## Relevamiento Situación actual de Convenios

## Histórico de Convenios:

Se invoca el método HistConvenio desde Oficina que se mapea con GetHisConvenio en AutoConsultaT1 dando como parámetro el número de cliente.

*ModeloCliente().HistConvenio(Nrocliente) -> internamente se traduce a AutoConsulta.****GetHisConvenio***

El método AutoConsultaT1. GetHisConvenio (NroCliente) se encuentra dentro del archivo SCliente.cs dentro del proyecto BusinessEntities.

Se ejecuta la siguiente query por medio de la conexión basemac con parámetro

**select first 10 \* from conve where (numero\_cliente = {0} and estado = 'S') or (numero\_cliente = {0} and estado = 'V') order by fecha\_creacion desc**

Por cada fila devuelta se ejecuta el método CrearConvenio que recibe como parámetro el row y se devuelve el siguiente conjunto de datos desde dicho método

|  |  |
| --- | --- |
| **Columna** | **Valor mapeado de la base de datos o fijo** |
| **Econvenio.Motivo** | **“OK”** |
| **Econvenio.DeudaConvenida** | **deuda\_convenida** |
| **EConvenio.Anticipo** | **valor\_cuota\_ini** |
| **EConvenio.ValorCuota** | **valor\_cuota** |
| **EConvenio.Cuotas** | **numero\_tot\_cuotas** |
| **EConvenio.EstadoConvenio** | **Toma su valor del campo estado pero luego pregunta:**  switch (EConvenio.EstadoConvenio)  {  case "V":  EConvenio.EstadoConvenio = "Vigente";  break;  case "S":  EConvenio.EstadoConvenio = "Solicitado";  break;  } |
| **EConvenio.Fecha** | **fecha\_creacion. Si el campo está vacío pone la fecha del sistema.** |
| **EConvenio.TasaInteres** | **Intereses** |
| **EConvenio.NroCliente** | **numero\_cliente** |
| **EConvenio.Msg** | **intereses** |
| **EConvenio.FechaVencimiento1** | **fecha\_vto\_1. Si el campo está vacío asigna la fecha del sistema** |
| **EConvenio.FechaVencimiento2** | **fecha\_vto\_2. Si el campo está vacío asigna la fecha del sistema** |
| **EConvenio.Motivo** | **Si se produce una excepción general dicho campo se asigna el literal “No”** |

**Si el anterior dataset fuera nulo entonces se devuelve a oficina un objeto EConvenio con el siguiente valor:**

|  |  |
| --- | --- |
| **Columna** | **Valor** |
| **EConvenio.existe** | **False** |
| **EConvenio.Motivo** | **“Sin información”** |
| **EConvenio.NroCliente** | **Es el número de cliente que se le pasa como parámetro al método GetHisConvenio** |

**Orden de Llamadas para el método HistConvenio**

|  |  |
| --- | --- |
| **Método** | **Ubicación** |
| 1. **HistConvenio con número de cliente** | **ModeloCliente.cs en el proyecto Edesur.OV.Modelo en oficina** |
| 1. **GetHisConvenio con número de cliente** | **SCliente.cs en el proyecto BusinessService en AutoConsultaT1** |

**Lugares de Invocación de HistConvenio**

|  |  |
| --- | --- |
| **Método** | **Ubicación** |
| **CargarClienteT1** | **solicitarconvenio.aspx.cs en el proyecto Edesur.OV.Web en oficina** |
| **CargarClienteT1** | **historicodeconvenio.aspx.cs en el proyecto Edesur.OV.Web en oficina.** |
| **NuevoIniWsPagoTarjeta (\*)** | **IniWsPagoTerjetaAnticipoConvenio.aspx.cs en el proyecto Edesur.OV.Web en oficina. Se invoca una vez en el código pero después no se usa** |
| **NuevoIniWsPagoTarjeta (\*)** | **IniWsPagoTerjetaAnticipoConvenioApp.aspx.cs en el proyecto Edesur.OV.Web en oficina. Se invoca una vez en el código pero después no se usa** |

**Lógica empleada en los lugares de invocación**

**solicitarconvenio.aspx.cs**

Se asigna el datasource obtenido a la grilla.

**historicodeconvenio.aspx.cs**

Se asigna el datasource obtenido a la grilla.

## Solicitar Convenios:

### Visualización de pantallas

Tiene lugar dentro del método CargarClienteT1 en solitarconvenio.aspx.cs dentro del proyecto Edesur.OV.Web. Se invoca a **GetEstadoConvenioT1** dándolo como parámetro el número de cliente. Dentro de dicho método tiene lugar la invocación de otros dos métodos que desarrollaremos después:

1. AutoConsultaT1.ExisteConvenio(nroCliente) -> Objeto devuelto: EConvenio que se mapeara con una variable EConvenio.
2. AutoConsultaT1.GetConvenio(nroCliente) -> Objeto devuelto: EConvenio que se mapeará con una variable GetEConvenio

Si EConvenio.Existe==False entonces

Si GetEConvenio.DeudaConvenida!=0 entonces

Invocar al método LlenarComboConvenios(GetEConvenio.Cuotas)

Determinar monto convenido que se calcula a partir de la siguiente

operación:

convenido = (convenido - (convenido \*

Convert.ToInt16(GetEConvenio.Anticipo) / 100));

Invocar al método EstablecerConvenios(GetEConvenio.Cuotas, convenido)

Sino

Informar “No tiene saldo apto para convenio”

Fin-Si

Si EConvenio.Existe==True entonces:

Si EConvenio.Motivo==”**Convenios Caducados**” entonces:

Informar “Llegaste al límite de 3 planes de pago caducados en 12 meses, por lo cual no podés generar uno nuevo en este momento.”

Sino EConvenio.Motivo==”**Antigüedad de Saldo”** entonces:

Informar “No es posible generar el convenio ya que la deuda supera los 180 días. Por favor, llamanos al 0810-222-0200”

Sino EConvenio.Motivo==”**Convenio Vigente**” entonces:

Si Session[“conveniogenerado”] distinto de “ahora” entonces

Se guarda en sesión “**numerocomprobante”** el valor dado por

EConvenio.NroComprob

Informar en el texto “No es posible generar el convenio ya que existe un plan

activo”

Fin Si

Sino EConvenio.Motivo==”**Convenio Pendiente**” entonces:

Si Session[“conveniogenerado”] distinto de “ahora” entonces

Se guarda en sesión “numerocomprobante” el valor dado por

EConvenio.NroComprob

Setea en un label “No es posible generar el convenio ya que existe un plan

activo”

Invocar al método **LeerConvenioVigente** con número de cliente

que llama al método **AutoConsultaT1.GetConvenioVigente** que

devuelve un objeto de tipo **EConvenio** y carga los datos en

sesion. Desde LeerConvenio se despliega pantalla con cuotas

para solicitar convenio. Además el label se hace invisible

Fin Si

Fin Si

Si el valor de sesion de “convenio generado”!=null entonces

Si el valor de sesion de “convenio generado”==”ahora” entonces

Invocar al método LeerConvenioVigente con número de cliente

Configurar el valor de fracción con 4

Guardar en sesion los siguientes datos: nrocliente, fracción, permitidopago con valor true y EPagosOrigen con valor “ACONV”

MontoPVD con fAnticipo de la sesion o sino con 0

Recuperar el valor re de sesion.

Si re ==”true” entonces

Llamar al método GenerarValores sin parámetros

Fin-Si

Llamar al método EConv sin parámetros el cual invoca a ImprimirPagarConvenio que informa lo siguiente al usuario:

“**Se generó la solicitud de convenio. Para que quede activo tenés que pagar el anticipo dentro de las 48 hs. con tarjeta, desde aquí, o imprimir el comprobante y abonarlo por nuestros canales de pago habituales**”

Fin Si

Fin Si

Si Existe convenio (EConvenio.Existe==true) y la deuda convenida es distinto de cero (GetEConvenio.DeudaConvenida!=0) entonces

Invocar al método **LeerConvenioVigente** con número de cliente

que llama al método **AutoConsultaT1.GetConvenioVigente** que

devuelve un objeto de tipo **EConvenio** y carga los datos en

sesion. Desde LeerConvenio se despliega pantalla con cuotas

para solicitar convenio.

Si (EConvenio.Motivo==”Convenios Caducados” o EConvenio.Motivo==”Antigüedad de Saldos”) entonces

Deshabilitar botones pago anticipo Convenio

Si EConvenvio.Motivo==”Convenios Caducados” entonces

Informar “Llegaste al límite de 3 planes de pago caducados en 12 meses, por lo cual no podés generar uno nuevo en este momento.”

Fin Si

Si EConvenio.Motivo==”Antigüedad de saldos” entonces

Informar “No es posible generar el convenio ya que la deuda supera los 180 días. Por favor, llamanos al 0810-222-0200”

Fin Si

Fin Si

Fin Si

|  |  |
| --- | --- |
| Método | Descripción asociada |
| AutoConsultaT1.ExisteConvenio | Recibe el número de cliente y ejecuta la siguiente el siguiente stored procedure por medio de la instrucción  EXECUTE PROCEDURE web\_valida\_conve  Si se devuelve 1 ó 2 se setea EConvenio.existe en True. Se asignan los siguientes atributos:  EConvenio.Motivo  EConvenio.NroCliente  en caso contrario quedará en false, asignándose las propiedades de la siguiente manera:  EConvenio.existe=false  EConvenio.Motivo=”Sin Información”  EConvenio.NroCliente=numerocliente.  Para todos los casos se devuelve un objeto del tipo EConvenio. |
| AutoConsultaT1.GetConvenio | Al igual que en el caso anterior recibe el número de cliente y ejecuta el siguiente stored procedure por medio de la instrucción.  EXECUTE PROCEDURE web\_saldo\_conve  Se devuelve un objeto del tipo EConvenio y se cargan los siguientes atributos si hay datos.  **Motivo=OK**  **Existe=True**  **DeudaConvenida**  **Anticipo**  **Cuotas**  **NroCliente**  Sino hay datos en el resultset entonces se hace la siguiente asignación:  **Existe=False**  **Motivo=”Sin Información”**  **NroCliente** |
| LeerConvenioVigente | Método que recibe el número de cliente y llama a AutoConsultaT1.GetConvenioVigente, informa la deuda a convenir del convenio vigente, el anticipo, las cuotas y el valor de cada cuota, llama a AutoConsultaT1.GetConvenioVigenteBarra, devolviendo un objeto del tipo EConvenio.  Se encuentra en oficina en solicitarconvenios.aspx.cs |
| AutoConsultaT1.GetConvenioVigenteBarra | Se ejecutan dos stored procedures. El primero de ellos carga los siguientes datos para un objeto EConvenio  EXECUTE PROCEDURE web\_conve\_solic  Si hay datos se asignan los siguientes atributos  EConvenio.Existe=True  EConvenio.Motivo=”OK”  EConvenio.DeudaConvenida  EConvenio.Anticipo  EConvenio.Cuotas  EConvenio.ValorCuotas  EConvenio.Fecha  EConvenio.NroCliente  EConvenio.TasaInteres  EConvenio.FechaVencimiento1  EConvenio.FechaVencimiento2  EConvenio.NroComprob  Sino hay información del convenio:  EConvenio.Motivo=”Sin Información”  EConvenio.NroCliente  Luego se ejecuta el otro storedprocedure  EXECUTE PROCEDURE web\_conve\_anticipo  Que recibe como parámetros el número de cliente, EConvenio.anticipo, EConvenio.FechaVencimiento1 y EConvenio.FechaVencimiento2  Si hay datos asigna los siguientes valores:  EConvenio.CodBarraAnticipoConvenio  EConvenio.NroComprob = EConvenio.CodBarraAnticipoConvenio.substring(40,3)  Sino hay datos asociados al código de barra:  EConvenio.Motivo=”Sin Codigo de Barras”  EConvenio.NroCliente  En todos los casos se devuelve un objeto del tipo EConvenio. |
| AutoConsultaT1.GetConvenioVigente | Recibe el número de cliente y ejecuta el siguiente stored procedure por medio de la instrucción.  EXECUTE PROCEDURE web\_conve\_solic  Se devuelve un objeto del tipo EConvenio.  Si hay registros entonces se realiza la carga de datos para los siguientes atributos  EConvenio.Existe=True  EConvenio.Motivo=”OK”  EConvenio.DeudaConvenida  EConvenio.Anticipo  EConvenio.ValorCuotas  EConvenio.Cuotas  EConvenio.TasaInteres  EConvenio.FechaVencimiento1  EConvenio.FecgaVencimiento2  EConvenio.NroComprob  Si no hay registros entonces se cargan los siguientes atributos:  EConvenio.Existe=False  EConvenio.Motivo=”Sin Información”  EConvenio.NroCliente=numerocliente |
| GenerarValores | No recibe parámetros. Método definido en solicitarconvenios.aspx.cs. Llama a obtenerPorCliente con los siguientes valores:  NroCliente  esPago=” PagoAnticipoConvenio”  CodigoBarraPVD: De sesión toma CodBarraAnticipoConvenio.  obtenerPorCliente se encuentra definido en Oficina, Edesur.OV.Modelo.ModeloComprobantePago.  Se devuelve un tipo file que tiene el comprobante generado por tipo de cliente, pero no devuelve el resultado al método que lo invocó. |
| LlenarComboConvenios | Recibe como parámetros las cuotas y efectua el llenado de los combos que visualizará el usuario. Se encuentra definida en solicitarconvenios.aspx.cs |
| EstablecerConvenios | Recibe dos parámetros: Cuotas y el importe convenido: Divide el convenido por las cuotas y determina el monto de la cuota. Se encuentra definida en solicitarconvenios.aspx.cs |

### Pagos de Convenio

Los componentes intervinientes del lado de oficina son los siguientes:

|  |  |
| --- | --- |
| Componente | Descripción asociada |
| MisFacturas.aspx.cs | Invocación del método btnIniPagoAntConvenio\_Click. Se invoca al método PagoHistAnticipoConvenio.  Configura en memoria los siguientes datos:  nroCliente  Fracción: 4  PermitidoPago: True  EPagosOrigen: ACONV  Recupera el valor re de sesion. Si el valor es true entonces invoca a la rutina GenerarValores y redirige ~/Paginas/IniWsPagoTerjetaAnticipoConvenio.aspx |
| PagoHistAnticipoConvenio | Se trata de una rutina que se encuentra en MisFacturas.aspx.cs. Tiene por misión descargar el comprobante de pago a través de la llamada ModeloComprobantePago.obtenerPorCliente en donde se le pasan como parámetros:  NroCliente  EsPago=”HistAnticipoConvenio”  CodigoBarraAnticipoConvenio = nulo.  La variable CodigoBarraAnticipoConvenio se carga dentro del método ModeloComprobantePago.obtenerPorCliente al invocar a AutoConsultaT1.GetConvenioVigenteBarra dándole como parámetro el número de cliente.  Se utiliza en solicitarConvenio y en historicodeconvenio. |
| IniWsPagoTerjetaAnticipoConvenio.aspx.cs | Se llama al método NuevoIniWsPagoTarjeta(). Si el flag de UsarPrisma esta seteado en 1 invoca al método WS\_Prisma con los siguientes valores recuperados de sesion  NroCliente  Importe -> MontoPVD en la sesion  CodBarraAnticipoConvenio  EmailPagoTarjeta (2 veces)  PaymentRedirectURL  MerchantRedirectURL  Sino se usa prisma entonces  Se invoca a AutoConsultaT1.GetConvenioVigenteBarra con el numerodeCliente.  Si EConvenio.existe==true entonces  Llamar a InitTransaccion con los siguientes parámetros NroCliente, Importe y EConvenio.CodBarraAnticipoConvenio. |
| InitTransaccion | Llama a ExecuteTransaction y redirecciona a WSPagoTarjetaAnticipoConvenio.aspx para cargar los datos de la tarjeta. Completado esta fase se invocará a RespuestaPagoTarjetaAnticipoConvenio. |

* WSPagoTarjetaAnticipoConvenioapp
* solicitarConvenioapp
* RespuestaPagoTarjetaAnticipoConvenioApp
* GenerarAdjuntoAnticipoConvenioApp
* DescargaPdfAnticipoConvenioApp
* PuenteDescargaConvenio
* PuenteDescargaConvenioApp
* PuenteEstadoCuentaAppConvenio
* PuenteHistoricoAppConvenio
* PuenteInicioDescargaConvenioApp
* PuenteInicioImpresionConvenio
* PuenteInicioImpresionConvenioApp

**Pagos:**

ASUMIENDO QUE LA PERSONA TIENE UN CONVENIO SOLICITADO

* PuenteDescargaConvenio – descarga de un convenio solicitado
* PuenteInicioDescargaConvenio
* PuenteInicioImpresionConvenio
* IniWsPagoTerjetaAnticipoConvenio.aspx.cs
* GenerarAdjuntoAnticipoConvenio
* WSPagoTarjetaAnticipoConvenio
* RespuestaPagoTarjetaAnticipoConvenio
* DescargaPdfAnticipoConvenio

**Convenio App**

* PuenteDescargaConvenioApp

PuenteInicioImpresionConvenioApp

* PuenteInicioDescargaConvenioApp
* PuenteEstadoCuentaAppConvenio -> solicitarconvenioapp -> al de abajo
* IniWsPagoTerjetaAnticipoConvenioApp.aspx.cs
* GenerarAdjuntoAnticipoConvenioApp
* WSPagoTarjetaAnticipoConvenioApp.aspx.cs
* RespuestaPagoTarjetaAnticipoConvenioApp
* DescargaPdfAnticipoConvenioApp

**Solicitud**

solicitarConvenioapp

solicitarConvenio

**Histórico:**

PuenteHistoricoAppConvenio

PuenteDescargaConvenioApp:

Dentro del método TraerConvenio(nroCliente) se invoca al método **AutoConsultaT1.GetConvenioVigenteBarra** mencionado anteriormente en este documento. Se guarda en sesión la información asociada al convenio, se considera un objeto de tipo EConvenio y los datos a considerar son los siguientes: DeudaConvenida, Cuotas, ValorCuotas, TasaInteres, Anticipo, CodBarraAnticipoConvenio, FechaVencimiento1, FechaVencimiento2 y Fecha. Cabe destacar que dentro del método ImpresionAnticipoConvenio2, el cual es invocado desde el método Page\_load tiene lugar la llamada a ModeloComprobantePago().obtenerPorCliente() que recibe el número de cliente, el string esPago y el CodigoBarraPVD. Dentro del método indicado se pregunta si el tipo de pago es “PagoAnticipoConvenio”, ”PagoAppConvenio” o “HistAnticipoConvenio”, en caso afirmativo se invoca a **AutoConsultaT1.GetConvenioVigenteBarra(numerocliente)**, el cual ya se indicó con anterioridad.

PuenteInicioDescargaConvenioApp

Se invoca desde PuenteDescargaConvenioApp. En el método TraerConvenio(nroCliente) se invoca al método **AutoConsultaT1.GetConvenioVigenteBarra**, mencionado anteriormente en este documento. Se guarda en sesión la información asociada al convenio, se considera un objeto de tipo EConvenio y los datos a considerar son los siguientes: DeudaConvenida, Cuotas, ValorCuotas, TasaInteres, Anticipo, CodBarraAnticipoConvenio, FechaVencimiento1, FechaVencimiento2 y Fecha. Cabe destacar que dentro del método ImpresionAnticipoConvenio2, el cual es invocado desde el método Page\_load tiene lugar la llamada a ModeloComprobantePago().obtenerPorCliente() que recibe el número de cliente, el string esPago y el CodigoBarraPVD. Dentro del método indicado se pregunta si el tipo de pago es “PagoAnticipoConvenio”, ”PagoAppConvenio” o “HistAnticipoConvenio”, en caso afirmativo se invoca a **AutoConsultaT1.GetConvenioVigenteBarra(numerocliente)**, el cual ya se indicó con anterioridad.

PuenteInicioImpresionConvenioApp

Se invoca desde IniWsPagoTerjetaAnticipoConvenioApp. En los métodos: ImpresionDeudor, ImpresionComprobante, ImpresionParcial, ImpresionEnre, ImpresionAnticipoConvenio e ImpresionTotal se utiliza el método ModeloComprobante().obtenerPorCliente, el cual recibe el NumeroCliente, esPago (con el tipo de pago) y CodigoBarraPVD. Dentro de dicho método (obtenerPorCliente) si el parámetro esPago es igual a PagoAnticipoConvenio, PagoAppConvenio o HistAnticipoConvenio se invoca a **AutoConsultaT1.GetConvenioVigenteBarra**(numeroCliente), el cual ya hemos mencionado en el presente documento.

PuenteHistoricoAppConvenio

Desde este punto se invoca a solicitarConvenioApp

solicitarConvenioapp

Similar a solicitarConvenio. Los métodos a considerar son los siguientes: CargarClienteT1, GetEstadoConvenioT1, LeerConvenioVigente, SolicitarConvenioT1\_V1, AutoConsultaT1.GenerarConvenioNew.

PuenteEstadoCuentaAppConvenio:

Desde este punto se invoca a solicitarConvenioApp

RespuestaPagoTarjetaAnticipoConvenioApp: Se menciona en el relevamiento. No requiere modificaciones si tomamos como referencia la oficina productiva.

WSPagoTarjetaAnticipoConvenioapp: Se menciona en el relevamiento. No requiere modificaciones si tomamos como referencia la oficina productiva.

GenerarAdjuntoAnticipoConvenioApp: Se menciona en el relevamiento. No requiere modificaciones si tomamos como referencia la oficina productiva.

DescargaPdfAnticipoConvenioApp: Se menciona en el relevamiento. No requiere modificaciones adicionales si tomamos como referencia la oficina productiva.

PuenteDescargaConvenio: Dentro del método TraerConvenio(nroCliente) se invoca al método AutoConsultaT1.GetConvenioVigenteBarra(nroCliente). Adicionalmente dentro en ImpresionAnticipoConvenio2 se invoca a ModeloComprobantePago().obtenerPorCliente, el cual recibe el NumeroCliente, esPago (con el tipo de pago) y CodigoBarraPVD. Dentro de dicho método (obtenerPorCliente) si el parámetro esPago es igual a PagoAnticipoConvenio, PagoAppConvenio o HistAnticipoConvenio se invoca a **AutoConsultaT1.GetConvenioVigenteBarra**(numeroCliente).

PuenteInicioImpresionConvenio:

Se invoca desde IniWsPagoTerjetaAnticipoConvenioApp. Dentro de los métodos: ImpresionDeudor, ImpresionParcial, ImpresionEnre, ImpresionAnticipoConvenio, ImpresionAnticipoConvenio2, ImpresionTotal se invoca a ModeloComprobantePago().obtenerPorCliente, el cual recibe el NumeroCliente, esPago (con el tipo de pago) y CodigoBarraPVD. Dentro de dicho método (obtenerPorCliente) si el parámetro esPago es igual a PagoAnticipoConvenio, PagoAppConvenio o HistAnticipoConvenio se invoca a **AutoConsultaT1.GetConvenioVigenteBarra**(numeroCliente).

## Relevamiento Modificación de Convenios

## Creación de los servicios:

Son los siguientes:

* **OpcionesConvenios**
* **SolicitudDeConvenio**
* **Convenio**

Deberán generarse los componentes adecuados en el proyecto **AutoConsultaSalesForce**. No se incluye el componente **AutoConsultaT1** ya que se pretende a futuro remover esta capa. Los campos de entrada y salida se encuentran definidos en la planilla cuyo nombre es ”Excel Interface de Canais Digital”.

## Integración de los servicios con oficina:

**OpcionesConvenios:** En la oficina actual no existe una pantalla que muestre estas opciones. Habrá que diseñar la pantalla para permitir al operador ingresar las opciones al momento de solicitar de llevar a cabo la solicitud de un nuevo convenio.

**SolicitudDeConvenio:** La llamada a este servicio debe tener en **solicitarconvenios.aspx.cs**, dentro del método *GetEstadoConvenioT1* presente en el proyecto de Oficina.

**Convenio:**  La invocación a este servicio deberá realizarse en **historicodeconvenio.aspx.cs** y en **solicitarconvenio.aspx.cs,** reemplazando las llamadas *ModeloCliente (). HistConvenio(Nrocliente).*

## Cuestiones para indagar.

1. Obtención del codigo de barras para el convenio, ya que actualmente se recupera de la base de datos. Hay que analizar que dato del servicio de SolicitudConvenio se puede utilizar en reemplazo.
2. Consultar acerca de la pantalla de OpcionesDeConvenio porque actualmente está funcionalidad no existe en la oficina actual.
3. Consultar si se va a querer aplicar filtros por fechas de vencimiento en la página de **historicodeconvenio.aspx.**
4. ¿Puede darse el caso que existeConvenio=false pero cuando queramos averiguar por la deuda u otro dato lo saquemos a través de la llamada a AutoConsultaT1.GetConvenio?
5. Consultar por otras páginas vinculadas a convenios y su intervención:

* WSPagoTarjetaAnticipoConvenio.app
* solicitarConvenioapp
* RespuestaPagoTarjetaAnticipoConvenioApp
* PuenteDescargaConvenio
* PuenteDescargaConvenioApp
* PuenteEstadoCuentaAppConvenio
* PuenteHistoricoAppConvenio
* PuenteInicioDescargaConvenioApp
* PuenteInicioImpresionConvenio
* PuenteInicioImpresionConvenioApp