**Procedimientos Creación del Pase de Optima**

1. **Creación del pase**

* Se crea un archivo JSON para cada tipo de wallet (cupon, Tickect, Membresía, Store Card, Transporte).
* Cada JSON tiene una estructura específica el cuál se tendrá que detallar más adelante y se tendrá que guardan en alguna parte de la base de datos de nuestro servidor.
* Dentro, existen imágenes para los diferentes tamaños que requiere el dispositivo. Entre ellos están los siguientes y deben establecerse con los nombres exactos tal cual se mencionan.
* Icon Image
* Logo Image
* Strip Image
* Thumbnail Image
* Background Image
* Footer Image

Cada una de estas imágenes se definen en diferentes tipos de pases. No necesariamente todos van en una sola tarjeta, sino este va a depender del tipo de pase que se vaya a usar.

* Para la obtención de las imágenes se tendrá que realizar un módulo de carga de archivos para el apartado del mismo panel. Para la API se tendrá que subir mediante una url que el usuario o cliente brindará o un archivo local dentro de su dispositivo, este tendrá que cargarse dentro de nuestro servidor para almacenarlo dentro de una carpeta y se le devolverá al cliente un ID único para esa imagen. Falta definir parte de este proceso
* Una vez se obtengan las imágenes, se pasa a comprimir el archivo con una extensión .pkpass. Esto lo hace el código ya hecho en PHP junto con la firma del pase y el certificado que subirán los clientes (este certificado puede ser opcional para versiones free).
* El archivo PKPASS tendrá que almacenarse en alguna parte de nuestro servidor a razón que se tenga que usar en un futuro para la obtención de las últimas versiones del pase.

1. **Instalación del pase**

* Cuando un pase es instalado, el dispositivo acede a dos webs service creados en nuestro servidor.
* El primero es el registro del dispositivo guardando el ID del dispositivo y un token de acceso para las notificaciones push.
* El segundo es el registro del mismo pase, guardando el ID del dispositivo junto al número de serie y el PassTypeIdentifier, que son los datos que identifican al pase.
* En este apartado, el ID del dispositivo según la documentación de Apple, sí puede cambiar, por lo que habrá que ver la manera de siempre tener actualizado el dispositivo al que le enviamos el pase.

1. **Notificaciones Push**

* Cuando se actualiza un pase, es de vital importancia informar al usuario final que el pase ha cambiado, esto dependiendo si el campo que se actualizó fue configurado con el valor para recibir notificaciones (changemessage). Para este paso, se actualiza el pase dentro de la plataforma de OptimaPass o mediante el API que se creará. Cuando se guarda el json dentro de la base de datos, este hará una comparación entre el antiguo pase con el nuevo, y si encuentra un cambio, este enviará una notificación vacía al dispositivo. Todos los pasos a describir, serán realizados mediante el servidor y el dispositivo, no hay intervención del usuario o el cliente.
* Primero se obtiene una lista de los pases asociados al dispositivo. Solo se debe devolver los pases afectados, que son aquellos que han tenido un cambio. No devolver números de series innecesarios, esto hace que se consuma mayor cantidad de datos en el dispositivo y sea más lento la consulta a la base de datos.
* Cuando se obtiene la lista de pases asociados, el dispositivo responde hacia otro web service pidiendo la obtención de la última versión del pase.
* El web service responde al dispositivo entregándole el archivo pkpass con los campos actualizados.
* Finalmente, se muestra una notificación en el dispositivo con los cambios realizados.

1. **Obtención de la última versión manualmente.**

* Aquí sí hay intervención del usuario final al que se le entrega el pase. En este caso, el usuario final tendrá que ir al pase que quiere actualizar, mirar en el reverso del pase y jalarlo hacia abajo para que el pase se actualice.
* Esta acción hará lo mismo descrito hace poco, con la diferencia que este no enviará ninguna notificación push.

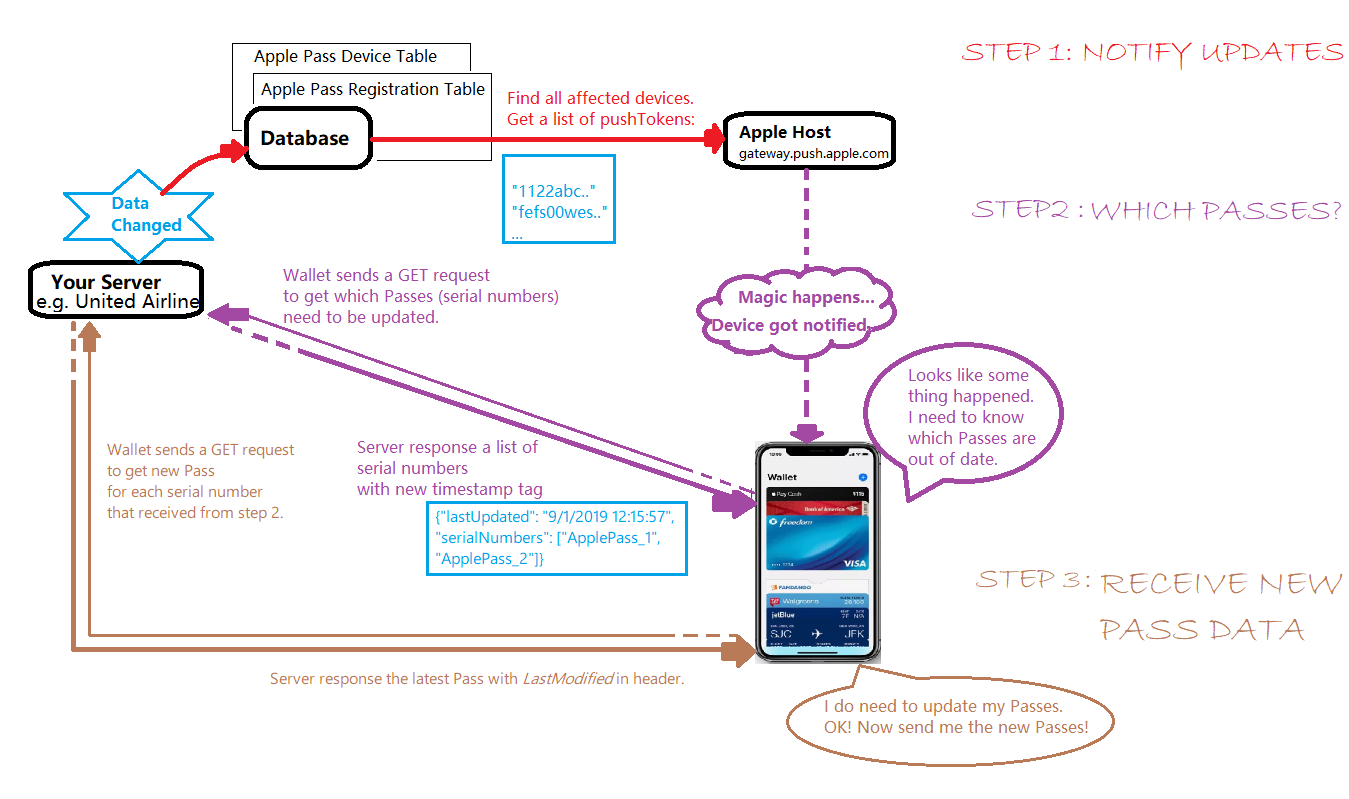
1. **Certificados Externos**

* Cómo se había explicado anteriormente, los pases se firman con un certificado que nos entrega nuestra propia cuenta de Apple developers. Este certificado se usará para versiones gratuitas dentro de la plataforma.
* Se tiene en mente que, para versiones de paga, el usuario puede tener dos opciones a contratar.
* Para la primera opción, el cliente puede contratar un certificado a nuestro nombre de empresa para evitarse la creación de una cuenta Apple developer y seguir las indicaciones de esta para el pase. Por este lado, todo el proceso del certificado correría por nuestra cuenta a cambio de un costo adicional dentro de su plan de servicio.
* La segunda opción consiste en que, si el cliente tiene su propia cuenta apple developer, puede gestionar la descarga del certificado él mismo a nombre de su empresa o cualquiera que le quiera asignar. Sólo será necesario subir su certificado obtenido en su cuenta Apple developer a nuestra plataforma y así poder vincular los pases que vaya a crear con su propio certificado.
* En ambos casos, el certificado tiene una extensión .cer, con el cuál se tendrá que convertir primero a una extensión .pem y guardarlo dentro de la base de datos del servidor. Este paso será inaccesible para el cliente, ya que solo aporta su propio certificado y el proceso de conversión será internamente.
* Luego que se obtenga el certificado con extensión .pem, un proceso interno obtendrá los datos configurados por el cliente, cómo lo que es el UID (un identificador único), OU (un identificador de cuenta), O (nombre a quien se emitió el certificado), C (country donde se asignó el certificado), CN (un commonName asignado a la creación del certificado). Muchos de estos datos serán necesarios para asignarlos en la creación de los pases para ese cliente.
* El certificado .pem nos servirá sólo para la obtención de los datos para la creación del pase. Una vez obtenido el archivo .pem, se pasará a crear el archivo .p12 asignándole una clave al nuevo certificado. Este certificado servirá para firmar los pases junto al certificado AppleWWRCA.cer emitido por Apple (un certificado intermediario).
* Ambos certificados nos sirven para firmar el pase final a crear, si un pase cambia, se tendrá que firmar con los mismos certificados por cada cambio realizado. Así se evita la falsificación de un pase.

1. **Accesos al usuario**

* En este apartado hay mucho que definir con respecto a la plataforma dónde va a ingresar el cliente para gestionar sus pases. Aún tengo algunas ideas de las cuales puede ser interesante.

1. **API**

* Este paso es el último a realizar, ya que va a servir para clientes que ya cuentan con un CRM y solo necesitan integrar la creación de los pases dentro de su sistema.