Centro Universitario de Occidente

Seminario de Sistemas 1

Ing. Pedro Domingo

Segundo Semestre 2024

“Manual Técnico
Portal Crédito”

#3 - 201930699 - Erick Daniel Morales Xicará

MANUAL TÉCNICO

Requisitos del Sistema

El software puede ser utilizado en cualquier sistema operativo, ya que es un proyecto web. Está desarrollado utilizando **TypeScript** tanto en el frontend como en el backend. A continuación, se listan los requisitos principales:

- **Frontend:** Angular 17.x+
- **Backend:** Node.js 16.19.0+ con Express
- **Base de datos:** MariaDB 10.x+
- **Herramientas de desarrollo:** IntelliJ, Visual Studio Code
- **Navegadores soportados:** Google Chrome, Mozilla Firefox, Microsoft Edge (últimas versiones)

Descripción del Proyecto

Este software es un portal financiero especializado en la gestión de cuentas de tarjeta de crédito, permitiendo a los clientes administrar su saldo, consultar movimientos, y gestionar el estado de sus tarjetas de crédito. Las principales características incluyen:

- **Zona de información y registro**: Los usuarios pueden acceder al portal con su cuenta registrada y, si es necesario, utilizar la opción de “recordatorio de PIN” para recibir su clave en el correo asociado. La sección de información también presenta los últimos comentarios de los usuarios sobre los beneficios de sus tarjetas.
- **Comentarios sobre beneficios**: Los usuarios autenticados tienen la opción de dejar comentarios acerca de su experiencia con la tarjeta, los cuales quedan registrados en su perfil.

- **Gestión de saldo**: Los usuarios pueden consultar los movimientos de aumento o reducción de saldo en su cuenta hasta una fecha específica, obteniendo detalles sobre el monto, tipo de movimiento y saldo actual.
- **Vista de administración**: Los administradores tienen acceso a un módulo para gestionar las cuentas de usuarios. Pueden crear, modificar o eliminar cuentas con una sintaxis estandarizada, así como habilitar o deshabilitar tarjetas en caso de extravío, robo o situación de morosidad.
- **Ajuste de tipo de cambio**: La plataforma permite al administrador ajustar la tasa de cambio predeterminada, que inicia con $1 = Q7.50.
- **Control de transacciones y cobros**: El sistema aplica automáticamente un cobro del 0.25% sobre el monto total cada vez que se utiliza la tarjeta, cargo que se refleja en el saldo de la cuenta del titular.
- **Reportes financieros**: El portal permite la generación de reportes detallados, como movimientos de saldo, cuentas bloqueadas, detalles completos de cuentas, estado general de cuentas, y reportes de cierre, brindando al personal administrativo una visión clara y precisa para la toma de decisiones.

Esta plataforma web permite a los usuarios gestionar de manera segura sus tarjetas de crédito y consultar sus transacciones financieras desde cualquier dispositivo con acceso a Internet, ofreciendo una experiencia de usuario confiable y fácil de usar.

Tecnologías Utilizadas

- **Frontend:** Angular 17.x, utilizando TypeScript.
- **Backend:** Express con Node.js.
- **Base de datos:** MariaDB, con consultas y migraciones para el manejo eficiente de datos.

Funcionalidades Clave

- **Zona de Información de Tarjetas**: Acceso a información detallada de la cuenta de tarjeta de crédito del usuario, con opciones para ver los últimos comentarios de beneficios ingresados por los usuarios y solicitar un recordatorio de PIN enviado al correo electrónico asociado.
- **Gestión de Movimientos de Saldo**: Permite a los usuarios consultar los movimientos de saldo en su cuenta de tarjeta de crédito, incluyendo aumentos y reducciones hasta una fecha específica, mostrando el tipo de movimiento, monto y saldo actualizado.
- **Control de Cuentas de Usuarios**: Módulo de administración que permite a los administradores crear, modificar o eliminar cuentas de usuario. Las cuentas se estructuran con una sintaxis estándar y pueden clasificarse en tipos de cuenta “normal” y “gold” según la moneda (quetzales o dólares).
- **Habilitación y Deshabilitación de Tarjetas**: Funcionalidad para activar o desactivar tarjetas de crédito en situaciones de robo, extravío o morosidad, registrando el motivo en cada caso.
- **Ajuste de Tipo de Cambio**: Opción de configuración para actualizar la tasa de cambio predeterminada, fijada inicialmente en $1 = Q7.50.
- **Cobro por Uso de Tarjeta**: Implementación automática de un cobro del 0.25% sobre el monto total en cada uso de la tarjeta, reflejado directamente en el saldo del titular.
- **Reportes Financieros**: Generación de reportes detallados para la administración, incluyendo informes de movimiento de saldo, cuentas bloqueadas, detalles de cuentas específicas, estado general de cuentas y reportes de cierre para la toma de decisiones estratégicas.

Instalación

Frontend (Angular)

- Clonar el repositorio del proyecto:

```
git clone https://github.com/9601dani/SS1-WebService
```

- Instalar las dependencias:

```
cd SS1-WebService/app-frontend
npm install
```

- Levantar el servidor de desarrollo:

```
ng serve
```

Backend (Express)

- Clonar el repositorio del backend:

```
git clone https://github.com/9601dani/SS1-WebService
```

- Instalar las dependencias:

```
cd SS1-WebService/app-backend
npm install
```

1. Configurar las variables de entorno en un archivo .env para conectar la base de datos MariaDB, configurando parámetros como host, usuario y contraseña.

- Levantar el servidor:

```
pm2 start all
```

Base de Datos (MariaDB)

- Instalar MariaDB:

```
sudo apt install mariadb-server
sudo systemctl start mariadb
```

1. La base de datos se encuentra en la nube, solamente es acceder mediante las credenciales

Estructura del Proyecto

Backend

El backend maneja la lógica del negocio y la interacción con la base de datos, implementado con **Express**. Algunas de las principales carpetas son:

config: Esta carpeta contiene la configuración global de la aplicación, como parámetros generales y posibles configuraciones específicas que no pertenecen directamente a la base de datos. En esta carpeta podrían alojarse archivos de configuración JSON o módulos que gestionen el entorno.

db: Aquí se encuentra el archivo `db.js`, el cual se encarga de la configuración y conexión a la base de datos. Este archivo contiene la lógica para establecer la conexión con el sistema de gestión de bases de datos (DBMS) y exporta la conexión para ser utilizada en otras partes de la aplicación.

middleware: En esta carpeta está el archivo `verifytoken.middleware.js`, un middleware de autenticación. Este archivo contiene la lógica para verificar los tokens de los

usuarios (como los JSON Web Tokens - JWT) y proteger rutas específicas, asegurando que solo los usuarios autenticados puedan acceder a ciertas funcionalidades de la aplicación.

src: Esta es la carpeta principal que contiene los módulos funcionales de la aplicación. Dentro de `src`, se encuentran las siguientes subcarpetas:

- **controllers**: Esta subcarpeta contiene los controladores que manejan la lógica de negocio para cada una de las rutas de la API. Cada archivo de controlador gestiona las operaciones de una entidad específica en el sistema:
    - `admin.controller.js`: Este controlador se encarga de las operaciones administrativas, como la gestión de usuarios, roles, y otras configuraciones de administración.
    - `api.controller.js`: Controlador para las operaciones generales de la API, que puede incluir lógica para funcionalidades comunes en el sistema.
    - `test.controller.js`: Controlador utilizado para pruebas o rutas de ejemplo, que permite verificar el correcto funcionamiento de la API.
    - `user.controller.js`: Este controlador maneja las operaciones relacionadas con los usuarios, como la creación de cuentas, autenticación, y actualización de perfiles.
- **routes**: Esta subcarpeta contiene los archivos de rutas, donde se definen las rutas de la API y los controladores correspondientes que manejarán cada una. Aquí, cada archivo de rutas vincula las rutas HTTP con los métodos de los controladores específicos:
    - `admin.routes.js`: Define las rutas para las funcionalidades de administración, enlazando cada ruta con los métodos de `admin.controller.js`.

- `api.routes.js`: Contiene las rutas generales de la API, usualmente para aquellas operaciones que no están relacionadas con un módulo específico.
- `test.routes.js`: Rutas que se utilizan para pruebas o para verificar el funcionamiento básico de la API.
- `user.routes.js`: Define las rutas para las operaciones de usuario, como autenticación, registro y gestión de perfiles.

index.js: Este es el archivo principal de la aplicación. Aquí inicializar la aplicación, configurar los middlewares globales y agregó las rutas principales. Este archivo también contiene la lógica para iniciar el servidor, poniendo en marcha la aplicación.

Rutas Compartidas para garantizar la interoperabilidad

- http://34.42.51.137:3001/api/auth/token
- http://34.42.51.137:3001/api/card
- http://34.42.51.137:3001/api/pay

Frontend

El frontend de esta aplicación está desarrollado con **Angular**, proporcionando una interfaz visual y responsiva para los usuarios. La organización de carpetas facilita la gestión de componentes, servicios y rutas en un entorno modular y mantenible. A continuación, se describe cada sección del proyecto.

Estructura Principal

1. **app/frontend**: Carpeta raíz del frontend que contiene las configuraciones de Angular, módulos y archivos de recursos.
2. **src/app/components**: La carpeta `components` contiene todos los componentes organizados por módulos específicos, cada uno con una funcionalidad definida.
    - **admin**: Componentes relacionados con la administración del sistema.
        - `add-user`: Componente para agregar nuevos usuarios en la zona administrativa.
        - `balance-reduction`: Componente para realizar la reducción de saldo en las cuentas de usuario.
        - `report-accounts`: Componente para generar reportes de cuentas.
        - `report-blocked`: Componente para reportar y ver cuentas bloqueadas.
        - `report-closed`: Componente para el reporte de cuentas cerradas.
        - `report-details`: Componente para visualizar los detalles de cada cuenta.
        - `report-trans`: Componente para consultar reportes de transacciones.

        - `show-cards`: Componente que permite visualizar las cuentas de los usuarios.
        - `show-exchange`: Componente para mostrar y administrar el tipo de cambio.
    - **commons**: Componentes de uso común en diferentes partes de la aplicación.
        - `dynamic-page`: Componente que permite mostrar páginas de contenido dinámico según la ruta.
        - `home`: Componente principal de la página de inicio.
        - `login`: Componente para la autenticación de usuarios.
        - `navbar`: Barra de navegación para facilitar el acceso a las secciones de la aplicación.
        - `not-found`: Componente para la página de error 404, mostrada cuando la ruta no es encontrada.
    - **user**: Componentes específicos de la funcionalidad del usuario.
        - `edit-profile`: Componente para la edición de información de perfil de usuario.
        - `show-transactions`: Componente para que el usuario pueda visualizar sus transacciones.
3. **src/app/interfaces**: Esta carpeta contiene interfaces TypeScript que definen los tipos y estructuras de datos utilizados en la aplicación.
    - `interfaces.ts`: Archivo con las definiciones de interfaces utilizadas para estructurar datos en los servicios y componentes.
4. **src/app/services**: Aquí se encuentran los servicios que gestionan la lógica de negocio y las interacciones con la API.

- `admin.service.ts`: Maneja las operaciones administrativas como la gestión de usuarios, reportes y configuración del sistema.
- `external.service.ts`: Servicio para manejar la comunicación con APIs externas o servicios externos al sistema principal.
- `local-storage.service.ts`: Gestiona el almacenamiento local en el navegador, como guardar tokens de sesión y otros datos temporales.
- `user.service.ts`: Servicio para operaciones relacionadas con el usuario, como la autenticación, la gestión del perfil y la obtención de transacciones.

Vistas (Rutas)

Las rutas de la aplicación se encuentran definidas en `app-routing.module.ts` y permiten el acceso organizado a cada sección de la aplicación. A continuación, se describen las rutas principales:

- **Rutas de navegación principal**
  - `/home`: Dirige al componente `HomeComponent`, que representa la página de inicio de la aplicación.
  - `/login`: Dirige al componente `LoginComponent` para la autenticación de usuarios.
  - `/add-accounts`: Redirige al componente `AddUserComponent`, donde se permite agregar cuentas nuevas en la administración.
  - `/page/:page`: Dirige al `DynamicPageComponent`, que muestra contenido dinámico basado en la URL.

    - `/show-accounts`: Muestra las cuentas de los usuarios mediante el componente `ShowCardsComponent`.
    - `/show-change`: Dirige al `ShowExchangeComponent`, para mostrar y gestionar el tipo de cambio.
    - `/balance-reduction`: Dirige al `BalanceReductionComponent` para realizar reducciones de saldo en cuentas.
    - `/show-transactions`: Permite al usuario visualizar sus transacciones mediante el componente `ShowTransactionsComponent`.
- **Rutas de administración y reportes**
    - `/transactions`: Muestra el reporte de transacciones a través del `ReportTransComponent`.
    - `/blocked`: Dirige al `ReportBlockedComponent`, donde se visualizan las cuentas bloqueadas.
    - `/details`: Muestra los detalles de cuentas con el `ReportDetailsComponent`.
    - `/report-accounts`: Accede al reporte general de cuentas mediante el `ReportAccountsComponent`.
    - `/closed`: Presenta el reporte de cuentas cerradas a través del `ReportClosedComponent`.
- **Rutas de edición de perfil**
    - `/edit/profile`: Permite la edición del perfil del usuario en el `EditProfileComponent`.
- **Rutas de manejo de errores**

- `/**`: Muestra el `NotFoundComponent` cuando se ingresa una ruta no válida, mostrando una página de error 404

Base de Datos (MariaDB)

Descripción General

La estructura de la base de datos está diseñada en un modelo relacional, donde cada tabla representa una entidad específica del sistema. La organización permite gestionar de forma eficiente usuarios, roles, tarjetas de crédito, transacciones y configuraciones administrativas. A continuación se describen las tablas principales y las relaciones clave en la base de datos.

Tablas Principales

1. **user**
    - **Descripción**: Almacena la información de los usuarios registrados en el sistema.
    - **Campos principales**:
        - id: Identificador único del usuario.
        - username: Nombre de usuario.
        - password: Contraseña encriptada.
        - email: Correo electrónico único del usuario.
        - notifyme: Indica si el usuario recibe notificaciones (0 o 1).
        - auth_token: Token de autenticación para sesiones activas.
    - **Relaciones**:

- Relacionada con la tabla role a través de la tabla relacional user_has_role para asignar roles a los usuarios.
- Conectada a comment para gestionar comentarios de usuarios.
- Conectada a credit_card para asociar tarjetas de crédito con los usuarios.
- Relacionada con user_information para almacenar información adicional del usuario.

2. **credit_card**
   - **Descripción**: Contiene los detalles de las tarjetas de crédito de los usuarios.
   - **Campos principales**:
     - id: Identificador único de la tarjeta de crédito.
     - credit_card_number: Número de la tarjeta, único por tarjeta.
     - account_type: Tipo de cuenta (normal o gold).
     - credit_limit: Límite de crédito de la tarjeta.
     - current_balance: Saldo actual de la tarjeta.
     - state: Estado de la tarjeta (activo, deshabilitado, bloqueado).
   - **Relaciones**:
     - Relacionada con credit_card_report para registrar reportes de la tarjeta.
     - Conectada con transaction para almacenar las transacciones de la tarjeta.
3. **transaction**
   - **Descripción**: Registra las transacciones realizadas con las tarjetas de crédito.
   - **Campos principales**:
     - id: Identificador único de la transacción.
     - transaction_type: Tipo de transacción (incremento o reducción).

        - amount: Monto de la transacción.
        - fee: Comisión aplicada a la transacción.
        - exchange_rate: Tipo de cambio utilizado en la transacción.
    - **Relaciones**:
        - Relacionada con credit_card a través de FK_Card, indicando la tarjeta a la que pertenece cada transacción.

4. **role**
    - **Descripción**: Define los roles de los usuarios en el sistema, permitiendo un control de acceso diferenciado.
    - **Campos principales**:
        - id: Identificador único del rol.
        - name: Nombre del rol (por ejemplo, administrador, cliente).
    - **Relaciones**:
        - Conectada a user mediante la tabla user_has_role para asignar roles a los usuarios.
        - Relacionada con page a través de role_has_page, para definir permisos de acceso en el sistema.

5. **comment**
    - **Descripción**: Almacena los comentarios de los usuarios sobre sus experiencias con el sistema.
    - **Campos principales**:
        - id: Identificador único del comentario.
        - comment: Texto del comentario.
    - **Relaciones**:

        - Relacionada con user a través de FK_User, indicando el autor de cada comentario.

Tablas Relacionales

1. **user_has_role**
    - **Descripción**: Relaciona los usuarios con sus roles asignados.
    - **Campos principales**:
        - user_id: Identificador del usuario.
        - role_id: Identificador del rol.
    - **Relaciones**:
        - Conectada con user y role para asignar múltiples roles a un usuario.
2. **credit_card_report**
    - **Descripción**: Registra incidentes o reportes relacionados con las tarjetas de crédito.
    - **Campos principales**:
        - id: Identificador del reporte.
        - report_type: Tipo de reporte (robo, pérdida, tardanza, disponible, bloqueado, rechazado).
    - **Relaciones**:
        - Conectada a credit_card a través de FK_Card.
3. **role_has_page**
    - **Descripción**: Define los permisos de los roles sobre las diferentes páginas del sistema.
    - **Campos principales**:

        - FK_Role: Identificador del rol.
        - FK_Page: Identificador de la página.
        - can_create, can_edit, can_delete: Permisos específicos (1 o 0).
    - **Relaciones**:
        - Conectada con role y page, permitiendo definir permisos de acceso por rol para cada página.

4. **api_keys_has_routes**
    - **Descripción**: Asocia claves API con rutas específicas, limitando el acceso de clientes API a ciertas rutas.
    - **Campos principales**:
        - FK_Api_Key: Identificador de la clave API.
        - route: Ruta específica accesible mediante la clave API.
    - **Relaciones**:
        - Conectada a api_keys, vinculando rutas específicas con claves API.

Tablas Auxiliares

1. **api_keys**
    - **Descripción**: Almacena claves API para clientes que acceden al sistema mediante integraciones.
    - **Campos principales**:
        - id: Identificador de la clave API.
        - client_id: ID del cliente.
        - client_secret: Secreto de cliente para autenticación.
2. **company_settings**

- **Descripción**: Configuraciones generales del sistema, permitiendo personalizar aspectos de la empresa.
- **Campos principales**:
  - key_name: Nombre de la configuración.
  - key_value: Valor de la configuración.

3. **module**
   - **Descripción**: Define los módulos funcionales disponibles en el sistema.
   - **Campos principales**:
     - id: Identificador del módulo.
     - name: Nombre del módulo.
     - is_available: Estado del módulo (disponible o no).
4. **page**
   - **Descripción**: Contiene las páginas o vistas del sistema y permite gestionar su disponibilidad.
   - **Campos principales**:
     - name: Nombre de la página.
     - path: Ruta de la página.
5. **user_information**
   - **Descripción**: Almacena información adicional del usuario, como su nombre completo y detalles de contacto.
   - **Campos principales**:
     - name: Nombre del usuario.
     - phone: Número de teléfono.
     - address: Dirección del usuario.

Relaciones Clave

- **user - role**: Un usuario puede tener múltiples roles, definidos en la tabla user_has_role, lo que permite controlar los permisos de cada usuario en el sistema.
- **credit_card - user**: Cada tarjeta de crédito está asociada a un usuario mediante FK_User.
- **transaction - credit_card**: Las transacciones están asociadas a una tarjeta de crédito específica, permitiendo registrar las operaciones realizadas en cada cuenta.
- **credit_card_report - credit_card**: Cada reporte se asocia con una tarjeta de crédito, registrando incidentes como robo, pérdida o bloqueo.
- **role - page**: Mediante role_has_page, los roles tienen permisos específicos para cada página del sistema, permitiendo gestionar el acceso a las funcionalidades.

Ejecución

- **Frontend**:

  bash

  ng serve
- **Backend**:

  bash

  pm2 restart all
- **Base de Datos**: Asegurarse de que el servidor MariaDB esté corriendo:

  bash

  sudo systemctl start mariadb

Créditos

Este manual ha sido creado para servir como una guía técnica completa para la implementación y uso del **Portal de Crédito** desarrollado en el marco del curso **Seminario de Sistemas 1** en el **Centro Universitario de Occidente**. El contenido del manual abarca la configuración del sistema, especificaciones técnicas, arquitectura del software, procesos de instalación, mantenimiento, solución de problemas y mejores prácticas para optimizar el rendimiento de la plataforma.

Equipo de Desarrollo

- **Erick Daniel Morales Xicará**

  Estudiante de Ingeniería en Sistemas - #3 - 201930699

Este manual fue elaborado como parte del curso **Seminario de Sistemas 1**, bajo la supervisión del **Ing. Pedro Domingo** en el **Centro Universitario de Occidente**, durante el segundo semestre de 2024.

Agradecemos su confianza en nuestro equipo para la implementación de esta solución tecnológica. Para cualquier consulta técnica o soporte adicional, no dude en contactar a nuestro equipo de atención técnica.

Manual Técnico - Portal de Crédito