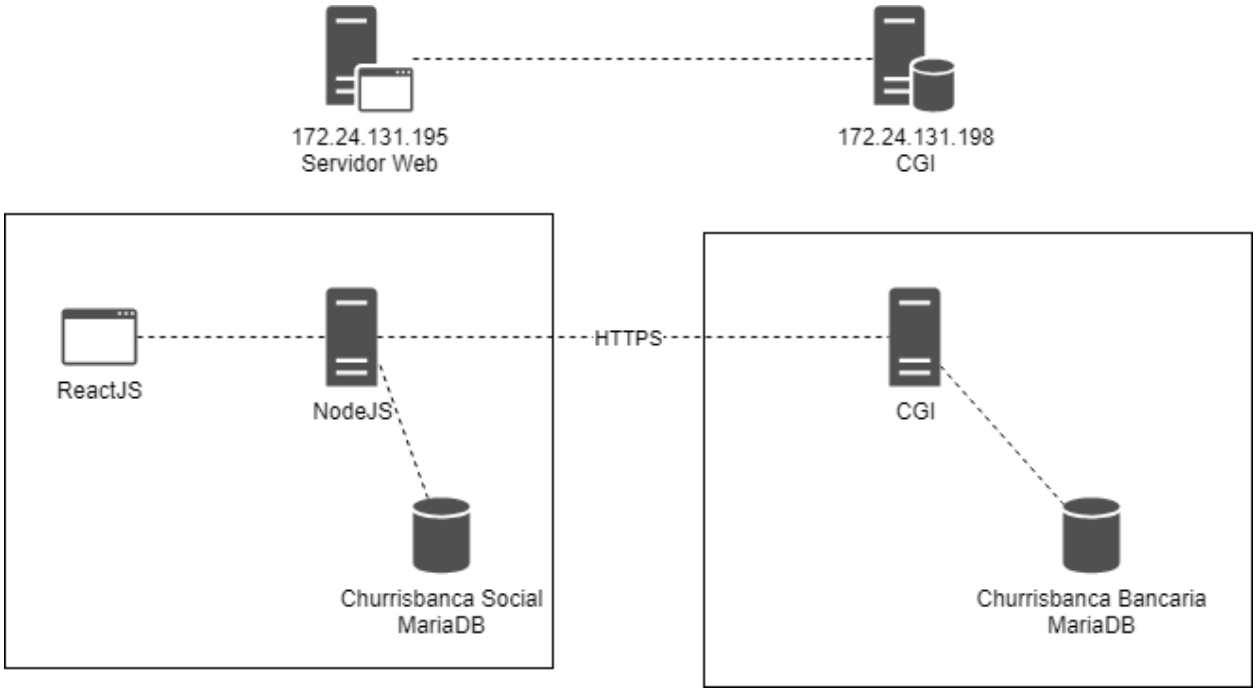
Alonso León Brandon Mora Carlos Sánchez Emilia Víquez

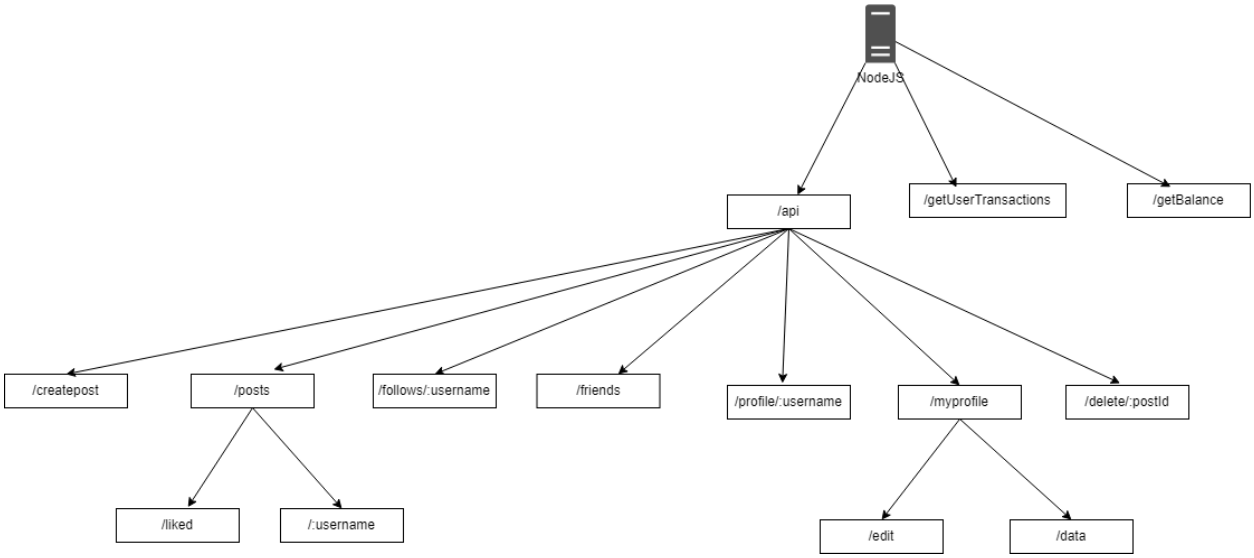
Proyecto Final
Churrisbanca Equipo 01

Diagramas de uso del programa

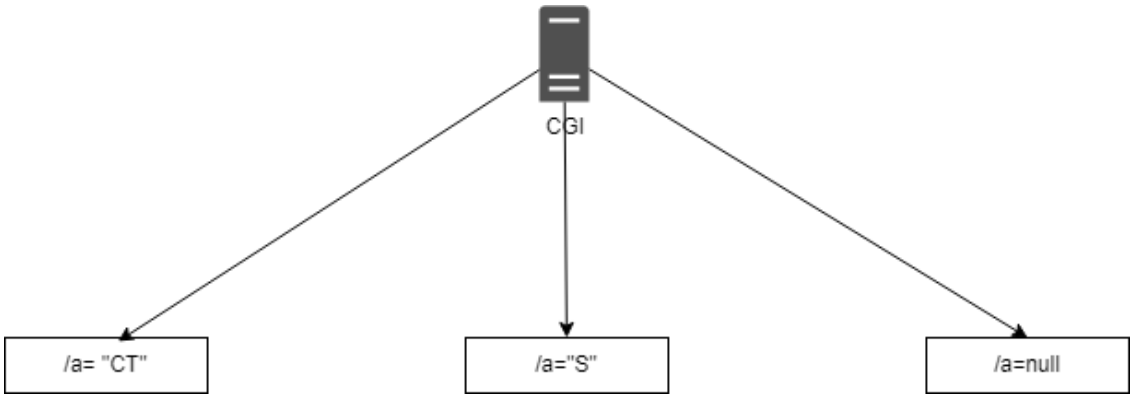
Como puede notarse en el diagrama, la aplicación consta de dos servidores, uno para el sistema Web y otro para el aplicativo CGI. Cada uno con su respectiva IP. Dentro del primer servidor, existe un aplicativo implementado con ReactJS y un servidor implementado con la tecnología NodeJS. Estos se conectan a una base de datos MariaDB, la cual contiene la información sobre usuarios, contraseñas, posts realizados, usuarios seguidos y demás información de interés sobre los usuarios.



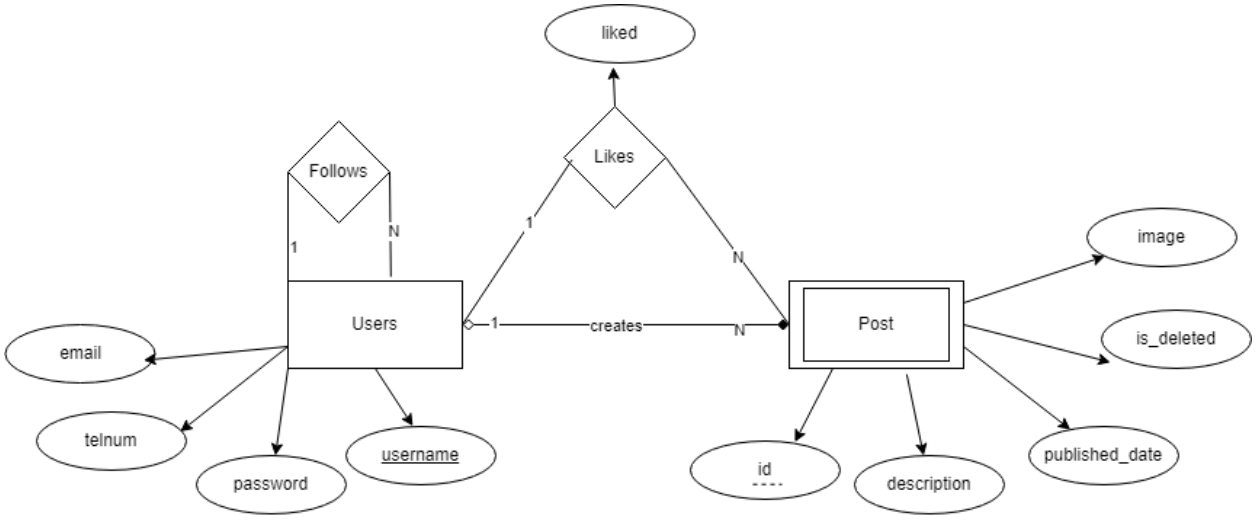
Además, dentro del servidor NodeJS, hay diversos servicios, como crear post, ver los post que un usuario tiene disponibles en su muro social, ver los posts que un usuario ha publicado, ver los likes en un post, crear una relación de seguimiento, buscar amigos, ver el perfil de un amigo, o el perfil del usuario mismo, editar o ver los datos propios y eliminar un post. Las mencionadas anteriormente se realizan dentro del url api, para indicar que involucra funciones dentro del mismo servidor. Las funciones para ver las transacciones de un usuario y obtener el balance de un usuario no se encuentran con la dirección /api, dado que estas se dirigen al servidor CGI. Con respecto a crear una transacción, esta se realiza en el servidor CGI, aunque posee una comprobación de firma en el servidor de NodeJS.



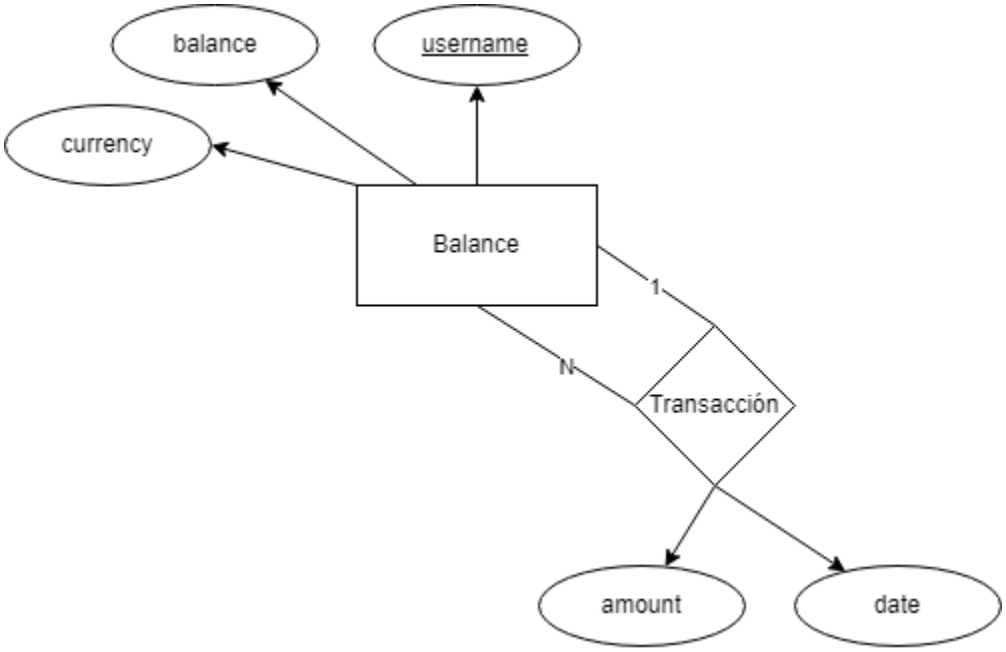
En el servidor CGI, se indica la acción con un parámetro en el url recibido, estos indican si se quiere crear una transacción (create transaction), ver las transacciones disponibles (show transactions) o simplemente ver los datos del usuario (nulo).



A continuación se muestran diagramas de entidad-relación para las bases de datos. El primero corresponde a la base de datos en el servidor web. Este tiene dos entidades: usuario y publicación. La segunda corresponde a una entidad débil, dado que depende del usuario que lo crea. Además, existe una relación de seguimiento que se da entre dos usuarios y una relación de *Like* (que puede ser positiva o negativa), que se da entre un usuario y una publicación.

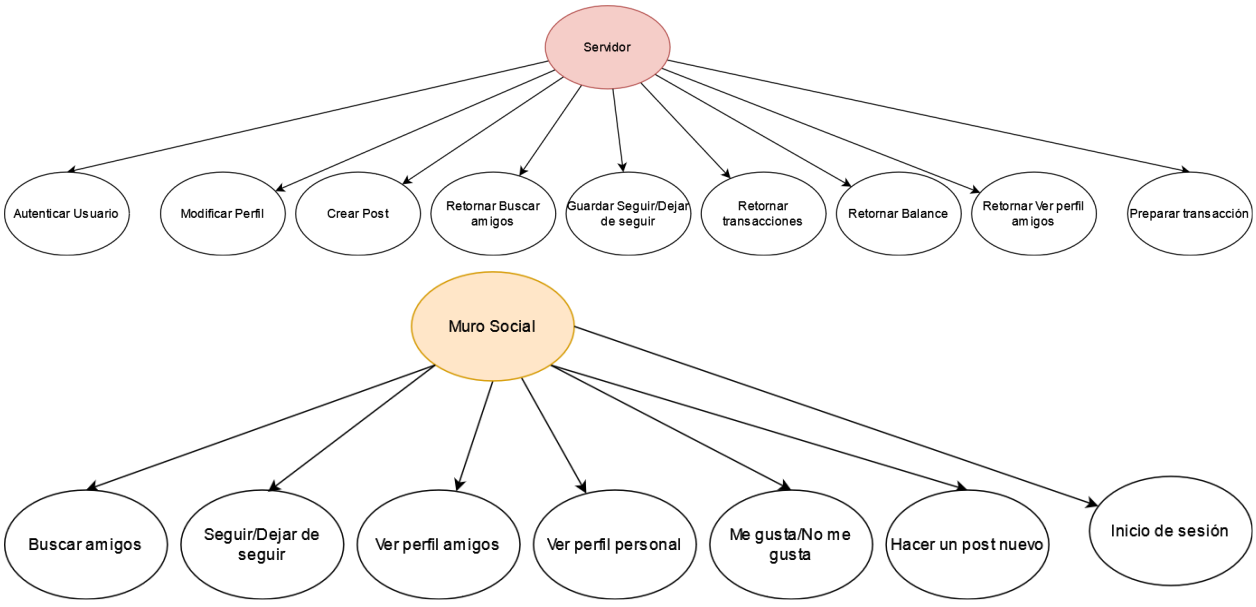


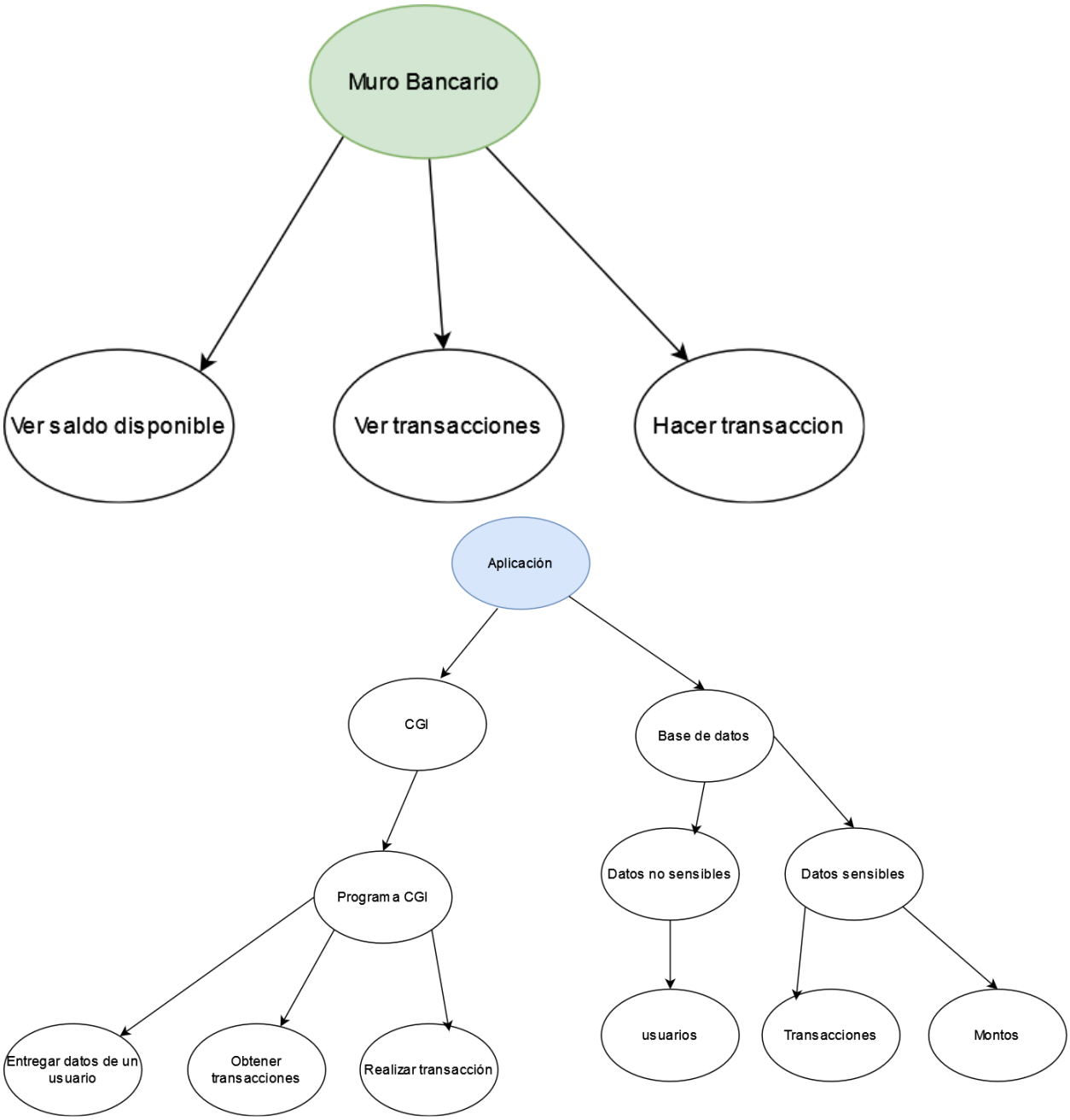
Seguidamente, puede observarse el diagrama para la base de datos en el servidor CGI. El cuál tiene una única entidad, que corresponde al balance de cada uno de los usuarios, el cual se puede relacionar con el balance de otro usuario por medio de una transacción.



Árbol del todo y las partes

Subárboles

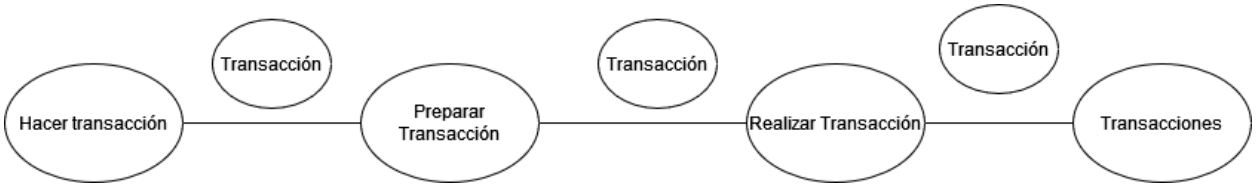




Interacciones

Con respecto a las interacciones entre componentes dentro de la Churris Banca, se resaltan 8 cadenas principales, en donde cada una consta con sus respectivos objetivos directos, objetivos indirectos, vulnerabilidades, riesgos y controles de seguridad, los cuales se detallan a continuación.

Cadena 1: Transacción con firma digital



Esta cadena representa las interacciones entre componentes que se realizan cuando un usuario crea una transacción nueva desde la página web. La información traspasada entre cada componente está representada por otro componente llamado “Transacción”; este componente representa la información necesaria para incluir una nueva fila a la tabla de “Transacciones” de la base de datos bancaria traspasada mediante canales de comunicación.

Objetivos directos:

- Prevenir que la información contenida en Transacción sea alterada
- Prevenir que Hacer Transacción sea utilizado por un usuario no autorizado

Objetivos indirectos:

- Prevenir que Hacer Transacción sea utilizado por un usuario no autorizado
    - Prevenir que Preparar transacción sea utilizado por un usuario no autorizado
    - Prevenir que Realizar transacción sea utilizado por un usuario no autorizado
    - Prevenir que Transacciones sea utilizado por un usuario no autorizado
    - Prevenir que Transacción sea emitido por un usuario no autorizado
- Prevenir que la información contenida en Transacción sea alterada
    - Prevenir que la información en Hacer transacción sea alterada
    - Prevenir que la información en Preparar Transacción sea alterada
    - Prevenir que la información en Realizar Transacción sea alterada
    - Prevenir que la información en Transacciones sea alterada

Vulnerabilidades:

- Es posible cambiar la cuenta de origen de una Transacción
- Es posible cambiar el destino de una Transacción
- Es posible cambiar el monto de una Transacción
- Es posible hacer solicitudes directamente a Prepara Transacción o Realizar transacción sin una entrada directa de la persona dueña de la cuenta de origen

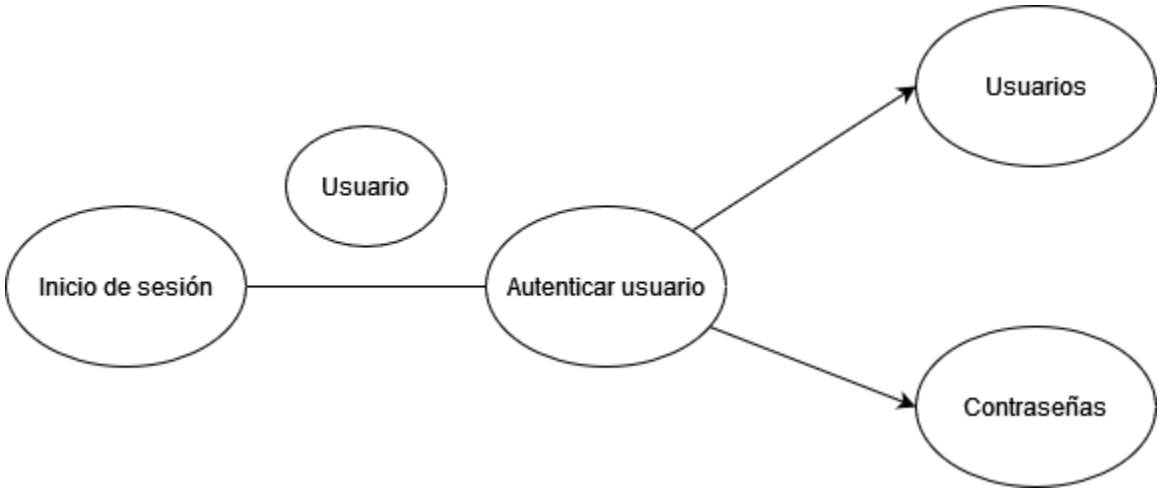
Riesgos:

- Si se cambia la cuenta de origen de una transacción, es posible hacer operaciones a nombre de otro usuario de la Churrisbanca
- Se puede modificar la información en tránsito para cambiar la cuenta de destino de la transacción, beneficiando al atacante o solamente atentando contra la integridad del sistema
- De igual manera, se podría modificar el monto para afectar a la persona remitente y beneficiar a la persona receptora

Controles:

- Para manejar la autorización a nivel de Hacer Transacción, se implementa un mecanismo de autenticación con Json Web Tokens que valida la sesión del usuario y corrobora su identidad. Para la autorización (e integridad) a nivel de Transacción, se implementa la firma de los datos con la llave privada del usuario; esta firma se agrega a los datos contenidos en Transacción, por lo que sus objetivos de seguridad también se aplican, ocasionando que conservar la integridad de la firma sea un objetivo. En cada componente (exceptuando el componente “Hacer transacción”) se realiza una comprobación de la firma de los datos contra la llave pública del usuario remitente, cubriendo ambos objetivos de autorización e integridad.
- La integridad de la información contenida en Transacción emitida mediante canales de comunicación, se resguarda asegurando la conexión entre cada componente. Esto implica, entre otras cosas, la necesidad de utilizar un protocolo seguro de transmisión de datos (HTTPS) entre cada servidor del sistema y de confiar, mediante la utilización de certificados digitales, en la identidad de cada uno de los servidores que interactúan en esta cadena. Ambos controles fueron implementados en la aplicación “ChurrisBanca”

Cadena 2: Login



Esta cadena representa la acción de iniciar sesión para un usuario, implica la unión de un nombre de usuario y una contraseña. Nace de un control implementado para manejar la autenticación y autorización en diversas partes de la aplicación; como tal, definimos objetivos de seguridad para los componentes involucrados con el fin de minimizar las posibles vulnerabilidades.

Objetivos directos:

- Prevenir que la información contenida en Usuario sea alterada
- Prevenir que sea posible suplantar la identidad de un Usuario

Objetivos indirectos:

- Prevenir que la información de Usuario sea alterada en el Inicio de Sesión
- Prevenir que la información de Usuario sea alterada al Autenticar Usuario
- Prevenir que sea posible suplantar la identidad de un usuario en el Inicio de Sesión
- Prevenir que sea posible suplantar la identidad de un usuario al Autenticar Usuario

Vulnerabilidades:

- Alterar el nombre de usuario en la solicitud
- Que un usuario revele su contraseña

Riesgos:

- Suplantar la identidad de un usuario al modificar el nombre de usuario en las solicitudes
- Robo de contraseñas por medio del cliente de la aplicación

Controles:

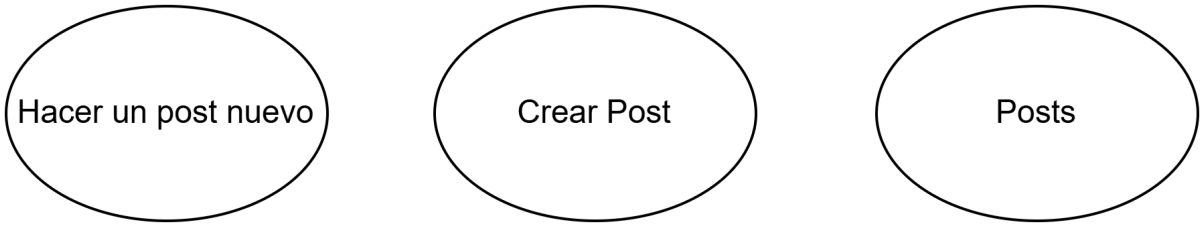
- Para evitar la suplantación de identidad, se genera un token de representación para el usuario utilizando JWT (Json Web Tokens), este token se almacena de forma segura en el navegador del usuario y representa al usuario en la aplicación. En él, está escrito el

nombre de usuario; su validez se comprueba implementando un control de llave secreta que verifica la veracidad del token.

- Para evitar el robo de contraseñas, se implementa una cobertura visual para la escritura de la contraseña

Cadena 3: Crear post

Esta cadena de seguridad está enfocada en permitir la creación de posts de los usuarios en el muro social, y consiste en el componente de “Hacer un post nuevo” el cual es el botón en el muro social para crear un post, el usuario llena los datos sobre lo que va a tener su post, el cual puede ser un texto y una imagen del post, luego “Crear Post” es un endpoint de la aplicación Node que recibe los datos del post por crear y los almacena en la base de datos en los “Posts”.



Objetivos directos:

- Prevenir que un usuario pueda crear posts en nombre de otros usuarios.
- Prevenir datos maliciosos por parte del usuario en la creación del post que serán almacenados.
- Prevenir el acceso no autenticado al endpoint de “Crear Post”.

Objetivos indirectos:

- Prevenir que una persona acceda al endpoint de “Crear Post” y crear posts de forma no autenticada.

Vulnerabilidades:

- El endpoint de crear post puede ser accedido sin restricciones.
- Almacenamiento de cualquier tipo de dato en la base de datos de la imagen del post.
- Tamaño de almacenamiento de la imagen del post indefinido.

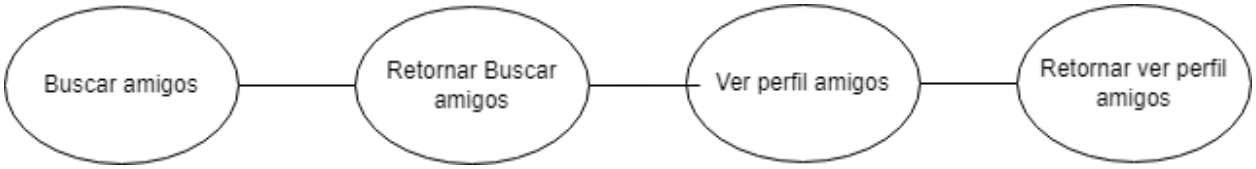
Riesgos:

- Un usuario puede subir archivos diferentes a imágenes a la base de datos.
- Una persona puede comunicarse directamente con el endpoint de crear post sin estar autenticado.

Controles:

- Definir controles que verifiquen el tipo de archivo subido en el endpoint de crear post para permitir solo imagenes.
- Definir un control en el endpoint de crear post para permitir archivos que no excedan un tamaño definido.
- Poner medidas de autenticación en el uso del endpoint de crear post.

Cadena 4: Ver perfil amigos



Esta cadena de seguridad consiste en que, para poder ver el perfil de otro usuario, la persona debe haber ingresado al sistema y debe tener una relación de seguimiento mutuo con el usuario cuyo perfil quiere ver.

Objetivos directos:

- Queremos prevenir que Buscar amigos permite seleccionar el perfil de un usuario con el que no hay una relación de seguimiento mutua.
- Queremos prevenir que un usuario que acaba de dejar de seguir a otro pueda ver el perfil de este.
- Queremos prevenir que un usuario acceda a la información de alguien con quien la relación de seguimiento no es mutua.

Objetivos indirectos:

- Queremos evitar que un usuario intente acceder a un perfil que no debe poder ver.

Vulnerabilidades:

- Para poder mostrar la página de otro usuario, es necesario hacer una solicitud al servidor que contenga el nombre de dicho usuario. Esto da lugar a que se intente atacar al usuario.
- El nombre del usuario cuyo perfil se desea acceder se envía como parámetro en el URL de la página, volviendo este vulnerable a ser cambiado manualmente.
- Al dejar de seguir un usuario, es posible que el sistema no refresque correctamente la página.

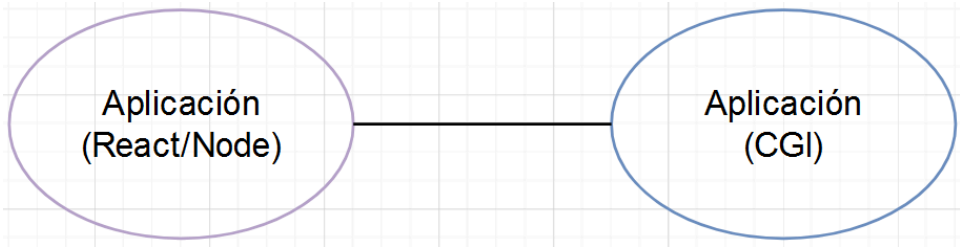
Riesgos:

- Un usuario podría modificar el enlace y acceder a información de un usuario que no es su amigo, escalando así, los privilegios que tiene.
- Un usuario podría modificar el enlace e intentar acceder a la información de un usuario que no existe, causando comportamiento indeterminado en el sistema.
- Un usuario podría acceder al servidor de NodeJS por otros medios que le permitirían acceder a la información de cualquier usuario.

Controles:

- A la hora de mostrar la página de un usuario, se revisa que el usuario exista y tenga una relación de seguimiento mutua con quien realizó la búsqueda.

Cadena 5: Aplicación(React/Node) - Aplicación(CGI)



Esta cadena consiste en la relación a alto nivel que tienen la aplicación de React/Node y CGI entre sí. Más específicamente en cómo se maneja la sesión de un usuario a través de las dos aplicaciones.

Objetivos directos:

Aplicación

- Prevenir que usuarios no autenticados ingresen a la aplicación.
- Prevenir que la identidad de un usuario autenticado sea modificada en una comunicación saliente (e.g. con el CGI)

CGI

- Prevenir que se realicen operaciones directamente sobre el programa CGI.

Objetivos indirectos:

Aplicación

- Prevenir la divulgación de una representación autenticada del usuario para asegurar que solo usuarios autenticados ingresen.
- Prevenir la manipulación de una representación autenticada del usuario para asegurar que la identidad de un usuario no sea modificada.

CGI

- Prevenir que usuarios no autenticados/autorizados realicen operaciones sobre el programa CGI.

Vulnerabilidades:

Aplicación

- La aplicación puede ser accedida por cualquiera.
- La identidad de un usuario puede ser modificada y ser utilizada para comunicaciones salientes.
- El token asociado a la autenticación de un usuario puede ser robado.
- El token asociado a la autenticación de un usuario puede ser modificado.

CGI

- Es posible comunicarse directamente con el CGI para realizar operaciones.
- Cualquier usuario puede realizar operaciones en el programa CGI.

Riesgos:

- Un usuario puede llegar directamente a la aplicación CGI y hacer operaciones maliciosas (e.g. robar dinero).
- Un usuario puede acceder y autenticarse con sus credenciales, pero modificar su identidad para pretender ser alguien más.
- Un usuario puede saltarse la autenticación inicial si logra obtener el token de otra persona.
- Un usuario malicioso puede realizar operaciones en el CGI sin estar autenticado.

Controles:

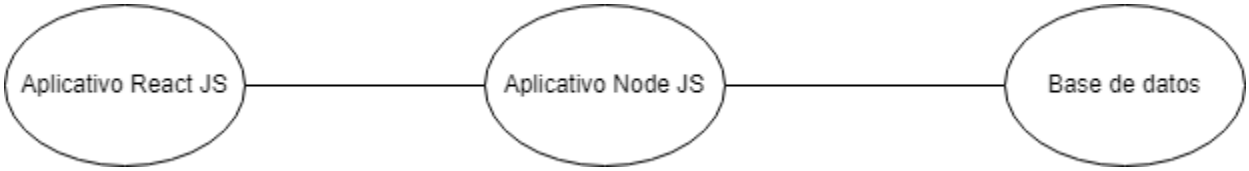
Aplicación

- Utilizar mecanismos de autenticación segura como validación de tokens y 2FA.
- Crear una representación del usuario autenticado, además de enviar exclusivamente la representación autenticada en comunicaciones salientes.
- Las operaciones deben utilizar únicamente la representación del usuario autenticado, y ninguna otra cosa adicional. Los datos asociados al usuario provienen únicamente a esa representación autenticada.
- Utilizar HTTPS y firmas digitales para asegurar la integridad de las comunicaciones.
- Usar opciones de “secure” y “HttpOnly” para evitar que el token sea robado, modificado y utilizado de forma sencilla.

CGI

- Utilizar HTTPS y firmas digitales para asegurar la integridad de las comunicaciones.
- Admitir sólo conexiones desde la aplicación, no conexiones directas. Esto se puede lograr mediante firewall.
- Utilizar sólamente el “paquete” que contiene la representación autenticada del usuario recibida desde la aplicación, para realizar operaciones.

Cadena 6: React-Node-Base de datos



Esta cadena de seguridad consiste en que, para poder desplegar información en la aplicación Web, el cliente de ReactJS debe solicitarla al servidor de NodeJS. Estos deben estar escuchando en diferentes puertos y, la solicitud se hace desde la IP del cliente, no del servidor React, por lo que no es posible cerrar el puerto del servidor NodeJS. Además, el servidor de NodeJS es quien tiene acceso a la base de datos de Churris Banca Social, por lo que todas las solicitudes a la base de datos deben ser implementadas desde dicho servidor.

Objetivos directos:

- Queremos prevenir que un usuario haga consultas directamente a la aplicación de NodeJS.
- Queremos prevenir que un usuario pueda accesar a las respuestas del aplicativo de NodeJS desde un medio diferente a la aplicación de ReactJS.
- Queremos prevenir que un usuario se haga pasar por otro ante el aplicativo de NodeJS.
- Queremos prevenir que un usuario acceda a la base de datos sin antes pasar por el aplicativo de ReactJS.
- Queremos prevenir que un usuario realice modificaciones indebidas a la base de datos mediante una entrada inválida.

Objetivos indirectos:

- Queremos evitar que un usuario intencional o accidentalmente ingrese valores que no debería en el sistema.
- Queremos evitar que un usuario haga solicitudes al aplicativo de Node o a la base de datos directamente (sin pasar por el aplicativo de ReactJS).

Vulnerabilidades:

- El token puede ser utilizado para hacer solicitudes en las que el identificador del usuario puede ser modificado.
- El aplicativo de Node JS puede ser accesado por medios externos a la aplicación de ReactJS.
- La base de datos puede ser accesada por medio del puerto que esta utiliza.

Riesgos:

- Un usuario puede enviar y recibir paquetes al servidor Node utilizando una herramienta que le permita enviar paquetes de red.
- Un usuario puede utilizar el token de sesión para hacer solicitudes sin necesidad del aplicativo de ReactJS.
- Un usuario puede ingresar una entrada con formato inválido en el aplicativo de React, la cual podría llegar hasta la base de datos.
- Un usuario puede acceder a los métodos del aplicativo Node haciendo un llamado al puerto que este utiliza.

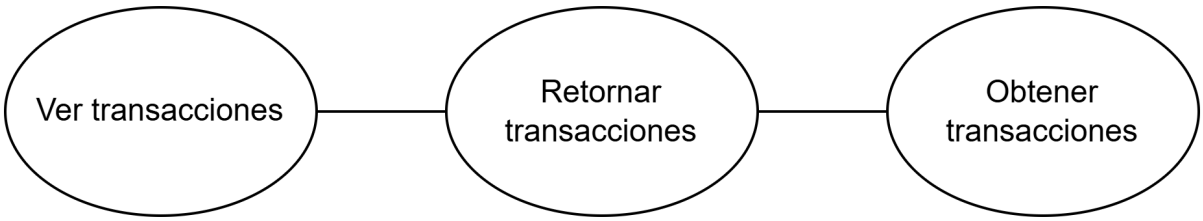
Controles:

- Revisar en cada transacción que el usuario que solicita los datos sea realmente quien dice ser según el token.
- Cerrar el puerto de la base de datos mediante la modificación de las *iptables* con el fin de que esta solo pueda ser accesada por medio del servidor de NodeJS.
- Agregar un nuevo factor de autenticación, un token de actualizar. Mientras este segundo token esté válido, refrescamos el token, si no entonces se elimina el token principal después de un tiempo y ya no se puede volver a usar. Esto permite estar generando nuevos token para los usuarios por un lapso de tiempo, volviendo más complejo el utilizar el token de forma externa a la aplicación.

Cadena 7: Ver transacciones

Esta cadena de seguridad permite ver las transacciones que un usuario ha hecho hacia otros usuarios, y en la siguiente imagen se muestran los componentes que permiten esta acción

Donde “Ver transacciones” es un botón en la aplicación web en el muro bancario, “Retornar transacciones” es el endpoint del servidor Node que va a devolver las transacciones de un usuario que están en el servidor CGI, y “Obtener transacciones” consigue las transacciones desde la base de datos del servidor CGI y las devuelve a “Retornar transacciones”.



Objetivos directos:

- Prevenir que usuarios que no están autenticados en la aplicación React vean transacciones.
- Prevenir que un usuario haga consultas directamente a la aplicación de Node y al endpoint de retornar transacciones.
- Prevenir que un usuario haga consultas directamente al servidor CGI y al endpoint de obtener transacciones.
- Prevenir que un usuario vea las transacciones de otros usuarios.

Objetivos indirectos:

- Prevenir que los usuarios se puedan hacer pasar por otros usuarios en la aplicación React para ver las transacciones de otros usuarios.

Vulnerabilidades:

- El endpoint de retornar transacciones puede ser accedido sin restricciones.
- El endpoint de retornar transacciones permite ver las transacciones de cualquier usuario de forma no autenticada ni autorizada.
- El endpoint de obtener transacciones puede ser accedido sin restricciones.
- El endpoint de obtener transacciones permite ver las transacciones de cualquier usuario de forma no autenticada ni autorizada.

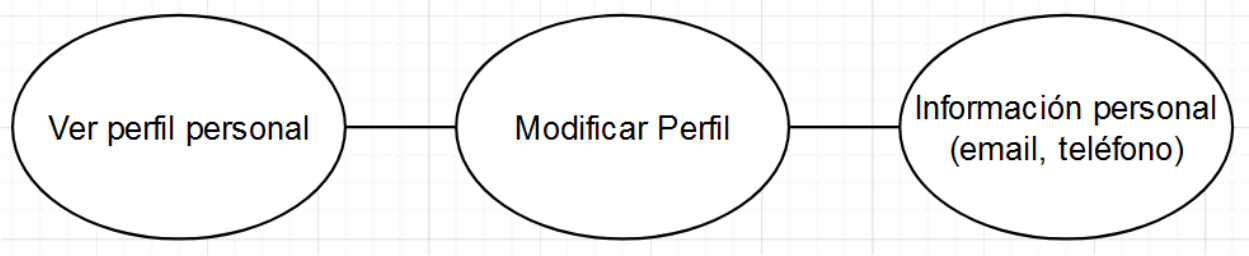
Riesgos:

- Una persona puede comunicarse directamente con el endpoint de retornar transacciones y ver sus transacciones sin autenticación.
- Una persona puede editar la comunicación con el endpoint de retornar transacciones para ver las transacciones de otros usuarios sin autenticación ni autorización.
- Una persona puede comunicarse directamente con el endpoint para obtener transacciones y ver sus transacciones sin autenticación.
- Una persona puede editar la comunicación con el endpoint de obtener transacciones para ver las transacciones de otros usuarios sin autenticación ni autorización.

Controles:

- Cerrar el acceso al endpoint de retornar transacciones a cualquier entidad y permitir que solo la aplicación React pueda comunicarse con la aplicación Node.
- Cerrar el acceso al endpoint de obtener transacciones a cualquier entidad y permitir que solo la aplicación Node pueda comunicarse con el servidor CGI.
- Aplicar medidas de verificación de identidad del usuario que realice la acción de ver sus transacciones para permitir ver sus transacciones o no dependiendo de lo que sea recibido en el servidor Node, en caso de modificación de la comunicación entre aplicaciones React y Node.

Cadena 8: Modificar perfil



Esta cadena de seguridad sigue la relación entre los componentes necesarios para modificar la información del perfil de un usuario. Para modificar el perfil, un usuario deberá entrar a su perfil personal primero y ahí verá un botón para modificarlo.
La cadena contempla el flujo desde el acceso al perfil personal, la comunicación con el backend y endpoint de la modificación de perfil, y la interacción con la base de datos para guardar los cambios.

Objetivos directos:

- Prevenir que personas no autenticadas/autorizadas ingresen al perfil personal.
- Prevenir que personas no autenticadas/autorizadas modifiquen los datos del perfil personal.
- Prevenir que se ingresen datos erróneos en los campos de modificación.

Objetivos indirectos:

- Prevenir que se acceda a la base de datos de forma directa.
- Prevenir que usuarios no autorizados hagan operaciones sobre la base de datos.
- Prevenir que usuarios no autenticados/autorizados accedan al endpoint de “Modificar perfil” directamente.

Vulnerabilidades:

- Se puede ingresar al perfil personal de un usuario.
- Se puede modificar el perfil de cualquier persona.
- Se pueden llegar y realizar acciones directamente en la base de datos.
- Se puede llegar directamente al endpoint de “Modificar perfil”.
- La entrada de datos es libre y sin ninguna limitante o regla.

Riesgos:

- Un usuario malicioso puede modificar el perfil de otra persona.
- Una persona no autorizada puede obtener y modificar datos confidenciales de la base de datos.
- Un usuario malicioso puede hacer inyección de SQL y tomar el control total.

Controles:

- Crear una representación autenticada del usuario y verificar contra ésta las acciones a las que está autorizado.
- Sólo permitir el acceso al perfil al usuario autenticado/autorizado.
- Utilizar expresiones regulares para validar la entrada del usuario, establecer un tamaño máximo y una lista de caracteres permitidos.
- En la base de datos, sólo permitir conexiones desde el backend de la aplicación. Esto se puede lograr con firewall.
- Utilizar un usuario con contraseña y permisos mínimos requeridos para la base de datos.
- La representación autenticada del usuario debe ser utilizada para acceder al endpoint de “Modificar perfil”, y sólo se podrá modificar el perfil asociado a esa representación.