**Supuestos Banco Ivan Valdes**

1. **Entorno del Servidor**:
   * El servidor backend está corriendo en localhost:3000.
   * Las rutas del servidor siguen las convenciones mencionadas en el código (/api/cuentas, /api/login, /api/logout, /api/registro, /api/transaccion, /api/reporte, /api/asignarCuenta).
2. **Interacción del Usuario**:
   * Hay una interfaz de usuario con varios botones y formularios para login, registro, transacciones, etc.
   * El usuario puede iniciar sesión, cerrar sesión, crear cuentas, realizar transacciones, y generar reportes.
3. **Estructura del DOM**:
   * Existen elementos con IDs específicos (menu-auth, menu-logout, cerrarSesion, bienvenida, mensajeBienvenida, botonIniciarSesion, cuentasContainer, mensaje, etc.).
   * Se espera que ciertos formularios y botones (asignarCuentaForm, transaccionForm, loginForm, registroForm, generarReporteButton) existan y tengan ciertos campos.

**Supuestos Específicos**

1. **Autenticación y Autorización**:
   * El estado de autenticación del usuario se maneja a través de la variable usuarioActual.
   * La autenticación exitosa devuelve el nombre de usuario, que se almacena en usuarioActual.
2. **Navegación**:
   * Se utiliza page.js para la navegación entre secciones de la aplicación.
   * Cada sección del sitio está representada por un <section> que se muestra u oculta según sea necesario.
3. **Interacción con el Servidor**:
   * Las solicitudes al servidor son asíncronas y se manejan con fetch.
   * Las respuestas del servidor contienen un campo success para indicar el éxito de las operaciones.
   * La estructura de las respuestas JSON sigue un formato esperado (result.cuentas, result.success, etc.).
4. **Estado del Usuario y Visualización**:
   * La visualización del menú de autenticación depende del estado de usuarioActual.
   * La bienvenida y las cuentas de usuario se actualizan en función de si usuarioActual está definido.
   * Los mensajes de bienvenida y de error se muestran mediante manipulación del DOM.
5. **Manejo de Errores**:
   * Se maneja la visualización de mensajes de error y éxito mediante la función mostrarMensaje.
   * Se asume que los errores se pueden capturar y manejar adecuadamente con bloques try-catch.
6. **Transacciones y Cuentas**:
   * El formulario de transacciones espera campos específicos (cuentaID, tipoTransaccion, monto, moneda).
   * Se puede crear y asignar cuentas a usuarios autenticados.
   * Las cuentas tienen propiedades como tipoCuenta, id, saldo, y estado.
7. **Reportes**:
   * El reporte diario se genera y muestra en el DOM.
   * Las transacciones en el reporte tienen propiedades como tipoTransaccion, monto, comision, cuentaID.
8. **Conversiones de Moneda**:
   * La función convertirAMonedaLocal se usa para convertir montos a una moneda local basada en tasas de cambio específicas.
9. **Operaciones Programadas**:
   * Se ejecutan comisiones mensuales e intereses diarios con intervalos programados (setInterval).

**Resumen**

**Interacción con una API RESTful**: La aplicación depende de una API RESTful que sigue una estructura y convenciones específicas.

* **DOM Específico**: La estructura HTML y los IDs de elementos deben coincidir con los referenciados en el código.
* **Estado del Usuario**: El estado del usuario y la visualización del contenido se gestionan dinámicamente en función de la autenticación.
* **Manejo de Errores y Respuestas del Servidor**: Se supone que las respuestas del servidor están bien formateadas y que los errores se pueden manejar adecuadamente.
* **Operaciones Financieras**: La lógica de operaciones financieras (transacciones, cuentas, reportes) está implementada con ciertas expectativas sobre la estructura de datos y las acciones permitidas.