



# NUCBANKING

**Trabajo integrador de verano**  
Programación FullStack 2020/2021

# Objetivos

- Integrar los conocimientos adquiridos durante la cursada del año 2020, tanto dentro del plan de estudio perteneciente al módulo de **desarrollo web** como al de **JavaScript**.
- Completar un desafío que requiere de tiempo y constancia.
- Perfeccionar los planteos implementados ante cada situación problemática.
- Interiorizar la sintaxis del lenguaje.
- Continuar el aprendizaje, más allá de la pausa lectiva.

# Criterios de evaluación

- El trabajo integrador de verano **no será solicitado** para alcanzar la certificación del módulo de JavaScript.
- El criterio de aceptación es **autoevaluativo**, es decir, cada alumn@ evaluará si ha alcanzado las funcionalidades solicitadas.
- Es importante cumplir tanto con la parte estética como con aquella que refiere a lo funcional, puesto que el trabajo del frontend developer implica maquetación, lógica en el cliente e integraciones con el servidor (este último a estudiarse en el corriente año).
- Las dudas se despejarán por medio de canales habituales, como Slack o Discord. De ser necesario, se determinará una clase de consulta.

# Explicación del desarrollo

La idea es desarrollar una plataforma que emule el sistema utilizado para los cajeros tradicionales. La misma contará de cuatro secciones generales:

- Consultas
- Servicios
- Depósitos
- Transferencias

En cada sección, existen sub-secciones con funcionalidades y validaciones específicas, que serán detalladas a continuación.

Se recomienda abordar el desarrollo de la aplicación tal como se presentan en este documento, ya que el nivel de complejidad es ascendente.

Sin más, les deseamos muchos éxitos.-

**#happyCoding**

# Login y registro de usuarios

Como en toda aplicación “cerrada” (aquellas que requieren que el usuario se valide para realizar acciones en la plataforma), es necesario crear la pantalla de login (para el acceso de usuarios registrados) y de creación de usuarios (para posibilitar el acceso de usuarios no registrados).

En este caso, comenzamos la “base de datos” en 0 (cero) usuarios registrados, por lo que necesitamos, en primera instancia, la pantalla de **registro de usuarios**.

(\*) Recordatorio:

*No es necesario que las pantallas sean estéticamente iguales a los ejemplos, pero sí es necesario que contengan las mismas funcionalidades y validaciones.*

Nombre

---

Apellido

---

Usuario

---

Contraseña

---

Registrar

Ya tengo cuenta, [ingresar](#).

La pantalla de **registro de usuario** deberá contener los campos:

- Nombre
- Apellido
- Usuario
- Contraseña

También deberá tener un botón **registrar** que dispare la acción de “crear usuario”, el cual será persistido en un array de usuarios en el Local Storage.

Por último, también debe contener una pequeña leyenda que redirija al usuario a la pantalla de login, en caso de que el mismo ya posea una cuenta creada.

Por favor, complete todos los campos.

Nombre

---

Apellido

---

Usuario

---

Contraseña

---

Registrar

Ya tengo cuenta, [ingresar](#).

**Todos los campos son datos obligatorios**, por lo que, si el usuario presiona “registrar” habiendo uno de ellos vacío, debemos presentar una alerta al usuario para que complete la acción de registro correctamente.

Cabe aclarar que, en caso de que el alerta se dispare, **no debemos persistir los datos del usuario**. Sólo lo haremos en caso de que todos los campos estén correctamente completados.

Nombre

Homero

Apellido

Simpson

Usuario

cosmefulanito

Contraseña

.....

Registrar

Ya tengo cuenta, [ingresar](#).

- **Nombre:**  
Refiere al nombre del usuario.
- **Apellido:**  
Refiere al apellido del usuario.
- **Usuario:**  
Refiere al nombre con el que el usuario ingresará a la plataforma.
- **Contraseña:**  
Refiere a la contraseña con la que el usuario ingresará a la plataforma.

Una vez completos, el botón registrar disparará una acción que:

- Persista los datos del mismo en el LS.
- Redirija al usuario a la pantalla de login, para que éste ingrese a la plataforma.



## Importante:

A la creación del usuario, es decir, al objeto **usuario**, debemos agregarle los siguientes atributos:

- **CBU:** que consiste en un UUID (busquen alguna función que provoque una expresión regular para que se cree automáticamente). Este CBU pasará a jugar el rol de ID de usuario.
- **Saldo:** que comenzará en 0 (cero). Hace referencia al dinero disponible en la cuenta de cada usuario; por default, iniciará en 0 (cero).
- **UsuariosGuardados:** que comenzará en array vacío. Hace referencia a los usuarios que cada usuario “guarda” para poder hacerles transferencias.
- **Servicios:** que comenzará en array vacío. Hace referencia a los servicios guardados por el usuario (veremos más en profundidad su utilidad más adelante).

Usuario

Contraseña

Ingresar

No tengo cuenta, [regístrame](#).

La pantalla de **login** deberá contener los campos:

- Usuario
- Contraseña

También deberá tener un botón **ingresar** que dispare la acción de “login”, la cual deberá **validar si el usuario ingresado existe en el array de usuarios persistido en el LS** y, en caso de existir, deberá **validar si la contraseña es correcta**.

Por último, también debe contener una pequeña leyenda que redirija al usuario a la pantalla de registro de usuario, en caso de que el mismo no tenga una cuenta creada.

Por favor, complete todos los campos.

Usuario

---

Contraseña

---

Ingresar

No tengo cuenta, [regístrame](#).

**Todos los campos son datos obligatorios**, por lo que, si el usuario presiona “ingresar” habiendo uno de ellos vacío, debemos presentar una alerta al usuario para que complete la acción de ingreso correctamente.

Cabe aclarar que, en caso de que el alerta se dispare, **no debemos realizar la validación del usuario**, puesto que no tendremos la información completa. Sólo lo haremos en caso de que todos los campos estén correctamente completados.

Datos incorrectos.

Usuario

carobsts

Contraseña

\*\*\*\*\*

Ingresar

No tengo cuenta, [registrarme](#).

También debemos validar en caso de que **los campos estén completos pero la información ingresada no sea correcta** (es decir, no corresponda con aquella que tenemos persistida en nuestro array de usuarios del LS).

Usuario

carobsts

Contraseña

\*\*\*\*\*

Ingresar

No tengo cuenta, [regístrame](#).

En caso de que los **campos estén completos y la data sea correcta**, el botón “ingresar” deberá redirigir al usuario a la home de nuestra aplicación.

### Data importante:

Una vez el usuario y la contraseña ingresados por el usuario *matcheen* con un usuario y la correspondiente contraseña de un user que tenemos almacenado en nuestro array de usuarios, **será bastante útil persistir a este usuario (individualmente) en nuestra LS.**

De esta manera, haremos todas las validaciones con este usuario individual, sin necesidad de buscarlo en el array, una vez estemos autenticados.

# Home

La home es la pantalla donde se despliegan todas las acciones que puede realizar un usuario, una vez esté autenticado.

La misma es sumamente importante, ya que presenta un pantallazo de todas las secciones de nuestra aplicación.



# NUCBANKING

¡Bienvenid@, Carolina!

Consultas

Servicios

Depósitos

Transferencias

En la misma tenemos:

- Las cuatro acciones principales de la aplicación:
  - (a) Consultas,
  - (b) Servicios,
  - (c) Depósitos y
  - (d) Transferencias.

Iremos a fondo en cada una de ellas, más adelante.

- Un **botón de logout**: el mismo se encarga de eliminar ese usuario individual que hemos persistido en el momento en que ingresamos en la plataforma. Al accionarlo, el mismo debe: eliminar esa persistencia y redirigir al usuario a la pantalla del login.
- El logo y título de la empresa/organización de la app.
- Un título de bienvenida **dinámico**: en el mismo, saluda con el nombre del usuario.



# Consultas

La pantalla de consultas será accedida desde el botón “consultas” de la home.

La misma cuenta de dos sub-secciones:

- Saldo
- CBU

Ambos son dos botones que, al presionarlos, deben redirigir a la pantalla correspondiente.

También contiene un botón “atrás” en el margen superior izquierdo, que redirige a la pantalla “home”.



## Consultas

Saldo

CBU

## Saldo:

La misma consiste en una pantalla donde se le muestra al usuario su **saldo actual**, es decir, el dinero disponible en su cuenta.

Aquellos usuarios recién logueados, deben visualizar un saldo de \$0 (cero).-

Luego, otras funcionalidades serán las encargadas de incrementar o decrementar su valor.

Este valor debe ser **dinámico** y manipulado desde JS.

Esta pantalla debe tener un botón "atrás" que redirija al usuario a la pantalla "consultas".



## CBU:

La misma consiste en una pantalla donde se le muestra al usuario su **CBU**, es decir, aquel valor que se **autocreó** en el momento de su registro.

Este valor debe ser **dinámico** y manipulado desde JS.

Esta pantalla debe tener un botón "atrás" que redirija al usuario a la pantalla "consultas".

CBU

Su CBU es:

034aa372-1276-41c8-a74c-f48eeee380fb

# Depósitos

La pantalla de depósitos será accedida desde el botón “depósitos” de la home.

La misma cuenta de dos sub-secciones:

- Cuenta propia
- Cuenta terceros

Ambos son dos botones que, al presionarlos, deben redirigir a la pantalla correspondiente.

También contiene un botón “atrás” en el margen superior izquierdo, que redirige a la pantalla “home”.



## Depósitos

Cuenta propia

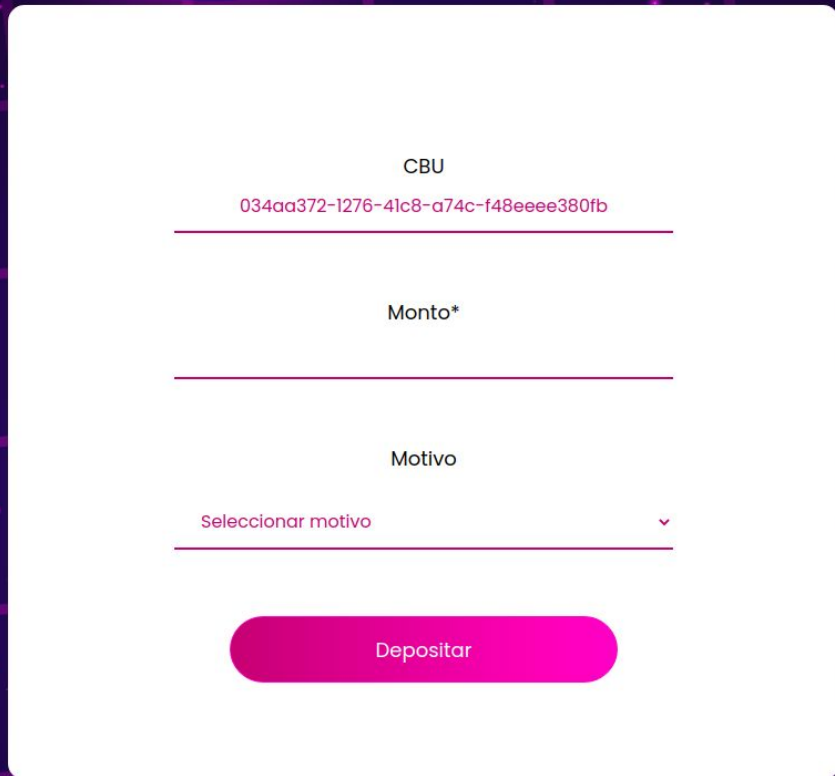
Cuenta terceros

## Cuenta propia:

La misma consiste en una pantalla donde el usuario podrá realizar depósitos a su propia cuenta (esto quiere decir que por cada depósito, **incrementará su saldo**).

Tiene tres campos:

- **CBU:** que viene pre-seteada, esto quiere decir, desde el LS, ya que sale de los datos que tenemos del usuario autenticado. Este campo **no se puede modificar**, por lo que el input tiene un atributo readonly.
- **Monto:** con valor numérico. Corresponde al valor que se depositará. Este es un **campo obligatorio**.



The screenshot shows a deposit form with the following elements:

- CBU:** A text input field containing the value "034aa372-1276-41c8-a74c-f48eeee380fb".
- Monto\*:** A text input field for the deposit amount, marked with an asterisk to indicate it is required.
- Motivo:** A dropdown menu with the placeholder text "Seleccionar motivo" and a downward arrow icon.
- Depositar:** A large, rounded, magenta button at the bottom of the form.



- **Motivo:** refiere al motivo del depósito. No es un campo obligatorio; de hecho, no hace falta que persistan esta información.

También contamos con un botón “depositar” que será el encargado de disparar la acción del depósito.

Por último, debe contar con un botón “atrás” que redirija a la pantalla “depósitos”.

En caso de que el campo obligatorio “monto” no esté completo, **debemos lanzar una alerta que notifique al usuario.**

Por favor, ingresar el monto.

CBU

034aa372-1276-41c8-a74c-f48eeee380fb

Monto\*

Motivo

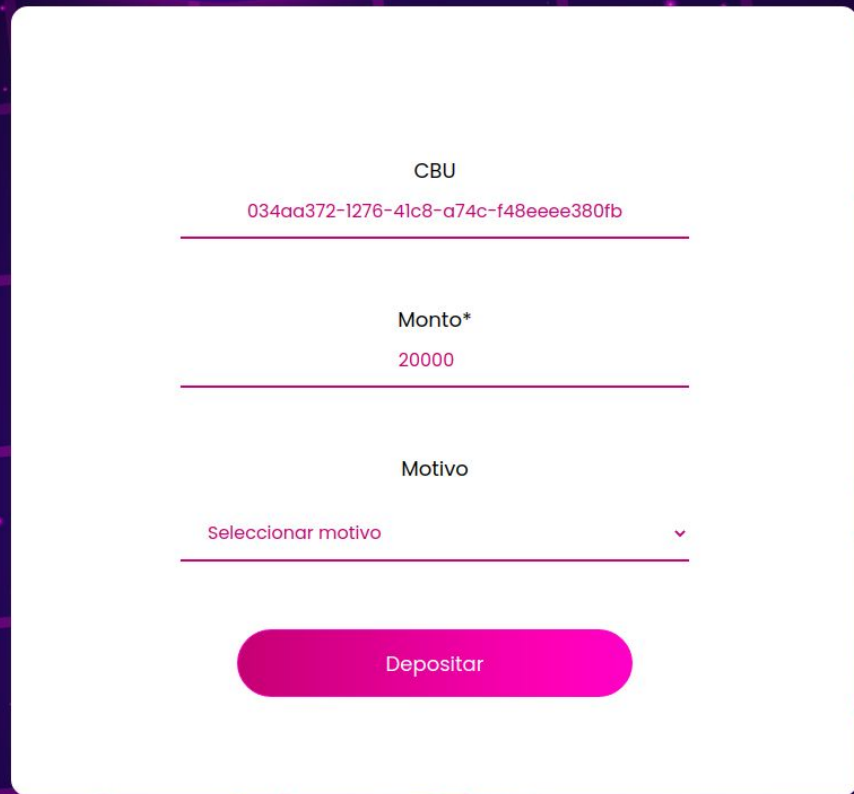
Seleccionar motivo

Depositar



En caso de que el campo obligatorio “monto” esté completo, el botón “depositar” debe:

- Sumar el monto al saldo actual del usuario.
- Redirigir al usuario a la pantalla de “depósitos”.



CBU

034aa372-1276-41c8-a74c-f48eeee380fb

Monto\*

20000

Motivo

Seleccionar motivo

Depositar

En caso de que todo salga bien, **el saldo actual debería mostrar el monto depositado.**

**Saldo**

Su saldo es de:

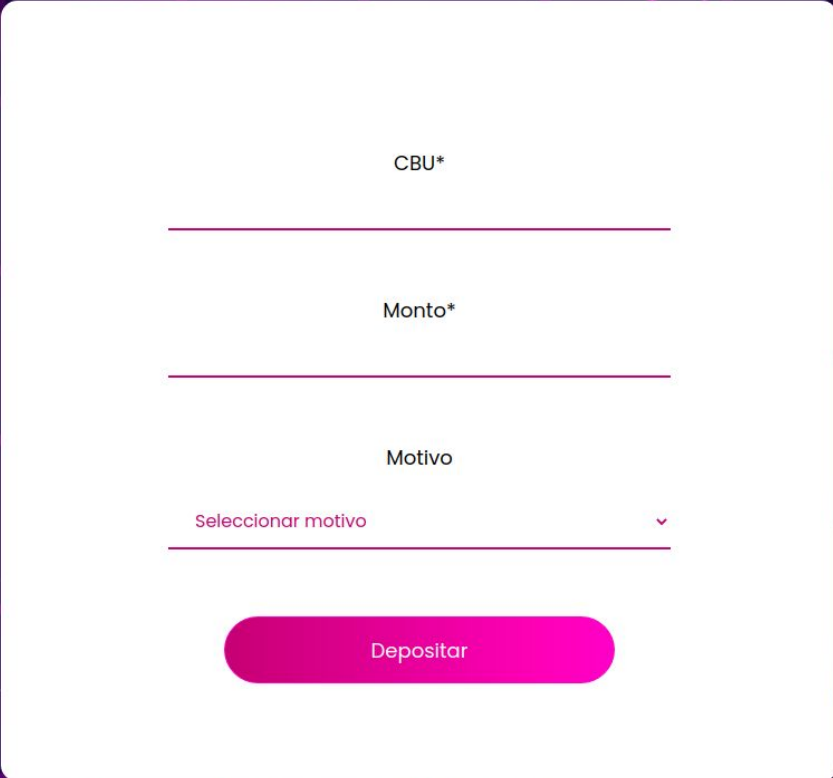
**\$ 20000**

## Cuenta terceros:

La misma consiste en una pantalla donde el usuario podrá realizar depósitos a la cuenta de un tercero (esto quiere decir que por cada depósito, **decrementará su saldo**).

Tiene tres campos:

- **CBU:** campo obligatorio. Este debe existir en la base de datos, es decir, pertenecer a un usuario creado y persistido en el LS.
- **Monto:** con valor numérico. Corresponde al valor que se depositará. Este es un **campo obligatorio**.

El formulario se encuentra en un recuadro blanco con esquinas redondeadas sobre un fondo espacial. Incluye tres campos de entrada con líneas horizontales azules, un menú desplegable y un botón de acción.

Formulario de depósito a terceros:

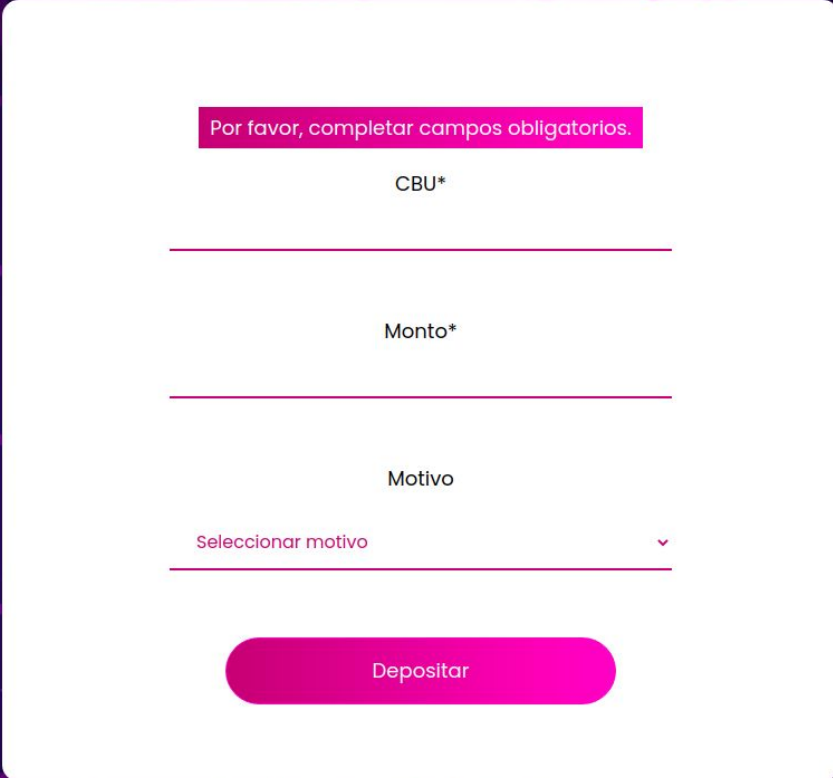
- CBU\*
- Monto\*
- Motivo
- Selecccionar motivo
- Depositar

- **Motivo:** refiere al motivo del depósito. No es un campo obligatorio; de hecho, no hace falta que persistan esta información.

También contamos con un botón “depositar” que será el encargado de disparar la acción del depósito.

Por último, debe contar con un botón “atrás” que redirija a la pantalla “depósitos”.

En caso de que el campo obligatorio “monto” y “cbu” no estén completos, **debemos lanzar una alerta que notifique al usuario.**



Por favor, completar campos obligatorios.

CBU\*

Monto\*

Motivo

Seleccionar motivo

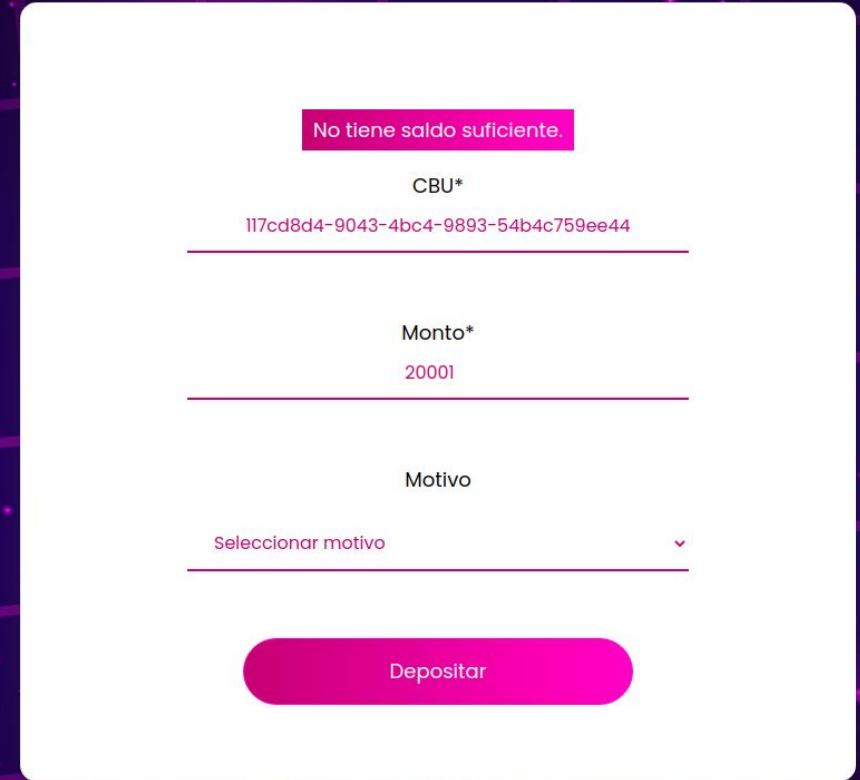
Depositar

Si los campos obligatorios “monto” y “cbu” están completos, el botón “depositar” debe:

- Validar que el monto que se quiere depositar a un tercero, **esté dentro del rango de saldo disponible**.

En caso de que no sea así, debemos lanzar una alerta que notifique al usuario.

- Validar que el CBU ingresado pertenezca a un usuario en nuestro LS.



No tiene saldo suficiente.

CBU\*

117cd8d4-9043-4bc4-9893-54b4c759ee44

---

Monto\*

20001

---

Motivo

Seleccionar motivo ▼

---

Depositar

Una vez el usuario ingrese un CBU existente y un monto que se encuentra dentro del rango de saldo disponible, el botón registrar debe:

- Restar el monto depositado al saldo actual del usuario autenticado.
- Sumar el monto depositado al saldo actual del usuario a quien pertenece el CBU ingresado.
- Redirigir a la pantalla “depósitos”.



CBU\*

117cd8d4-9043-4bc4-9893-54b4c759ee44

Monto\*

10000

Motivo

Seleccionar motivo

Depositar

En caso de que todo salga bien, el saldo actual debería mostrar un nuevo valor, **producto de la resta del depósito realizado a un tercero y el saldo anterior.**

Saldo

Su saldo es de:

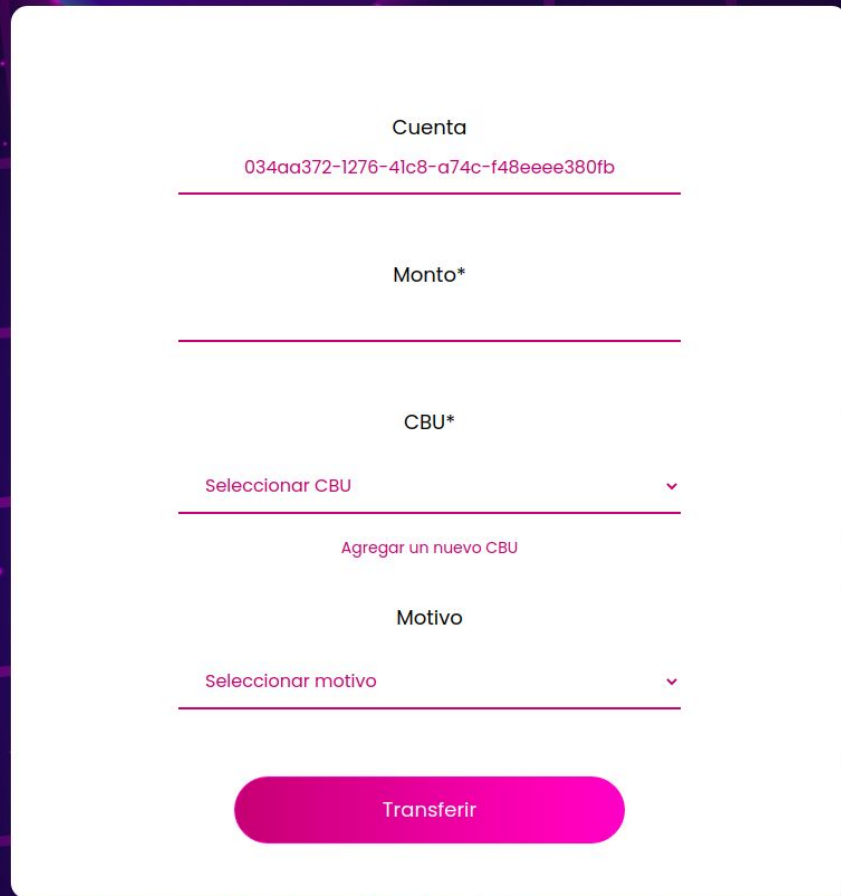
**\$ 10000**

# Transferencias

La pantalla de transferencias será accedida desde el botón “transferencias” de la home.

Se trata de una sola pantalla, que tiene cuatro campos:

- **Cuenta:** campo no-modificable. Esta información debe provenir de la data que tenemos del usuario autenticado (cuenta desde la cual se realiza la transferencia).
- **Monto:** campo obligatorio. Valor que se va a transferir.
- **CBU:** menú desplegable con las opciones a las cuales se puede transferir. Estas opciones provienen de las cuentas guardadas del usuario. Campo obligatorio.

A mockup of a mobile application screen for making transfers. The screen has a white background with a purple border. It contains four input fields: 'Cuenta' with a fixed value, 'Monto\*' (required), 'CBU\*' (dropdown menu), and 'Motivo' (dropdown menu). A large purple 'Transferir' button is at the bottom.

Cuenta

034aa372-1276-41c8-a74c-f48eeee380fb

Monto\*

CBU\*

Seleccionar CBU

Agregar un nuevo CBU

Motivo

Seleccionar motivo

Transferir



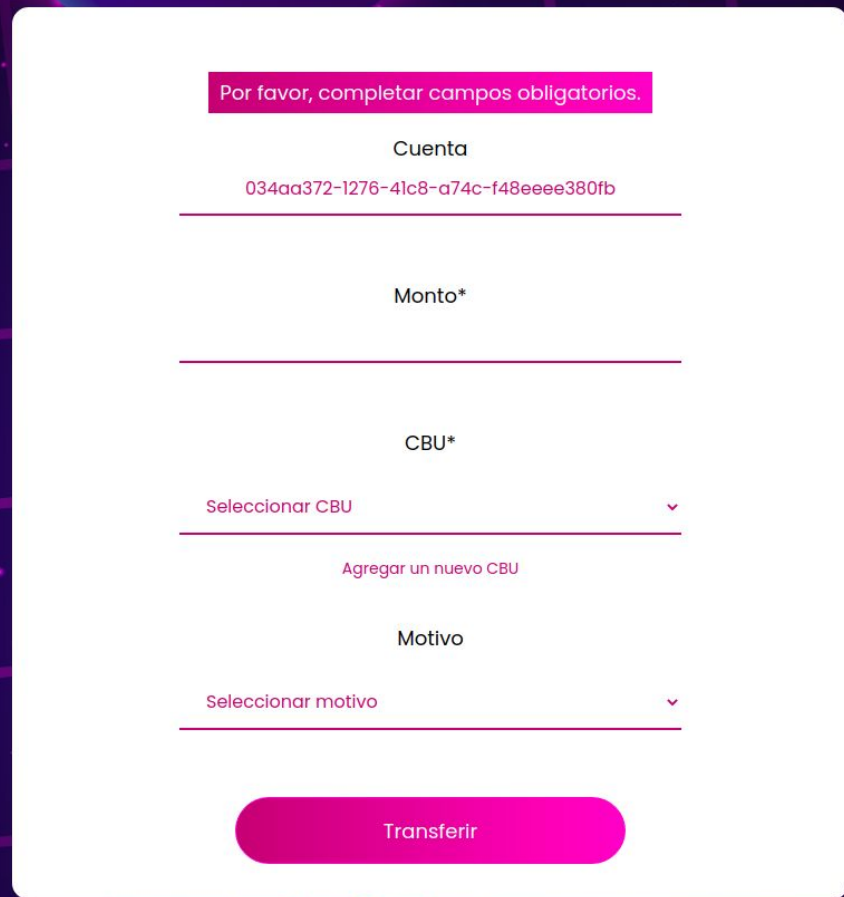
- **Motivo:** refiere al motivo de la transferencia. No es un campo obligatorio; de hecho, no hace falta que persistan esta información.

También contamos con un botón “transferir” que será el encargado de disparar la acción de la transferencia.

Por último, debe contar con un botón “atrás” que redirija a la pantalla “home”.

En caso de que los campos obligatorios “monto” y “cbu” no estén completos, **debemos lanzar una alerta que notifique al usuario.**

Como podemos ver, debajo del campo “CBU” hay una leyenda que dice: “agregar un nuevo CBU”. Hablaremos de ello a continuación...



Por favor, completar campos obligatorios.

Cuenta

034aa372-1276-41c8-a74c-f48eeee380fb

---

Monto\*

---

CBU\*

Seleccionar CBU

---

Agregar un nuevo CBU

Motivo

Seleccionar motivo

---

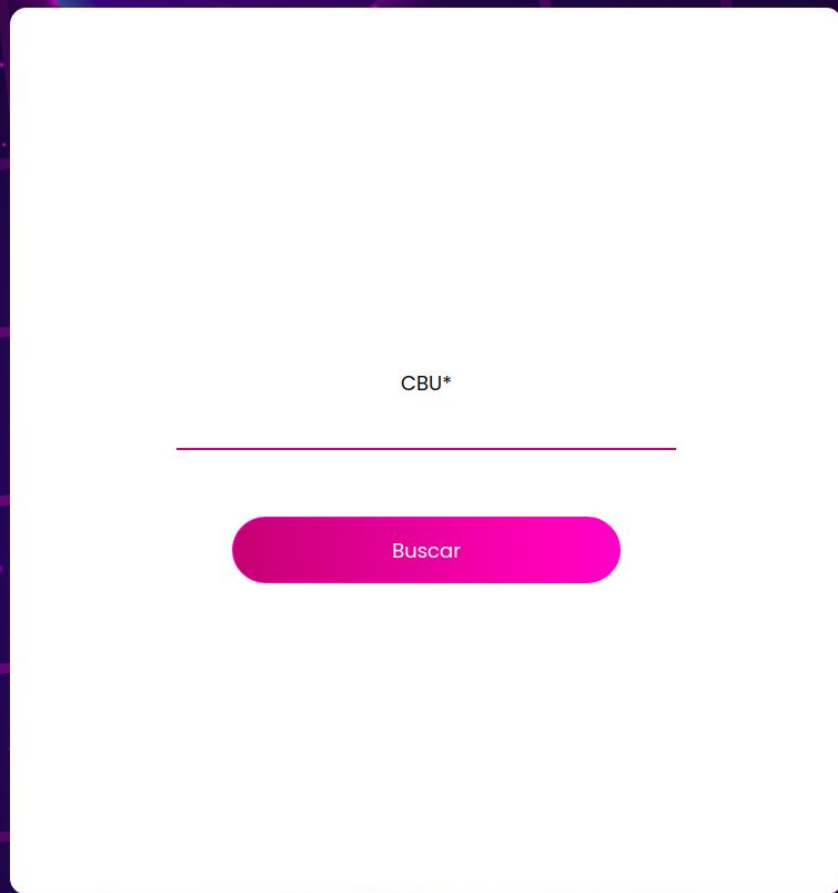
Transferir

“Agregar un nuevo CBU” se trata de un botón que redirige a otra pantalla donde podremos ver un campo “CBU” con un botón “Buscar”.

Esta pantalla será la encargada de guardar cuentas a las cuales el usuario autenticado quiere realizar transferencias (lo que sería “agendar cbus”).

Para ello, primero necesitamos **ingresar un CBU existente**.

En caso de que el valor ingresado al campo “CBU” no corresponda con un CBU existente, se le debe arrojar una alerta al usuario.



CBU\*

---

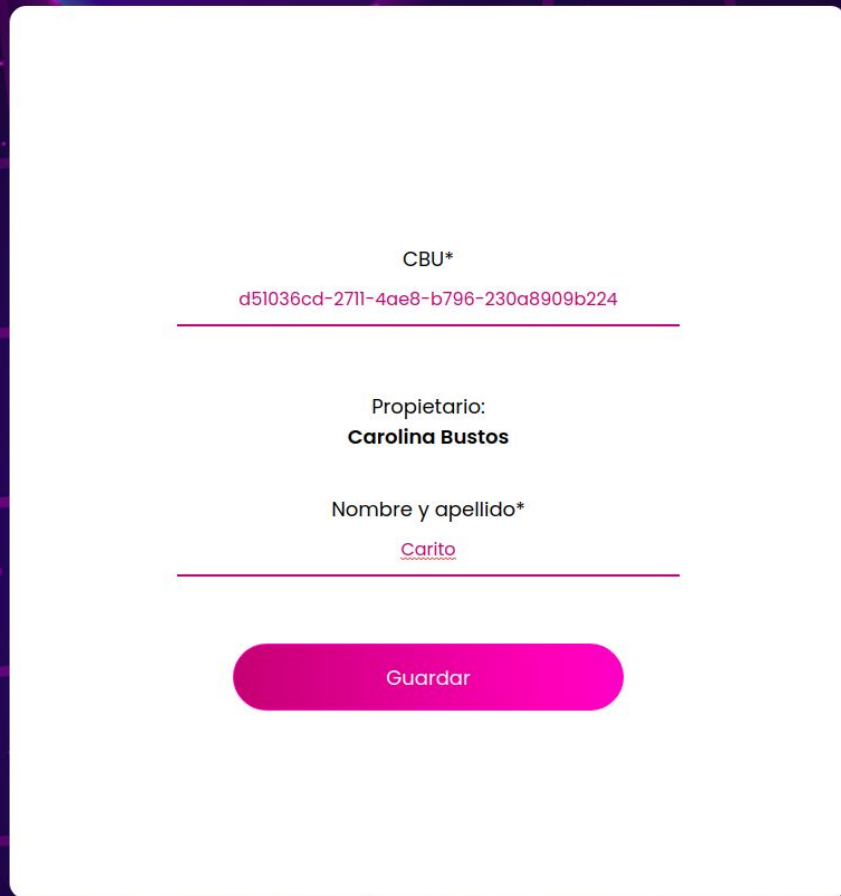
Buscar

En caso de que el valor ingresado corresponda efectivamente a un CBU existente, debemos traernos los datos del usuario que corresponde a ese CBU y mostrarlo en pantalla.

Junto a ello, aparecerá un nuevo campo obligatorio para ingresar un nombre con el cual será identificado este “contacto”.

El botón “guardar” deberá:

- Agregar esta cuenta guardada en el array de “cuentas guardadas” del usuario autenticado.
- Redirigir al usuario a la pantalla “transferencias”.

El formulario se encuentra en un recuadro blanco con bordes redondeados sobre un fondo de pantalla con un patrón de cuadrícula y círculos de colores morado y azul. El formulario contiene los siguientes elementos: un campo de texto con el label "CBU\*" y el valor "d51036cd-2711-4ae8-b796-230a8909b224"; un campo de texto con el label "Propietario:" y el valor "Carolina Bustos"; un campo de texto con el label "Nombre y apellido\*" y el valor "Carito"; y un botón de tipo "submit" con el texto "Guardar".

CBU\*

d51036cd-2711-4ae8-b796-230a8909b224

Propietario:

**Carolina Bustos**

Nombre y apellido\*

Carito

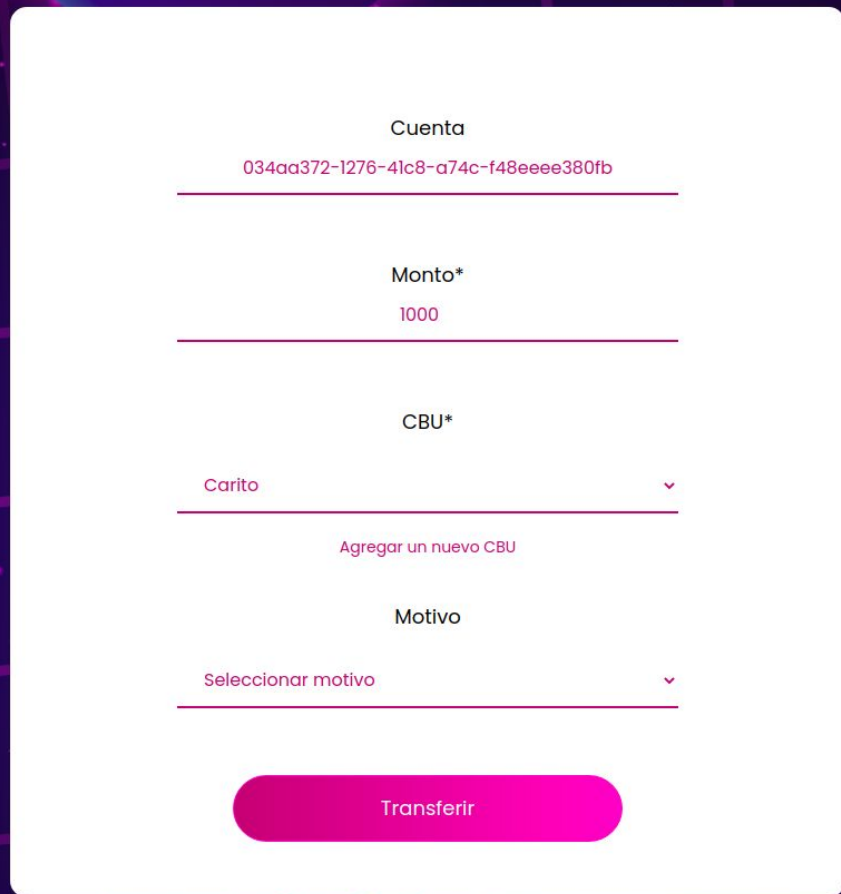
Guardar

El array de “cuentas guardadas” debe mostrarse en las opciones del campo “CBU”.

Tal como se ve en la imagen, debemos poder visualizar la cuenta guardada con el nombre que hemos ingresado para “agendarla”.

Una vez que todos los campos se encuentren completos, el botón transferir debe:

- Validar si el monto ingresado es inferior o igual al saldo disponible.
- Restar el valor transferido del saldo actual del usuario autenticado.
- Sumar el valor transferido al saldo actual del usuario al que corresponde la cuenta que hemos guardado anteriormente.



Formulario de transferencia de dinero:

- Cuenta:** 034aa372-1276-41c8-a74c-f48eeee380fb
- Monto\*:** 1000
- CBU\*:** Carito
- Agregar un nuevo CBU** (enlace)
- Motivo:** Seleccionar motivo
- Transferir** (botón)

En caso de que todo salga bien, el saldo actual debería mostrar un nuevo valor, **producto de la resta de la transferencia realizada y el saldo anterior.**

Saldo

Su saldo es de:

**\$ 9000**

# Servicios

La pantalla de servicios será accedida desde el botón “servicios” de la home.

Tiene cuatro sub-secciones:

- Adherir servicios
- Desvincular servicios
- Pagar servicios
- Mis comprobantes

Todos son botones que, al presionarlos, deben redirigir a la pantalla correspondiente.

También contiene un botón “atrás” en el margen superior izquierdo, que redirige a la pantalla “home”.



## Servicios

Adherir servicios

Desvincular servicios

Pagar servicios

Mis comprobantes

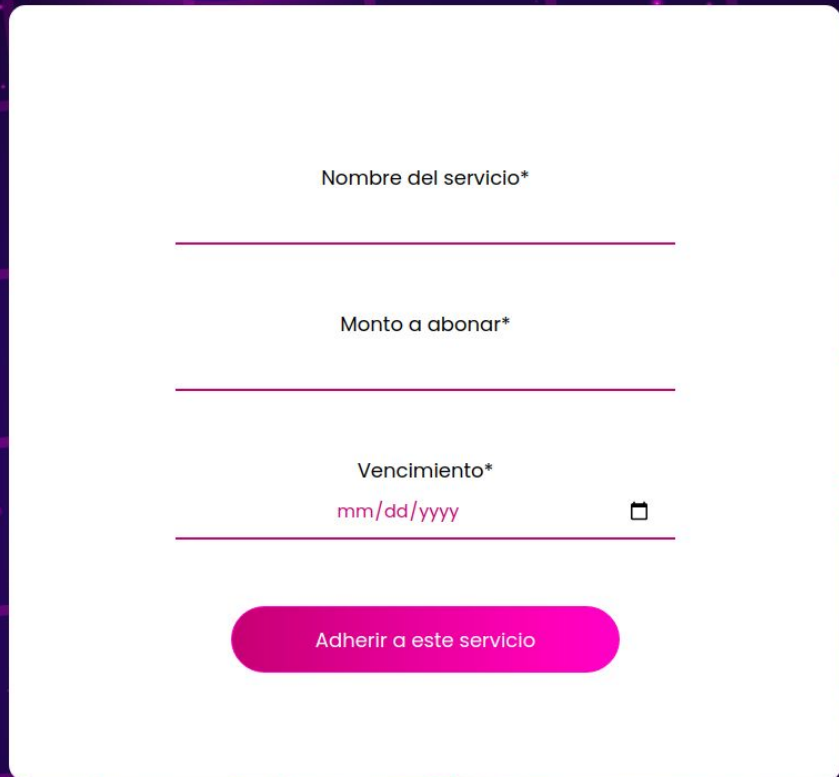
## Adherir servicios:

La misma consiste en una pantalla donde el usuario podrá **adherir a un servicio**.

Tiene tres campos:


- **Nombre del servicio:** campo obligatorio. Consiste en el nombre que damos al servicio para identificarlo.
- **Monto a abonar:** campo obligatorio. Se trata de lo que pagaremos por el servicio.
- **Vencimiento:** campo obligatorio. Se trata de la fecha en la cual vencerá el servicio.

También tenemos un botón “adherir a este servicio” y un botón “atrás” que redirige a la pantalla “servicios”.

El formulario se encuentra en un recuadro blanco con bordes redondeados sobre un fondo de espacio con estrellas y planetas. Tiene tres campos de entrada con líneas horizontales y etiquetas con asterisco. El tercer campo incluye un ícono de calendario. Debajo de los campos hay un botón redondo de color magenta.

Nombre del servicio\*

Monto a abonar\*

Vencimiento\*  
mm/dd/yyyy 

Adherir a este servicio



En caso de que el usuario presione el botón “adherir a este servicio”, con los campos en blanco, deberemos lanzar una alerta al usuario notificando que complete los mismos.

Por favor, completar los campos obligatorios.

Nombre del servicio\*

---

Monto a abonar\*

---

Vencimiento\*

mm/dd/yyyy



---

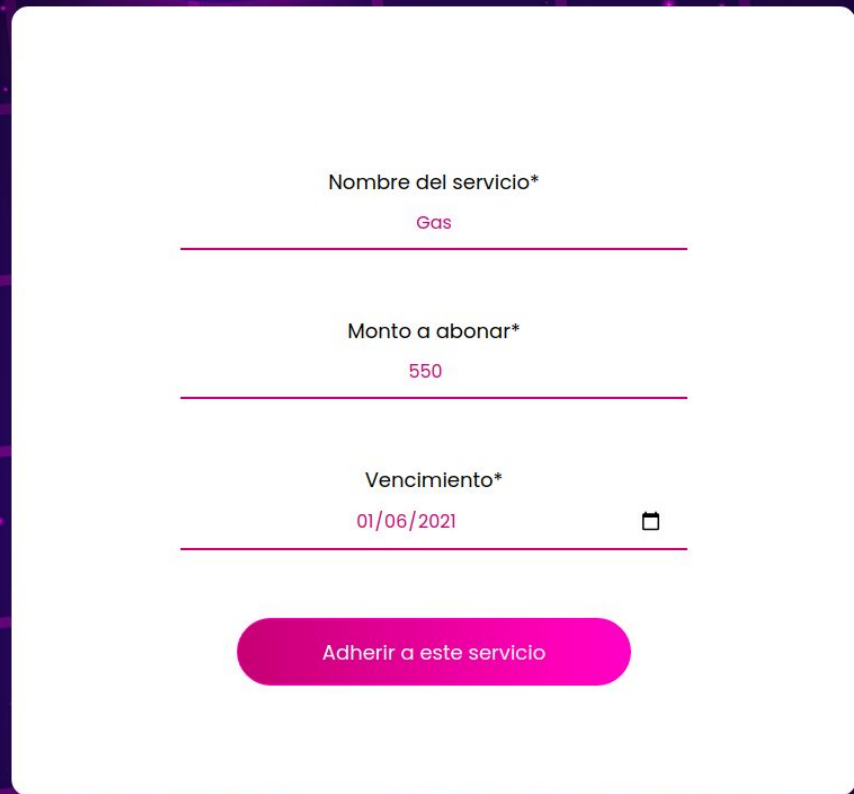
Adherir a este servicio

Una vez que el usuario complete todos los campos obligatorios, el botón “adherir a este servicio” deberá:

- Agregar el servicio ingresado al array de servicios que es parte de los atributos del usuario autenticado.
- Redirigir a la pantalla “servicios”.

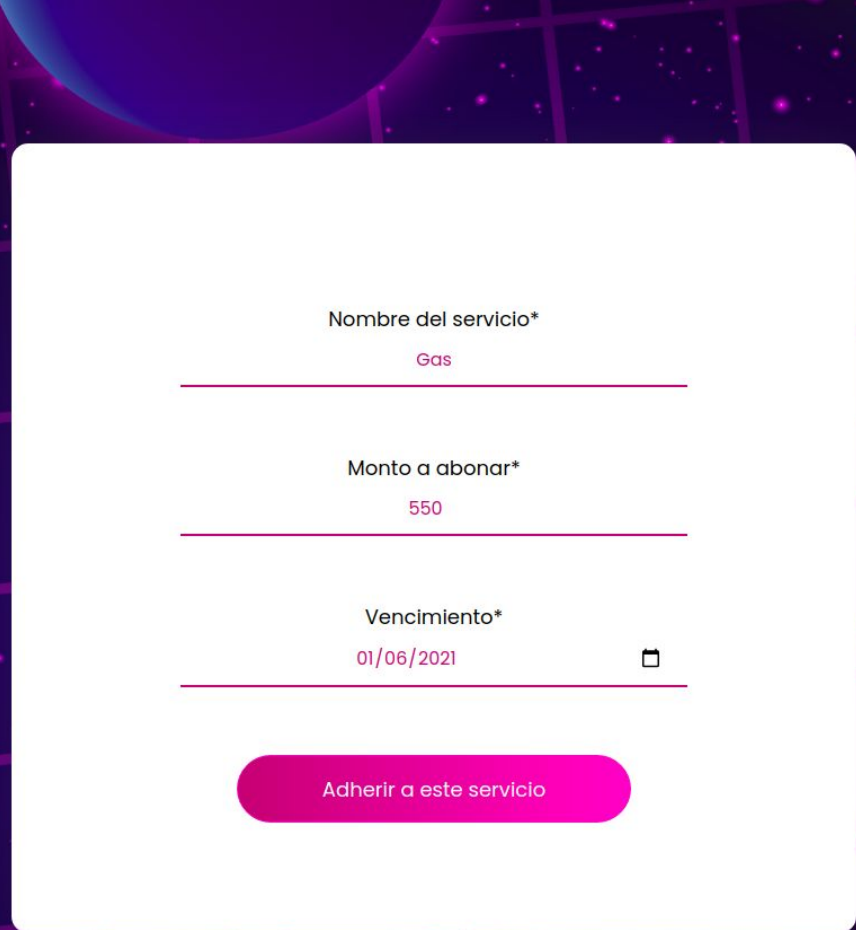
### Importante:

Al objeto **servicio**, debe agregarse un atributo por default que se llamará “pagado” y tendrá un valor “false”. Este servirá para identificar aquellos servicios pagados de aquellos impagos.

El formulario se encuentra en un recuadro blanco con bordes redondeados sobre un fondo espacial. Contiene tres campos de texto con etiquetas grises y valores de ejemplo en color magenta. El primer campo es 'Nombre del servicio\*' con el valor 'Gas'. El segundo es 'Monto a abonar\*' con el valor '550'. El tercer es 'Vencimiento\*' con el valor '01/06/2021' y un ícono de calendario. Debajo de los campos hay un botón magenta con el texto 'Adherir a este servicio'.

Una vez que el usuario complete todos los campos obligatorios, el botón “adherir a este servicio” deberá:

- Agregar el servicio ingresado al array de servicios que es parte de los atributos del usuario autenticado.
- Redirigir a la pantalla “servicios”.
- Validar que no exista un servicio con el mismo nombre en el array de servicios del usuario autenticado .



The image shows a white rectangular form with rounded corners, set against a dark purple background with a subtle grid and star patterns. The form contains three input fields, each with a label, a value, and a horizontal line below it. The first field is labeled 'Nombre del servicio\*' and contains the text 'Gas'. The second field is labeled 'Monto a abonar\*' and contains the text '550'. The third field is labeled 'Vencimiento\*' and contains the date '01/06/2021', followed by a small calendar icon. Below these fields is a large, rounded, magenta button with the text 'Adherir a este servicio' in white.

Nombre del servicio\*

Gas

Monto a abonar\*

550

Vencimiento\*

01/06/2021

Adherir a este servicio



Ya existe un servicio con ese nombre.

Nombre del servicio\*

Gas

Monto a abonar\*

550

Vencimiento\*

01/13/2021



Adherir a este servicio

## Pagar servicios:

La misma consiste en una pantalla donde el usuario podrá **pagar un servicio**.

Se trata de una pantalla donde se visualizan cards con cada uno de los servicios cuyo atributo "pagado" esté en "false".

Cada card tiene:

- Nombre del servicio.
- Monto a pagar por el servicio.
- Fecha de vencimiento.
- Una alerta que notifica al usuario si el servicio está vencido o no.
- Un botón "pagar".



Gas

\$ 550

Vencimiento: 5/1/2021

Al día

Pagar

Luz

\$ 450

Vencimiento: 31/12/2020

Vencido

Pagar

Rentas

\$ 660

Vencimiento: 29/1/2021

Al día

Pagar

También tiene un botón “atrás” que redirige al usuario a la pantalla “servicios”.

Al presionar el botón “pagar”, este debe:

- Validar que el monto a pagar es inferior al saldo disponible.

En caso de que el monto a pagar sea inferior al saldo disponible, procedemos a permitir la transacción de pago, que:

- Actualizará el atributo “pagado” del servicio, de false a true.
- Agregará un nuevo atributo “fecha de pago” al objeto servicio, con la fecha del día en que se realizó el pago.
- Restará el monto a pagar por el servicio, del saldo actual del usuario autenticado.
- Redirigirá a la pantalla “servicios”.

**Importante:**

- La pantalla del “saldo” debe mostrar el valor del saldo modificado.
- La pantalla de “pagar servicios” no debe mostrar el servicio abonado anteriormente.



## Mis comprobantes:

La misma consiste en una pantalla donde el usuario podrá **ver los comprobantes de los servicios pagados**.

Se trata de una pantalla donde se visualizan cards con cada uno de los servicios cuyo atributo “pagado” esté en “true”.

Cada card tiene:

- Nombre del servicio.
- Monto pagado por el servicio.
- Fecha de pago del mismo.

Además, contiene un botón “atrás” que redirige al usuario hacia la pantalla “servicios”.



## Desvincular servicios:

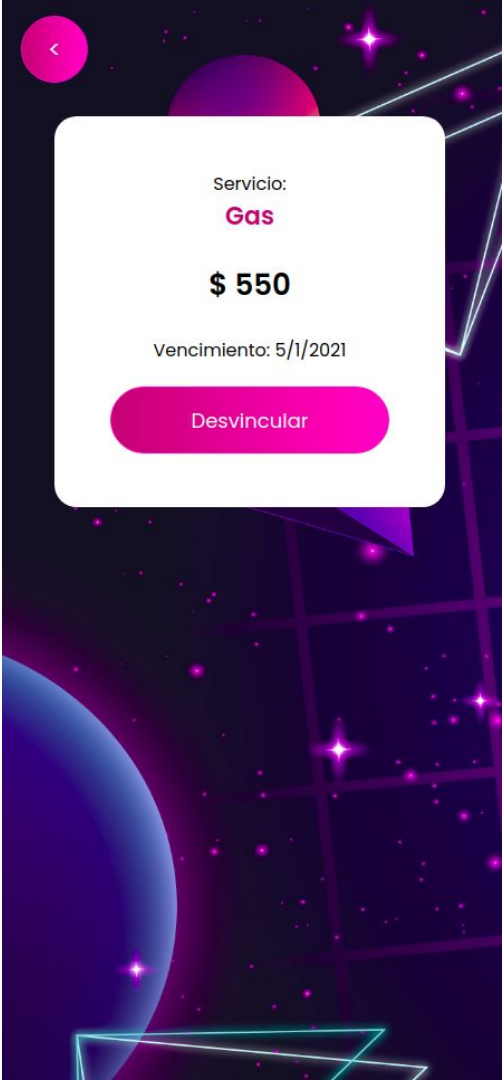
La misma consiste en una pantalla donde el usuario podrá **desvincularse de servicios previamente adheridos**.

Se trata de una pantalla donde se visualizan cards con cada uno de los servicios cuyo atributo “pagado” esté en “true”, ya que **no se puede desvincular de un servicio no pagado**.

Cada card tiene:

- Nombre del servicio.
- Monto pagado por el servicio.
- Fecha de vencimiento del mismo.
- Botón “desvincular”.

Además, contiene un botón “atrás” que redirige al usuario hacia la pantalla “servicios”.



El botón “desvincular” debe:

- Eliminar el servicio seleccionado del array de servicios que contiene el usuario autenticado.

# ¿Lograste cumplir el desafío?

Ahora podés deployarlo en vercel y ya tenés un trabajo muy completo que te va a servir como portfolio para futuras entrevistas laborales.

¡Felicitaciones! Y nos vemos pronto en clases.

Equipo de NUCBA.-

