strlen(telefono) == 0	***No válido***	***"Caracteres"***

Codigo

```
void registrarCliente() {
    if (numClientes >= 100) {
        printf("No se pueden registrar mas clientes.\n");
        return;
    }

    Cliente nuevoCliente;
    nuevoCliente.id = numClientes + 1;
    printf("Ingrese nombre del cliente: ");
    scanf("%s", nuevoCliente.nombre);

    int validoTelefono = 0;
    while (!validoTelefono) {
        printf("Ingrese telefono del cliente (solo numeros): ");
        scanf("%s", nuevoCliente.telefono);

        if (strlen(nuevoCliente.telefono) > 0 && strspn(nuevoCliente.telefono, "0123456789") == strlen(nuevoCliente.telefono)) {
            validoTelefono = 1;
        } else {
            printf("Error: El telefono debe contener solo números.\n");
        }
    }

    clientes[numClientes] = nuevoCliente;
    numClientes++;

    printf("Cliente registrado exitosamente con ID %d.\n", nuevoCliente.id);
}
```

PASS

```
MENU PRINCIPAL
1. Registrar cliente
2. Registrar transaccion financiera
3. Generar reportes
4. Consultar historial de transacciones
5. Gestionar permisos
6. Salir
Seleccione una opcion: 1
Ingrese nombre del cliente: Erick
Ingrese telefono del cliente: 0987654321
Cliente registrado exitosamente con ID 1.
```

NO PASS

```
MENU PRINCIPAL
1. Registrar cliente
2. Registrar transaccion financiera
3. Generar reportes
4. Consultar historial de transacciones
5. Gestionar permisos
6. Salir
Seleccione una opcion: 1
Ingrese nombre del cliente: erick
Ingrese telefono del cliente (solo numeros): Llllll
Error: El telefono debe contener solo n·meros.
Ingrese telefono del cliente (solo numeros):
```

Tabla 2: Requisito Funcional Nº 2 Registrar Transacción

Historia de usuario	
Número: RQ02	Usuario: Erick Jimenez
Nombre de la Historia: Gestión de transacciones	
Prioridad: Alta	
Programador responsable: Klever Jami	
Descripción: Implementar un módulo donde se pueda registrar la fecha, monto y descripción de cada transacción financiera realizada por el cliente.	
Validación: Comprobar que las transacciones se guarden correctamente en la base de datos.	

Variable	*Clase de Equivalencia*	*Estado*	*Representante*
numTransacciones	*EC1: numTransacciones < 100*	*Válido*	*50*
	EC2: numTransacciones >= 100	*No válido*	*100*
clienteId	*EC1: clienteId válido*	*Válido*	*1*

	EC2: clienteId inválido	***No válido***	***9999***
monto	***EC1: monto > 0***	***Válido***	***150.50***
	EC2: monto <= 0	***No válido***	***-50.00***

CODIGO

```c
void registrarTransaccion() {
    if (numTransacciones >= 100) {
        printf("No se pueden registrar mas transacciones.\n");
        return;
    }

    Transaccion nuevaTransaccion;
    nuevaTransaccion.id = numTransacciones + 1;
    printf("Ingrese ID del cliente: ");
    scanf("%d", &nuevaTransaccion.clienteId);

    int clienteEncontrado = 0;
    for (int i = 0; i < numClientes; i++) {
        if (clientes[i].id == nuevaTransaccion.clienteId) {
            clienteEncontrado = 1;
            break;
        }
    }

    if (!clienteEncontrado) {
        printf("Cliente no encontrado.\n");
        return;
    }

    printf("Ingrese descripcion de la transaccion: ");
    scanf("%s", nuevaTransaccion.descripcion);
    printf("Ingrese monto de la transaccion: ");
    scanf("%f", &nuevaTransaccion.monto);

    transacciones[numTransacciones] = nuevaTransaccion;
    numTransacciones++;

    printf("Transaccion registrada exitosamente con ID %d.\n", nuevaTransaccion.id);
}
```

PASS

```
MENU PRINCIPAL
1. Registrar cliente
2. Registrar transaccion financiera
3. Generar reportes
4. Consultar historial de transacciones
5. Gestionar permisos
6. Salir
Seleccione una opcion: 2
Ingrese ID del cliente: 1
Ingrese descripcion de la transaccion: 150
Ingrese monto de la transaccion: 150
Transaccion registrada exitosamente con ID 1.
```

NO PASS

```
MENU PRINCIPAL
1. Registrar cliente
2. Registrar transaccion financiera
3. Generar reportes
4. Consultar historial de transacciones
5. Gestionar permisos
6. Salir
Seleccione una opcion: 2
Ingrese ID del cliente: 999
Cliente no encontrado.
```

NO PASS

```
MENU PRINCIPAL
1. Registrar cliente
2. Registrar transaccion financiera
3. Generar reportes
4. Consultar historial de transacciones
5. Gestionar permisos
6. Salir
Seleccione una opcion: 2
Ingrese ID del cliente: 1
Ingrese descripcion de la transaccion: -50
Ingrese monto de la transaccion (no negativo):
```

Tabla 3: Requisito Funcional Nº 3 Generador de reportes

Historia de usuario	
Número: RQ03	**Usuario:** Erick Jimenez
Nombre de la Historia: Generación de reportes	
Prioridad: Alta	
Programador responsable: Klever Jami	
Descripción: Desarrollar una función que genere reportes de ingresos y egresos, exportables a PDF o Excel.	
Validación: Verificar que los reportes se generen correctamente y se exporten.	

Variable	Close de Equivalencia	Estado	Representante
rol/swatch	EC1: rol == administrador	Válido	"admin"
	EC2: rol != administrador	No válido	"cliente"
permisos	EC1: permisos definidos	Válido	Crear, Leer, Modificar, Eliminar
	EC2: permisos no definidos	No válido	NULL
generarReporte	EC1: rol == administrador y permisos incluyen "Leer"	Válido	Función generarReporte() accesible
	EC2: rol != administrador o permisos no incluyen "Leer"	No válido	Función generarReporte() no accesible

CODIGO

```
void generarReporte() {
    printf("\nREPORTE DE INGRESOS Y EGRESOS\n");
    printf("==============================\n");

    float totalIngresos = 0, totalEgresos = 0;
    for (int i = 0; i < numTransacciones; i++) {
        if (transacciones[i].monto > 0) {
            totalIngresos += transacciones[i].monto;
        } else {
            totalEgresos += transacciones[i].monto;
        }
    }

    printf("Total Ingresos: %.2f\n", totalIngresos);
    printf("Total Egresos: %.2f\n", totalEgresos);
}
```

int __cdecl printf (const char * __restrict__ _Format,

PASS

```
MENU PRINCIPAL
1. Registrar cliente
2. Registrar transaccion financiera
3. Generar reportes
4. Consultar historial de transacciones
5. Gestionar permisos
6. Salir
Seleccione una opcion: 3

REPORTE DE INGRESOS Y EGRESOS
==============================
Total Ingresos: 50.00
Total Egresos: 0.00
```

Tabla 4: Requisito Funcional Nº 4 Consultar Historial

Historia de usuario	
Número: RQ04	**Usuario:** Erick Jimenez
Nombre de la Historia: Historial de transacciones	
Prioridad: Alta	
Programador responsable: Klever Jami	
Descripción: Crear una sección en el sistema donde se pueda buscar a un cliente y visualizar su historial de transacciones.	
Validación: Verificar que el historial de transacciones se visualice correctamente.	

Variable	*Clase de Equivalencia*	*Estado*	*Representante*
clienteId	*EC1: clienteId válido*	*Válido*	*2*

CODIGO

```
void consultarHistorial() {
    int clienteId;
    printf("Ingrese ID del cliente para consultar historial: ");
    scanf("%d", &clienteId);

    printf("\nHISTORIAL DE TRANSACCIONES DEL CLIENTE %d\n", clienteId);
    printf("=======================================\n");
    for (int i = 0; i < numTransacciones; i++) {
        if (transacciones[i].clienteId == clienteId) {
            printf("ID Transaccion: %d\n", transacciones[i].id);
            printf("Descripcion: %s\n", transacciones[i].descripcion);
            printf("Monto: %.2f\n", transacciones[i].monto);
            printf("-------------------------\n");
        }
    }
}
```

PASS

```
MENU PRINCIPAL
1. Registrar cliente
2. Registrar transaccion financiera
3. Generar reportes
4. Consultar historial de transacciones
5. Gestionar permisos
6. Salir
Seleccione una opcion: 4
Ingrese ID del cliente para consultar historial: 2

HISTORIAL DE TRANSACCIONES DEL CLIENTE 2
========================================
```

Tabla 5: Requisito Funcional Nº 5 Gestión de Permisos

Historia de usuario	
Número: RQ05	Usuario: Erick Jimenez
Nombre de la Historia: Permisos de usuario	
Prioridad: Alta	
Programador responsable: Klever Jami	
Descripción: Implementar un sistema de roles y permisos donde el administrador puede asignar accesos específicos a los usuarios del sistema.	
Validación: Verificar que solo los usuarios con el rol adecuado puedan acceder a los datos sensibles.	

Variable	*Clase de Equivalencia*	*Estado*	*Representante*
rolUsuario	*EC1: rol == administrador*	*Válido*	*"admin"*

	EC2: rol != administrador	***No válido***	***"cliente"***
permisos	***EC1: permisos definidos***	***Válido***	***Crear, Leer, Modificar, Eliminar***
	EC2: permisos no definidos	***No válido***	***NULL***

CODIGO

```c
void gestionarPermisos() {
    char usuario[20];
    int opcion;

    printf("Ingrese el nombre del usuario para gestionar permisos: ");
    scanf("%s", usuario);

    if (strcmp(usuario, "admin") != 0) {
        printf("Solo el usuario 'admin' puede gestionar permisos.\n");
        return;
    }

    printf("\nGestion de permisos para %s\n", usuario);
    printf("1. Asignar permisos de administrador\n");
    printf("2. Revocar permisos de administrador\n");
    printf("3. Salir\n");
    printf("Seleccione una opcion: ");
    scanf("%d", &opcion);

    if (opcion == 1) {
        printf("Permisos de administrador asignados a %s.\n", usuario);
    } else if (opcion == 2) {
        printf("Permisos de administrador revocados a %s.\n", usuario);
    } else if (opcion == 3) {
        printf("Saliendo de la gestion de permisos...\n");
    } else {
        printf("Opcion invalida. Intente nuevamente.\n");
    }
}
```

PASS

```
MENU PRINCIPAL
1. Registrar cliente
2. Registrar transaccion financiera
3. Generar reportes
4. Consultar historial de transacciones
5. Gestionar permisos
6. Salir
Seleccione una opcion: 5
Ingrese el nombre del usuario para gestionar permisos: admin

Gestion de permisos para admin
1. Asignar permisos de administrador
2. Revocar permisos de administrador
3. Salir
Seleccione una opcion: 1
Permisos de administrador asignados a admin.
```

NO PASS

```
MENU PRINCIPAL
1. Registrar cliente
2. Registrar transaccion financiera
3. Generar reportes
4. Consultar historial de transacciones
5. Gestionar permisos
6. Salir
Seleccione una opcion: 5
Ingrese el nombre del usuario para gestionar permisos: ad
Solo el usuario 'admin' puede gestionar permisos.
```