Retiro 10% AFP

Diego Espinoza

06 de Dociembre de 2021

Base de datos y programación web

**Problema**

A día de hoy se llevan 4 retiros de fondos AFP aprobados, lo más probable es a lo largo del tiempo se lleguen a acuerdos legales para que existan aún más. Por esto se necesita crear una página web que se mantenga en el tiempo en la cual se pueda hacer el trámite independientemente de la AFP a la que se pertenezca.

**Solución**

Este proyecto está enfocado en realizar una página web para que las personas afiliadas al retiro del 10% de los fondos AFP puedan realizar este trámite. El sitio contara con soporte para las 7 administradoras de fondos de pensiones, ósea que los usuarios pueden hacer uso de las características del software independientemente de la AFP a la que pertenezcan.

El usuario (que ya está registrado dentro de la base de datos) ingresa RUN y contraseña de su cuenta, el sistema verificara los datos ingresados, ya verificado se preguntara cual es el porcentaje que desea retirar [1,10]% para finalmente escoger si desea hacer retiro en una cuenta de ahorro o bancaria.

Luego del retiro se debe descontar lo solicitado de su saldo y entregar una “boleta” donde se presente el estado actual de su cuenta.

**Requerimientos**

**Funcionales**

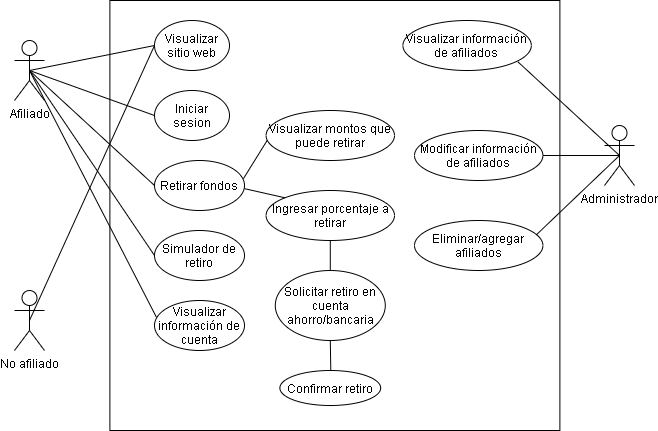
1. El sistema debe permitir informar sobre los montos que puede retirar el usuario.
2. El sistema debe presentar pestañas/viñetas de verificación de entrada de datos y ejecución de operaciones sobre su cuenta.
3. Se debe brindar la opción al usuario de elegir entre un rango de [1,10]%de sus fondos para retirar.
4. El sistema presentara un logeo solo para el administrador con el objetivo de obtener información de los afiliados.
5. El sistema presentará un logeo para afiliados, una vez iniciada sesión podrá acceder a las características del software.
6. El sistema deberá reconocer automáticamente a que AFP pertenece el afiliado una vez que ya haya iniciado sesión.

**No funcionales**

1. Los usuarios tendrán solamente una cuenta de capitalización individual por motivos de reducir la complejidad del desarrollo del software.
2. La aplicación será compatible con diferentes resoluciones de pantalla, haciendo que su contenido sea legible, imágenes, texto, etc.

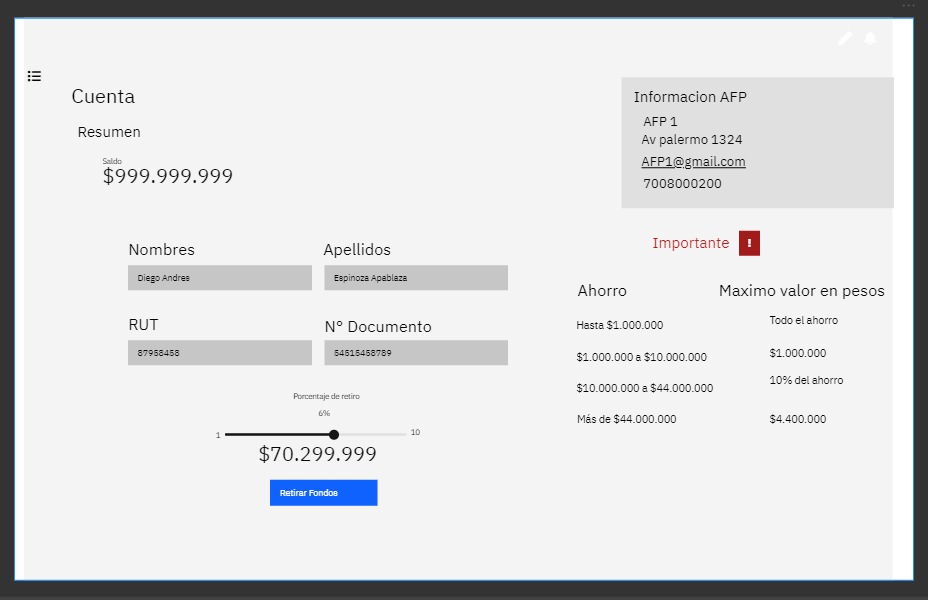
3. El sistema debe presentar una página web responsiva.

**Casos de uso**



**MOCKUPS INICIALES**

**FRONTEND**

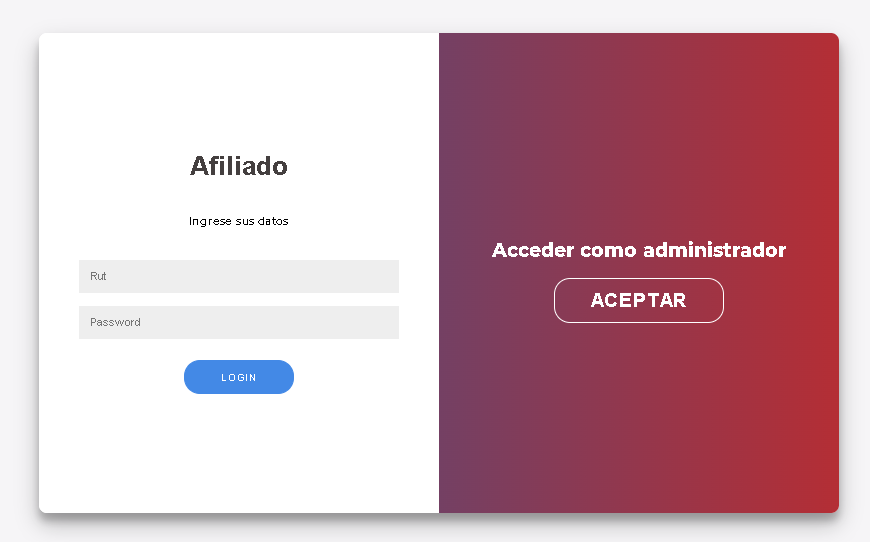


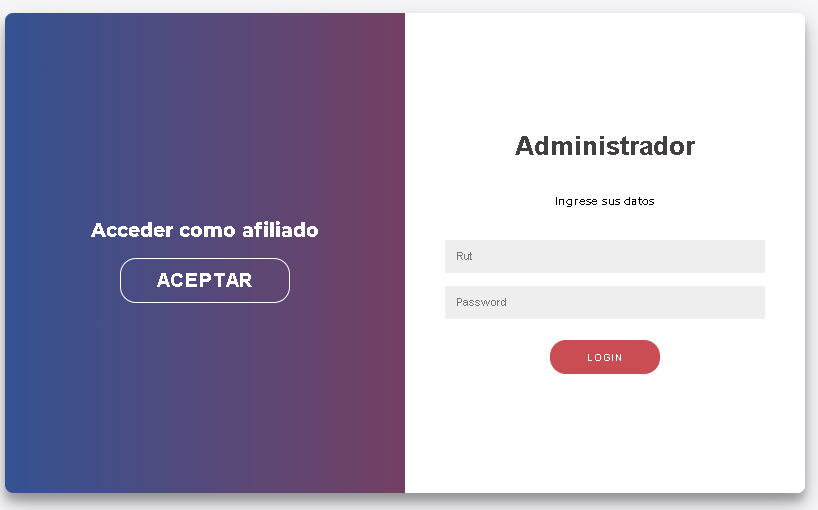
**BACKEND**



**Mockups Finales**

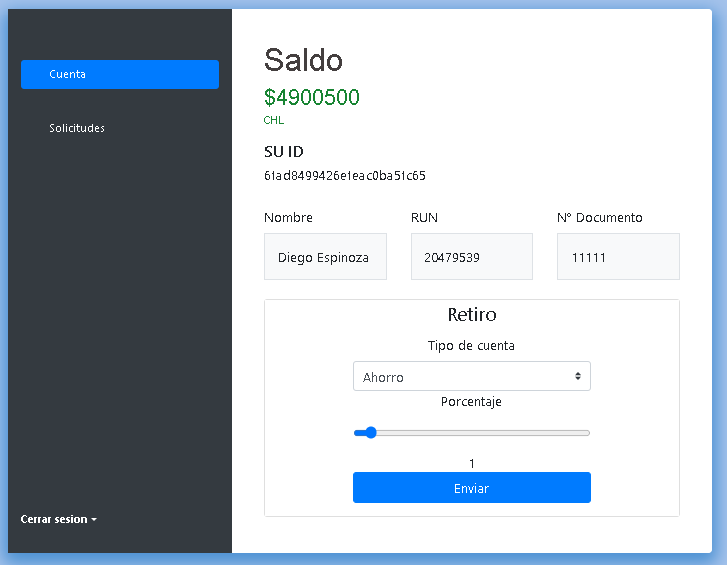
**Front**





Se implemento una única vista para el login donde se pueda acceder como administrador o afiliado, esto con el motivo de manejar una menor cantidad de vistas, anteriormente se estaba trabajando con 2 vistas, una para un login de admin y otra para el administrador.

Una vez logeado el afiliado se le muestra la siguiente interfaz.



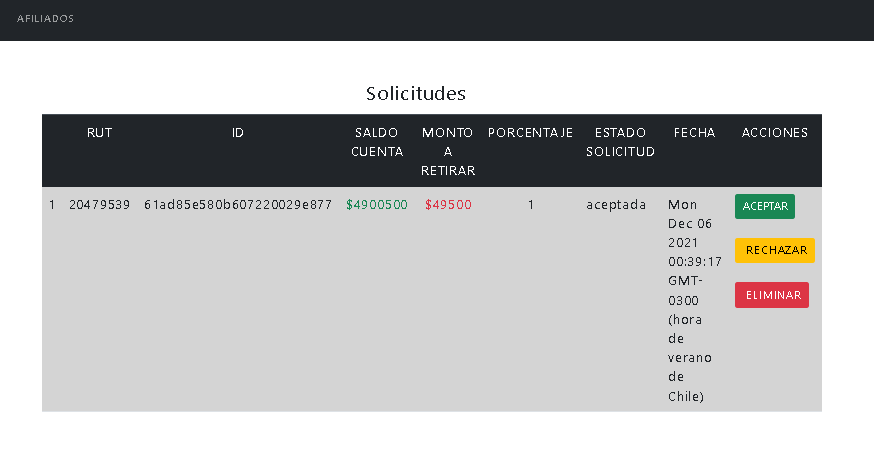
No existe gran diferencia con lo que es el mockup inicial, se presenta la información del afiliado, su cuenta y la opción para retirar, se decidió darle un estilo de wallet ya que la información que se muestra en esta pantalla no es mucha y por parte del desarrollo se tiene tendencia al minimalismo y lo simple.



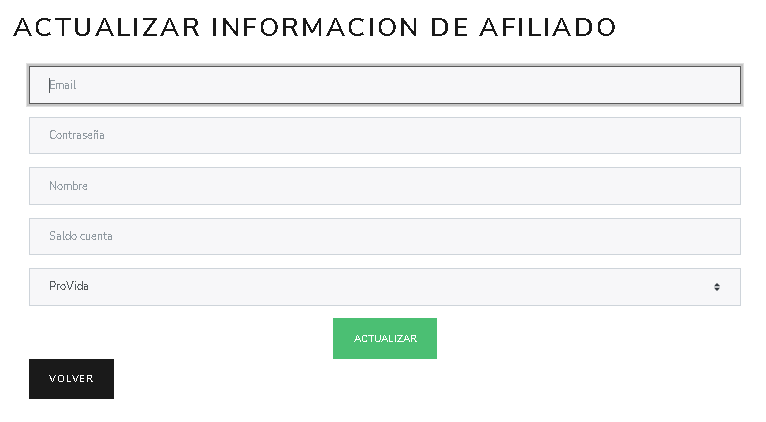
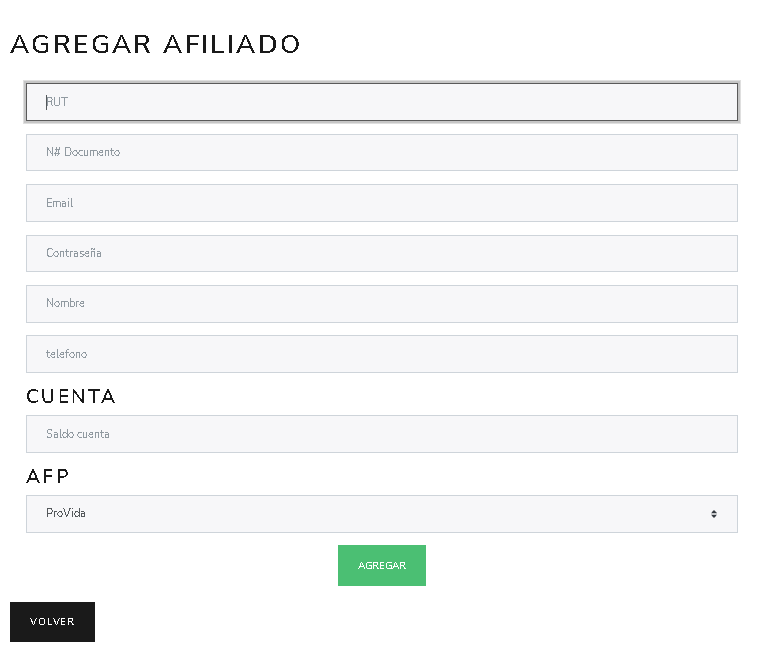
Se agrego un apartado donde el afiliado pueda ver las solicitudes que ha realizado.

**Back**

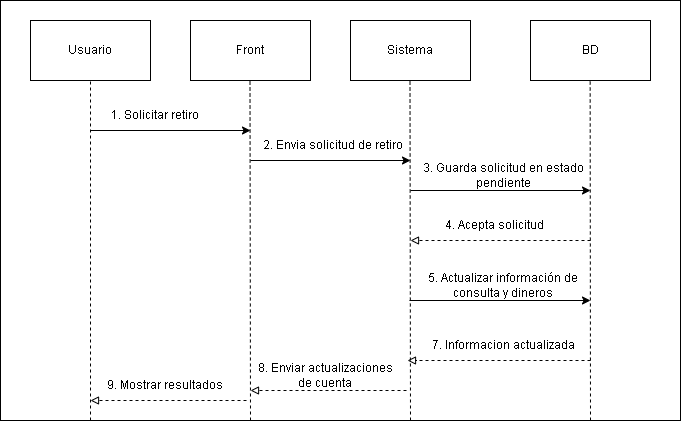
Por parte del back se hicieron cambios grandes, finalmente se tendrán 4 vistas, una donde ver las solicitudes realizadas por los usuarios, agregar afiliado, listar los afiliados y una vista para actualizar la información de un afiliado.





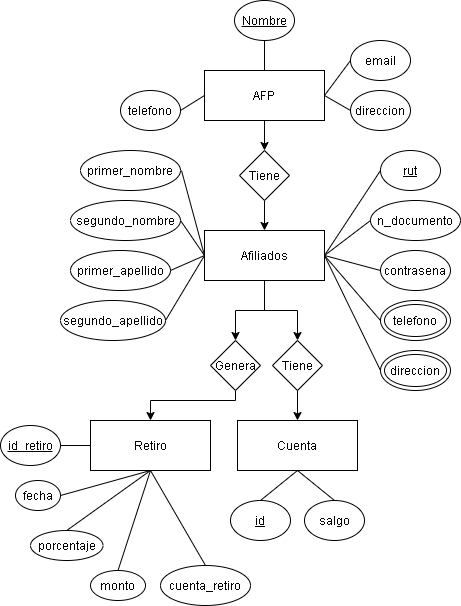


**Diagrama secuencial**

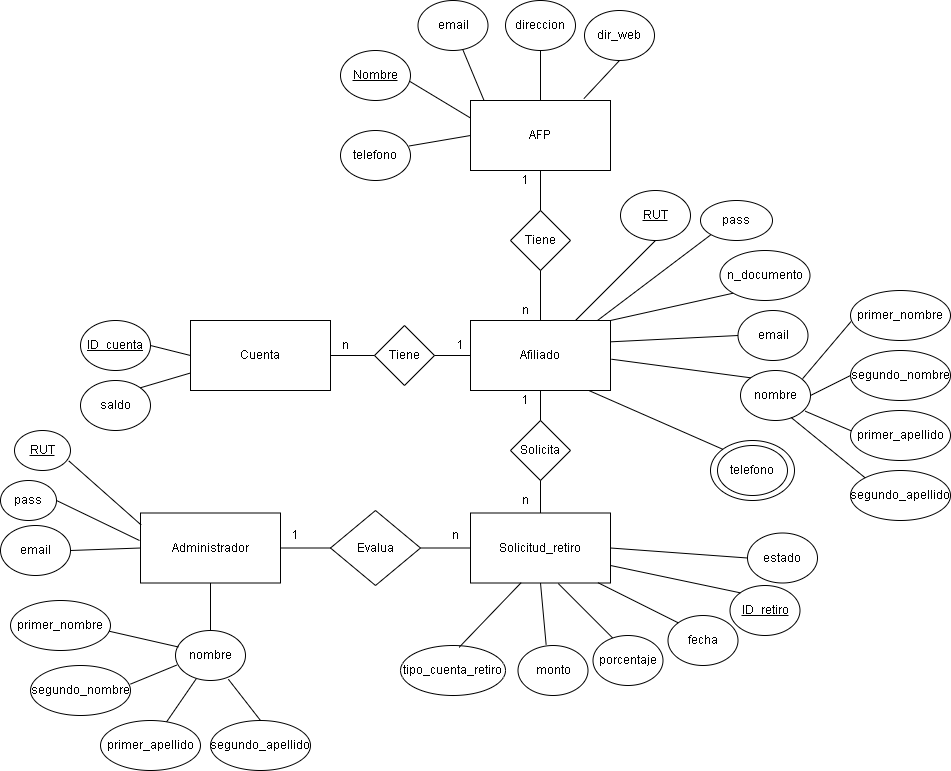
****

Para la figura anterior se considero el caso ideal, donde el afiliado solicita un retiro, no tiene solicitudes pendientes o retiros aceptados con anterioridad, por lo tanto, su solicitud se envía correctamente, un administrador acepta su solicitud cambiando su estado a “aceptada” y se pasa a actualizar los datos de su cuenta descontando el monto que solicito al saldo de su cuenta.

**Diagrama E/R inicial**



**Diagrama E/R final**



**Explicación de cambios realizados**

Se agrego la entidad **Administrador** con sus respectivos atributos, estos serán quienes administren las solicitudes realizadas por los afiliados, tienen acceso a la página web con características para administrar su información. **Retiro** paso a ser **Solicitud\_retiro** para mayor coherencia y se le agrego un atributo **estado** el cual representara el estado de la solicitud, pendiente, rechazada o aceptada.

Se agregaron las respectivas cardinalidades, un **afiliado** puede generar n **solicitud\_retiro** y n **solicitud\_retiro** puede ser analizada por 1 **administrador**, n **afiliado** pertenece a una **AFP**.

Un **afiliado** cuenta con 1 **cuenta**, por motivos de complejidad se modelo así.

Se quito la entidad **direccion**, para motivos del proyecto no tenía gran impacto.

Actualización de atributos:

**Silicitud\_retiro**:

**cuenta\_retiro** -> **tipo\_cuenta\_retiro**

Se agregado **estado**

**Afiliado**:

**contrasena** -> **pass**

Se agrega **email**

**AFP**:

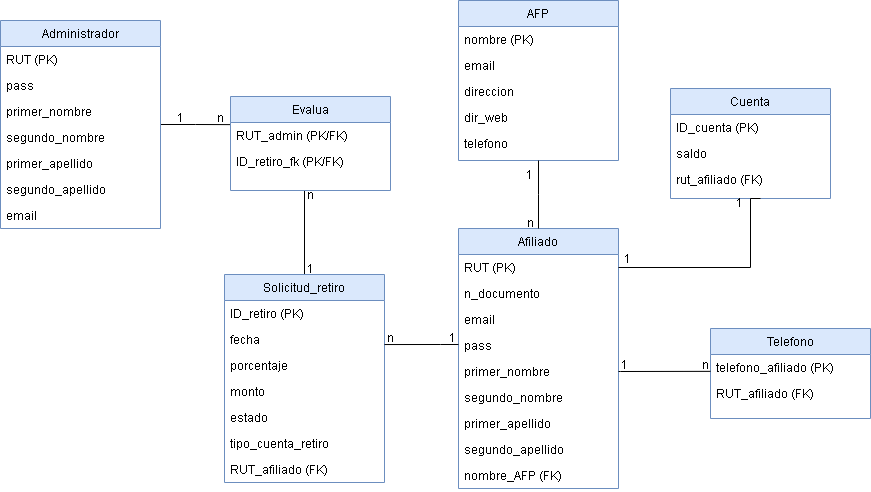
Se agrega **dir\_web**

**Cuenta**:

**salgo** -> **saldo**

Las actualizaciones de los atributos son con el fin de dar coherencia a sus nombres, reducir el tamaño de su nombre para un menor manejo o corregir de lleno su nombre.

**Diagrama relacional**



Se traspasa el modelo E/R a modelo relacional de la forma convencional. Se agrega una nueva entidad llamada **Evalua**, esto con el fin de representar que un **Administrador** pueda analizar n **Solicitud\_retiro**, esto en el modelo E/R se puede observar como **Administrador** evalua **Solicitud\_retiro** donde evalua representa la relación entre las 2 entidades.

El atributo multivaluado **telefono** pasa como una nueva entidad llamada **Telefono**, asignando **telefono\_afiliado** como su llave primaria/identificador y **RUT­­­\_afiliado** pasa como FK.

**Diccionario de datos**

**AFP**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nombre**  **atributo** | **Pk/Fk** | **Tipo de dato** | **Null/Not null** | **Valores por defecto** | **Comentario** |
| nombre | PK | Varchar (25) | Not null |  | Nombre de la AFP |
| email |  | Varchar (50) | Not null |  | Dirección única para la ubicación física de la AFP |
| direccion |  | Varchar (50) | Not null |  | Teléfono único de la AFP |
| dir\_web |  | Varchar (50) | Not null |  |  |
| teléfono |  | Varchar (25) | Not null |  |  |

**Afiliado**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nombre**  **atributo** | **Pk/Fk** | **Tipo de dato** | **Null/Not null** | **Valores por defecto** | **Comentario** |
| RUT | PK | Entero | Not null |  |  |
| n\_documento |  | Varchar (25) | Not null |  | Numero de documento de la cédula de identidad |
| email |  | Varchar (50) | Not null |  |  |
| pass |  | Varchar (50) | Not null |  | Contraseña del afiliado |
| primer\_nombre |  | Varchar (25) | Not null |  |  |
| segundo\_nombre |  | Varchar (25) | Not null |  |  |
| primer\_apellido |  | Varchar (25) | Not null |  |  |
| segundo\_apellido |  | Varchar (25) | Not null |  |  |
| nombre\_AFP | FK | Varchar (25) | Not null |  |  |

**Telefono**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nombre**  **atributo** | **Pk/Fk** | **Tipo de dato** | **Null/Not null** | **Valores por defecto** | **Comentario** |
| telefono\_afiliado | PK | Varchar (25) | Not null |  |  |
| rut\_afiliado | FK | Entero | Not null |  |  |

**Cuenta**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nombre**  **atributo** | **Pk/Fk** | **Tipo de dato** | **Null/Not null** | **Valores por defecto** | **Comentario** |
| ID\_cuenta | PK | Entero AUTO\_INCREMENT | Not null |  |  |
| saldo |  | Entero | Not null | 0 |  |
| rut\_afiliado | FK | Entero | Not null |  |  |

**Solicitud\_retiro**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nombre**  **atributo** | **Pk/Fk** | **Tipo de dato** | **Null/Not null** | **Valores por defecto** | **Comentario** |
| ID\_ retiro | PK | Entero AUTO\_INCREMENT | Not null |  |  |
| fecha |  | Date | Not null |  |  |
| porcentaje |  | Entero | Not null | 0 |  |
| monto |  | Entero | Not null | 0 |  |
| estado |  | Varchar (20) | Not null | 0 | Estado de la solicitud |
| tipo\_cuenta\_retiro |  | Varchar (20) | Not null |  | Tipo de cuenta en la que el afiliado retira el dinero |
| rut\_afiliado | FK | Entero | Not null |  |  |

**Administrador**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nombre**  **atributo** | **Pk/Fk** | **Tipo de dato** | **Null/Not null** | **Valores por defecto** | **Comentario** |
| RUT | PK | Entero | Not null |  |  |
| Pass |  | Varchar(50) | Not null |  |  |
| primer\_nombre |  | Varchar (25) | Not null |  |  |
| segundo\_nombre |  | Varchar (25) | Not null |  |  |
| primer\_apellido |  | Varchar (25) | Not null |  |  |
| segundo\_apellido |  | Varchar (25) | Not null |  |  |
| email |  | Varchar (50) | Not null |  |  |

**Evalua**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nombre**  **atributo** | **Pk/Fk** | **Tipo de dato** | **Null/Not null** | **Valores por defecto** | **Comentario** |
| Rut\_admin | PK/FK | Entero | Not null |  |  |
| ID\_retiro\_fk | PK/FK | Entero | Not null |  |  |

**Consultas etapa 1**

ALTER

|  |  |
| --- | --- |
| **Descripción** | **Consulta** |
| Borrar el atributo monto de la entidad tabla retiro. | alter table solicitud\_retiro drop column monto; |
| Añadir rango de restricciones para el atributo porcentaje. Debe ser menor o igual que 10. | alter table solicitud\_retiro add check (porcentaje<=10); |

SELECT

|  |  |
| --- | --- |
| **SQL** | |
| **Descripción** | **Consulta** |
| Mostrar el nombre y apellido de los afiliados. | select primer\_nombre, primer\_apellido from afiliado; |
| Mostrar la información y saldo de la cuenta de los afiliados. | select rut, primer\_nombre, primer\_apellido, saldo from afiliado join cuenta on afiliado.rut = cuenta.rut\_afiliado; |
| Mostrar la información de los afiliados que hicieron un retiro mayor a 500000. | select rut, primer\_nombre, primer\_apellido from (afiliado join solicitud\_solicitud\_retiro on afiliado.rut = solicitud\_retiro.rut\_afiliado) where monto >= 500000; |
| **Algebra relacional** | |
|  | pi afiliado.rut, afiliado.primer\_nombre, afiliado.primer\_apellido σ afiliado.rut = cuenta.rut\_afiliado (afiliado x cuenta) |
|  | pi afiliado.rut, afiliado.primer\_nombre, afiliado.primer\_apellido σ afiliado.rut = solicitud\_retiro.rut\_afiliado (afiliado x solicitud\_retiro) |
|  | pi afiliado.rut, afiliado.primer\_nombre, afiliado.primer\_apellido σ monto >= 500000 (σ afiliado.rut= solicitud\_retiro.rut\_afiliado (afiliado x solicitud\_retiro)) |

UPDATE

|  |  |
| --- | --- |
| **Descripción** | **Consulta** |
| Cambiar el nombre a “Daniel” del usuario con de RUT 20479539. | update afiliado set primer\_nombre = 'Daniel' where rut = 20479539 |
| Cambiar la pass(contraseña) del afiliado que tenga el número de documento 111111. | update afiliado set pass = 'uno2tres@' where n\_documento = 111111; |

INSERT

|  |  |
| --- | --- |
| **Descripción** | **Consulta** |
| Agregar la AFP Plan Vital y toda su información a la tabla AFP. | insert into afp values ('Plan Vital', 'afpplanvital@gmail.com','asdasd 123',’ www.planvital.cl’,'600 6000 301'); |
| Agregar la AFP Modelo y toda su información a la tabla AFP. | insert into afp values ('Modelo’, 'afpplanvital@gmail.com','asdasd 123', ‘www.afpmodelo.cl’,’600 828 7200’); |
| Insertar teléfono a la tabla telefono. | insert into telefono values('97745254',20479539); |

DELETE

|  |  |
| --- | --- |
| **Descripción** | **Consulta** |
| Borrar la información dentro de la tabla telefono del afiliado con rut 20479539 | delete from telefono where rut\_afiliado = 20479539; |
| Borrar la información dentro de la tabla afiliado de afiliado con rut 20479539 | delete from afiliado where rut = 20479539 |

DROP

|  |  |
| --- | --- |
| **Descripción** | **Consulta** |
| Borrar la tabla solicitud retiro si existe. | drop table if exists solicitud\_retiro; |

**Consultas etapa 2**

**OPERADORES + FUNCIONES AGREGAS**

**Descripción**

Obtener nombre y monto total de los afiliados que hicieron retiro de fondos (x) veces, (tiene que estar aceptada la solicitud de retiros).

**SQL**

select concat(primer\_apellido," ",primer\_nombre) as Nombre, sum(monto) as total\_retirado from (afiliado join solicitud\_retiro on rut = rut\_afiliado and estado = "aceptada") group by rut\_afiliado having count(\*) = 1 order by primer\_apellido desc;

**Algebra relacional**

τ primer\_apellido ↓π SUM (monto) → total\_retirado σ COUNT (\*) = 1 γ rut\_afiliado, COUNT (\*), SUM (monto) (afiliado ⋈ rut = rut\_afiliado AND estado = "aceptada" solicitud\_retiro)

**Descripción**

Mostrar el id y valor del retiro de la cuenta que hizo el retiro máximo x fecha solo si el porcentaje indicado es = 10% para todas las AFP menos PlanVital y Cuprum.

**SQL**

select id\_cuenta, monto as cantidad from (afp join afiliado on nombre = nombre\_afp join solicitud\_retiro on rut = solicitud\_retiro.rut\_afiliado join cuenta on rut = cuenta.rut\_afiliado ) where estado = "aceptada" and fecha = "2021-05-12" and porcentaje = 10 and nombre not in ("PlanVital","Cuprum");

**Algebra relacional**

π id\_cuenta, monto → cantidad σ estado = 1 AND fecha = "2021-05-12" AND porcentaje = 10 AND NOT (nombre = "PlanVital" OR nombre = "Cuprum") (afp ⋈ nombre = nombre\_afp afiliado ⋈ rut = solicitud\_retiro . rut\_afiliado solicitud\_retiro ⋈ rut = cuenta . rut\_afiliado cuenta)

**Descripción**

obtener la cantidad de retiros y total retirado por AFP de todos los afiliados que hicieron retiros antes del año (2022) ordenados por cantidad de retiros de manera descendente.

**SQL**

select nombre\_afp, count(rut\_afiliado) as cant\_retiros, sum(monto) total from afp join afiliado on nombre = nombre\_afp join solicitud\_retiro on rut = rut\_afiliado where estado = "aceptada" group by nombre having max(fecha) <= "2021-12-31" order by count(rut\_afiliado) desc;

**Algebra relacional**

τ COUNT (rut\_afiliado) ↓ π nombre\_afp, COUNT (rut\_afiliado) → cant\_retiros, SUM (monto) → total σ MAX (fecha) <= "2021-12-31"γ nombre, MAX (fecha), COUNT (rut\_afiliado), SUM (monto) (afp ⋈ nombre = nombre\_afp afiliado ⋈ rut = rut\_afiliado solicitud\_retiro)

**Descripción**

Mostrar la información de los administradores ordenados por la cantidad de solicitudes que analizaron**.**

**SQL**

select rut, concat(primer\_apellido," ",primer\_nombre) as Nombre, count(\*) as sol\_analizadas from administrador join evalua on administrador.RUT = evalua.rut\_admin order by sol\_analizadas;

**Algebra relacional**

τ sol\_analizadas π rut, primer\_apellido+' '+primer\_nombre → Nombre, COUNT (\*) → sol\_analizadas γ COUNT (\*) (administrador ⋈ administrador . rut = evalua . rut\_admin evalua)

**Descripción**

Mostrar los montos totales de las solicitudes según su estado.

**SQL**

select estado, sum(monto) as Total from solicitud\_retiro GROUP by estado;

**Algebra relacional**

τ estado π estado, SUM (monto) → total γ SUM (monto) solicitud\_retiro

**SUBCONSULTAS**

**Descripción**

Mostrar la información de los afiliados que hicieron retiros por sobre la media ordenados de manera descendente por el primer apellido.

**SQL**

select concat(primer\_apellido," " ,primer\_nombre) as Nombre, rut, monto from solicitud\_retiro join afiliado on solicitud\_retiro.rut\_afiliado = afiliado.rut where monto > (select avg(monto) from solicitud\_retiro) and estado = "aceptada " order by primer\_apellido;

**Descripción**

Mostrar la información de los afiliados que hicieron retiros por bajo la media.

**SQL**

select concat(primer\_apellido, " ",primer\_nombre) as Nombre, rut, monto from solicitud\_retiro join afiliado on solicitud\_retiro.rut\_afiliado = afiliado.rut where monto < (select avg(monto) from solicitud\_retiro) and estado = "aceptada" order by primer\_apellido;

**Descripción**

Mostrar la cantidad de usuarios que solicitaron retiros y que fueron rechazados o siguen estando pendiente.

**SQL**

select count(\*) as Total from afiliado where rut not in (select rut\_afiliado from solicitud\_retiro where estado = "aceptada");

**Descripción**

Mostrar la informacion, cantidad de solicitudes de los usuarios que solitaron retiros y que fueron rechazas o siguen estando pendiente.

**SQL**

select concat(primer\_apellido," ",primer\_nombre) as Nombre, rut\_afiliado, count(\*) as total from solicitud\_retiro join afiliado on afiliado.rut = solicitud\_retiro.rut\_afiliado where rut\_afiliado not in (select rut\_afiliado from solicitud\_retiro where estado = "aceptada") group by RUT order by count(\*) desc;;

**GRUOP BY**

**Descripción**

Mostrar la cantidad de retiros e información de cada afiliado, ordenados de manera desc por cantidad de retiros.

**SQL**

select concat(primer\_apellido, " ",primer\_nombre) as Nombre, rut, count(\*) as total from afiliado join solicitud\_retiro on solicitud\_retiro.rut\_afiliado=afiliado.RUT where estado = "aceptada" group by RUT;

**Algebra relacional**

π primer\_apellido, primero\_nombre, rut, COUNT (\*) → total γ rut, COUNT (\*) σ estado = "aceptada" (afiliado ⋈ solicitud\_retiro . rut\_afiliado = afiliado . rut solicitud\_retiro)

**Descripción**

Mostrar el conteo de solicitudes aceptadas organizadas por AFP.

**SQL**

select nombre\_afp, count(\*) as aceptadas from solicitud\_retiro join afiliado on solicitud\_retiro.rut\_afiliado = afiliado.rut join AFP on afp.nombre = afiliado.nombre\_afp where estado = "aceptada" group by nombre;

**Algebra relacional**

π nombre\_afp, COUNT (\*) → aceptadas γ nombre, COUNT (\*) σ estado = "aceptada" (solicitud\_retiro ⋈ solicitud\_retiro . rut\_afiliado = afiliado . rut afiliado ⋈ afp . nombre = afiliado . nombre\_afp afp)

**Descripción**

Mostrar el conteo de solicitudes rechazadas organizadas por AFP.

**SQL**

select nombre\_afp, count(\*) as rechazadas from solicitud\_retiro join afiliado on solicitud\_retiro.rut\_afiliado = afiliado.rut join AFP on afp.nombre = afiliado.nombre\_afp where estado = "rechazada" group by nombre;

**Algebra relacional**

π nombre\_afp, COUNT (\*) → rechazadas γ nombre, COUNT (\*) σ estado = "rechazada" (solicitud\_retiro ⋈ solicitud\_retiro . rut\_afiliado = afiliado . rut afiliado ⋈ afp . nombre = afiliado . nombre\_afp afp)

**Descripción**

Mostrar el conteo total de solicitud organizadas por su estado.

**SQL**

select estado, count(\*) as Cantidad from solicitud\_retiro group by estado;

**Algebra relacional**

π estado, COUNT (\*) → cantidad γ estado, COUNT (\*) solicitud\_retiro

**Descripción**

Mostrar el conteo de retiros dependiendo del porcentaje que se retiró ordenado por total de manera descendente.

**SQL**

select porcentaje, COUNT(\*) as total from solicitud\_retiro where estado = "aceptada" GROUP by porcentaje order by total desc;

**Algebra relacional**

τ total ↓π porcentaje, COUNT (\*) → total γ porcentaje, COUNT (\*) σ estado = "Aceptada" solicitud\_retiro

**Descripción**

Mostrar la media de porcentaje que se solicita para retiro y nombre de la AFP para los retiros que fueron aceptados.

**SQL**

select ROUND(AVG(porcentaje),1) as Media\_porcentaje, nombre\_afp from solicitud\_retiro join afiliado on solicitud\_retiro.rut\_afiliado = afiliado.rut join afp on afp.nombre = afiliado.nombre\_afp where estado = "Aceptada" group by nombre\_afp;

**Algebra relacional**

π AVG (porcentaje) → media\_porcentaje, nombre\_afp γ nombre\_afp, AVG (porcentaje) σ estado = 1 (solicitud\_retiro ⋈ solicitud\_retiro . rut\_afiliado = afiliado . rut afiliado ⋈ afp . nombre = afiliado . nombre\_afp afp)

**Conclusión**

El objetivo del presente proyecto era crear un sistema que brindara la posibilidad a los afiliados a los retiros de fondos AFP de hacer su retiro. El objetivo a grandes rasgos se pudo cumplir, la gran ayuda de las tecnologías actuales influyo mucho a que esto se pueda llevar a su término. Pasar de solo ideas y modelos mentales a algo concreto es muy satisfactorio para quien desarrolla, durante el transcurso del desarrollo de la aplicación pareciera que se llega a un cierto límite de complejidad, pero de la misma forma aparece esa sensación de necesitar mejorar, implementar algo más; esto aporta mucho en el desarrollo académico, sin conocer a fondo o incluso sin conocer algunas tecnologías como Node, Mongodb, Html, Css y Javascript, la necesidad de usar aplicaciones para el manejo de las anteriores hicieron sentir que ya me puedo considerar “Programador”.

Lo que más rescato de este proyecto es la parte en la cual debo modelarlo, soy de los que prefiere programar antes que modelar. Me di cuenta de la importancia que tiene tener todo bien planeado antes de “meter mano” a un código, en el transcurso me surgieron muchas dificultades por esto, cosas como: cambiar nombre de un atributo, diferentes nombres de un atributo en los modelos de BD, archivos mal colocados, programar algo que no modele y tener que modelar todo de nuevo o agregarlo y que esto me complique más. Entonces por fin entendí. Este sin duda es el proyecto que más me ha ayudado a avanzar en todo lo que llevo de estudiante universitario.

**Repositorio Github**

<https://github.com/DiegoEspinozaA/retiro-10-porciento>-3