Prueba Webservice SMS

Miquel Angel Caldentey Sbert

Introducción

La prueba a realizar es diseñar e implementar una plataforma de suscripción e integrar con un webservice. Dicho webservice se usará para el envío de SMS y cobros

La aplicación debe:

- Exponer métodos para el alta y baja de los usuarios en la plataforma.
- NO debe permitir el alta si el usuario no tiene fondos.
- Sistema para cobrar periódicamente a los usuarios suscritos.
- Soportar gran volumen de suscripciones y transacciones diarias (5-10 millones).
- Guardar un registro detallado de las altas, bajas, cobros y envío de SMS.
- Guardar registro de la comunicación con el webservice (peticiones / respuestas).

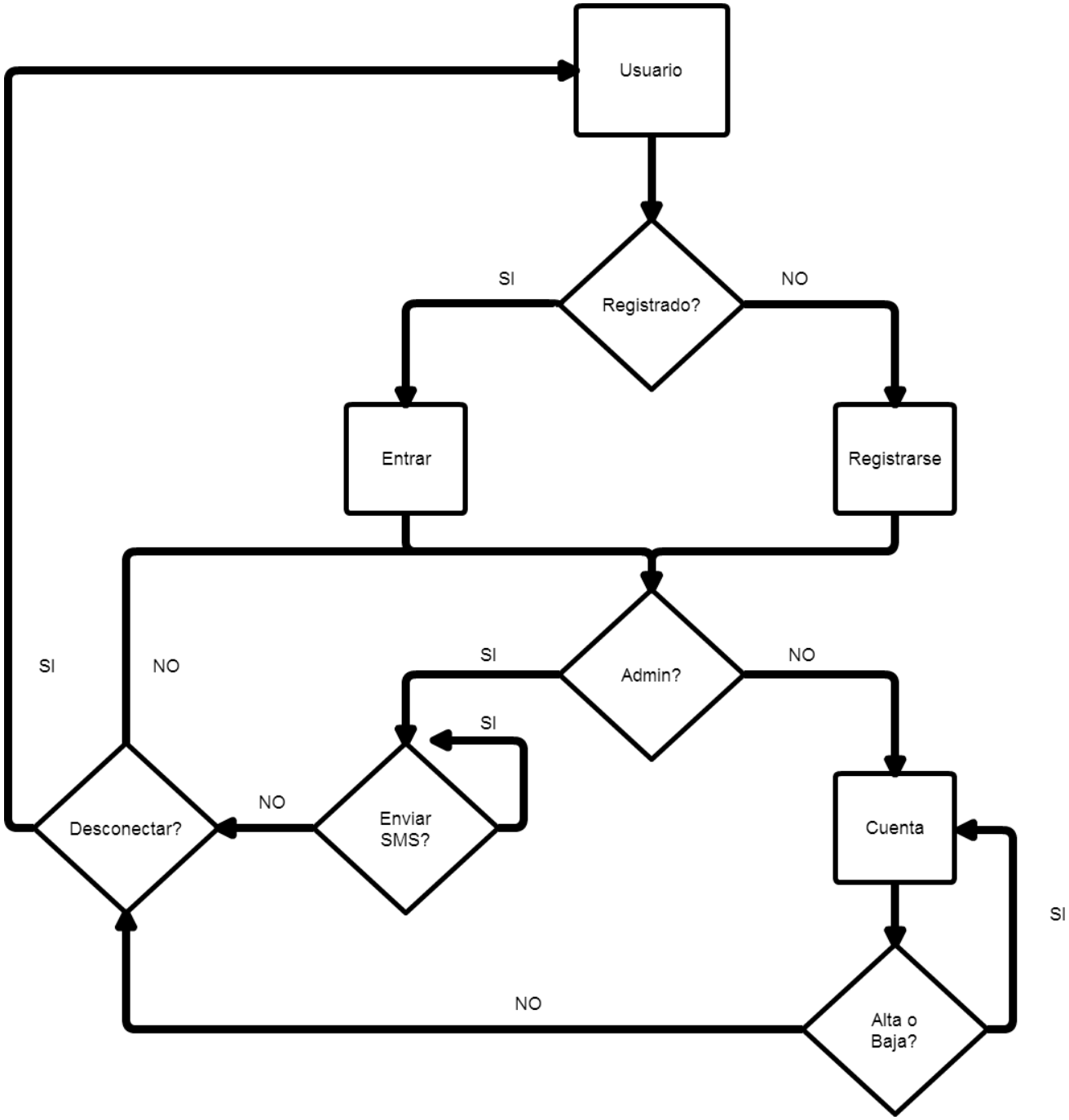
La aplicación principal debe desarrollarse en PHP, haciendo uso del framework MVC Codeigniter. El uso de tecnologías auxiliares queda a criterio del desarrollador.

Análisis del Proyecto

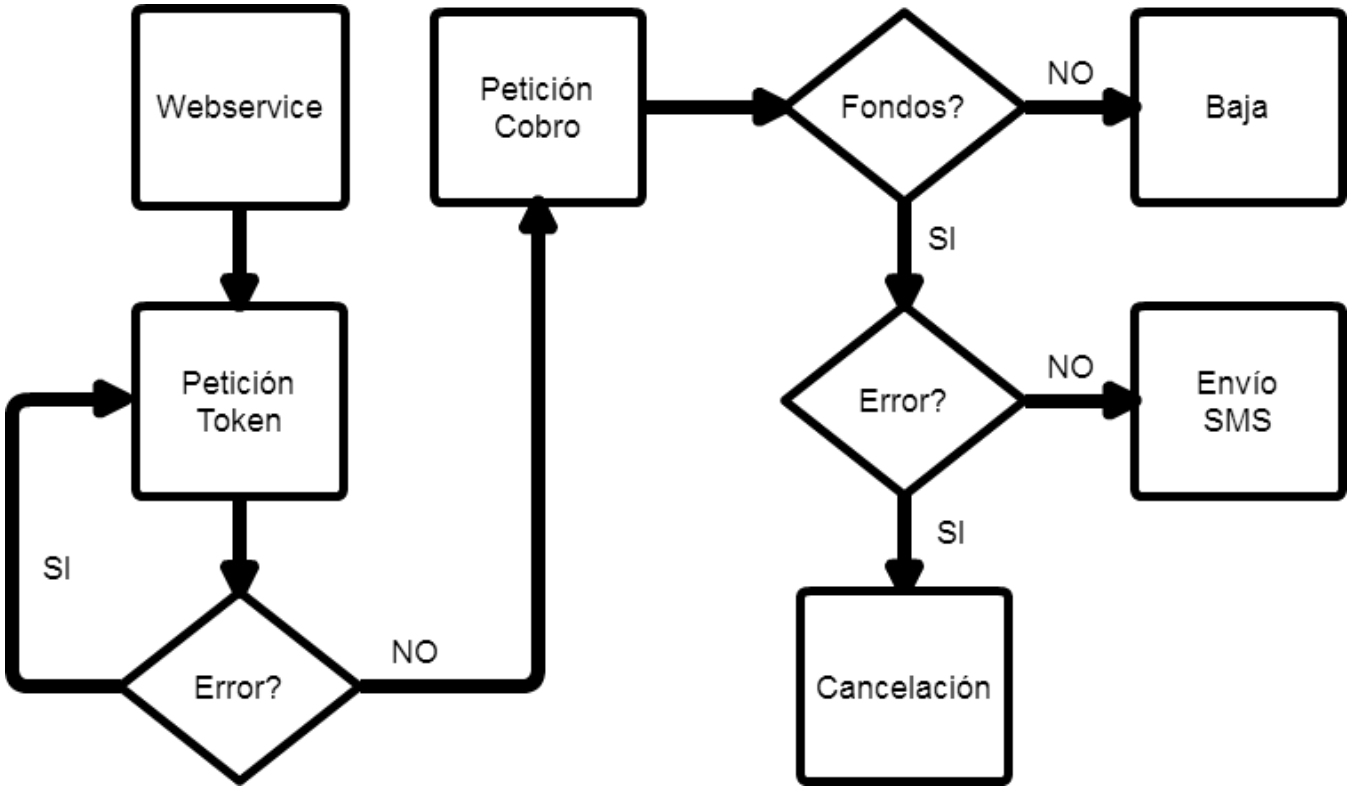
Al tener el problema, lo primero que vamos a hacer es determinar las funcionalidades y requisitos de la aplicación:

- Cualquier persona se puede registrar gratuitamente usando sus datos.
- Una usuario de la aplicación puede realizar un alta o una baja cuando quiera.
- Cuando se da de alta a un usuario, se realiza un cobro
- Cada semana se realiza un cobro a todos los usuarios de la plataforma.
- Si un usuario da error de saldo en el cobro, se le da de baja automáticamente.
- Un usuario se puede dar de baja cuando quiera.
- El administrador puede enviar un SMS a todos los usuarios registrados de la plataforma
- Al enviar el sms se realiza un cobro

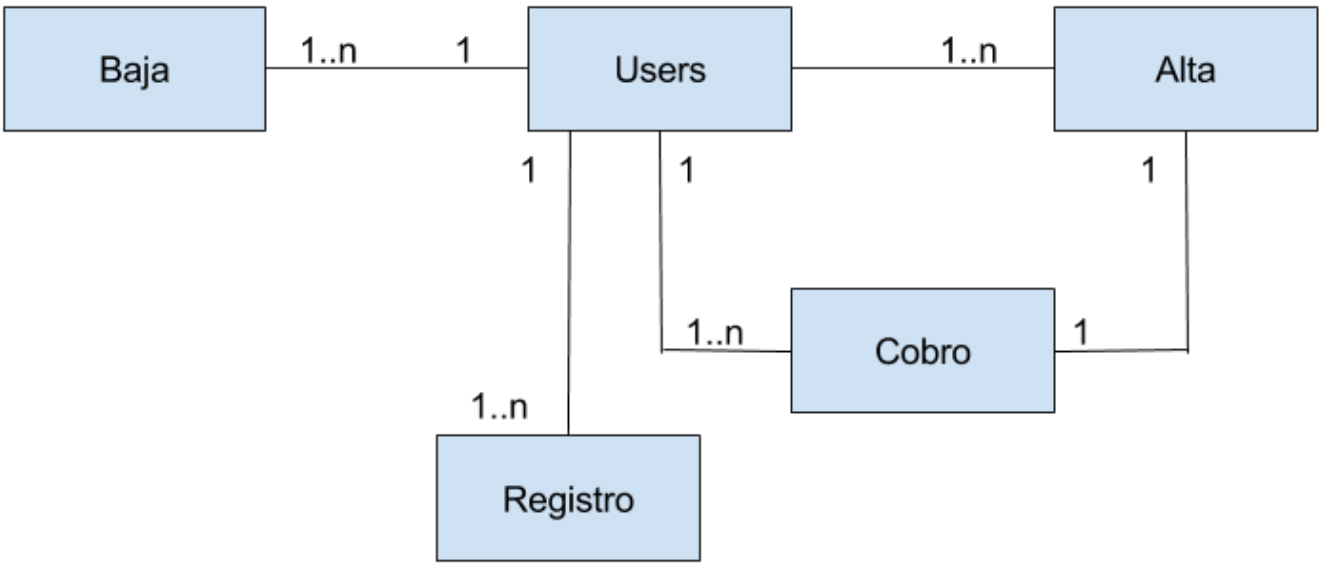
A continuación realizamos un diagrama de flujo del funcionamiento de la web para los usuarios



Ahora realizamos un diagrama de flujo para la comunicación con el webservice.



En nuestro proyecto vamos a tener que almacenar datos. Para eso hemos hecho un modelo entidad - relación para intentar ver las tablas que vamos a necesitar.



En el Modelo podemos observar las siguientes tablas:

- Tabla de Users donde tendremos guardados los usuarios de la plataforma y si están en alta o baja.
- Tabla de Alta y Baja donde guardamos todas las Altas y Bajas que se han hecho en la web respectivamente
- Tabla de Cobro donde se registran los cobros a los usuarios
- Tabla de Registro, que serán varias tablas, donde guardaremos todas las peticiones y respuestas con el webservice.

El proyecto lo he realizado usando el framework de PHP Codeigniter 3. Para la comunicación con el WebService he usado la biblioteca cURL de PHP.

Esta biblioteca permite el envío de datos a través de HTTP POST si en su configuración le especificamos que use esta opción:

- curl_setopt($conexion, CURLOPT_POST, 1);

Así que primero hemos creado un fichero XML y lo hemos guardado como un String. Después lo enviamos usando curl al WebService. De él recibimos la respuesta en formato XML, y para poder leerlo hemos usado la clase SimpleXMLElement, que és una clase de PHP que te permite tratar archivos XML para poder leer valores de sus nodos.

La plataforma también tenía que cobrar periódicamente a los usuarios registrados. Para realizar eso hemos creado un archivo task.cmd del cual con el .htaccess negamos el acceso de todo el mundo. Y en las tareas programadas de windows ponemos que se ejecute cada lunes a las 04:00.

Con un sistema ubuntu esto se haría con un Cronjob.

En el archivo task.cmd lo que hemos añadido es una llamada a una clase PHP que se encarga de realizar un cobro a cada usuario dado de alta.

- C:\xampp\php\php.exe C:\xampp\htdocs\musica\tasker.php "web_service/web_service_call" >> C:\xampp\htdocs\musica\tasker.log

Estructura del Proyecto

Aquí vamos a definir cúal es la estructura del proyecto, que archivos hemos creado y qué contiene cada uno.

Modelos

- **Operaciones_model**: Es el modelo donde se tratan las operaciones que pueden hacer los usuarios en la web, cómo cambiar el número de teléfono, darse de dalta o baja.
- **Requests_model**: En este modelo se llevan las interacciones con la base de datos que tienen a ver con los registros de las peticiones al WebService y el registro de los cobros.
- **Users_model**: Están los métodos de los usuarios de la plataforma, como insertar un usuario en la base de datos o validar las credenciales al inicio de sesión.

Vistas

- **Portada**: Página principal de la web, donde se te da la opción de Registrarte.
- **Registro**: La view donde está el formulario de registro para los usuarios.
- **Entrar**: Donde el usuario que tiene una cuenta puede entrar.
- **Cuenta**: Aquí el usuario puede ver su número de teléfono registrado, y puede elegir cambiarlo. También puede darse de Alta o de Baja.
- **Nav**: Es la barra de navegación superior que se añade a cada una de las vistas anteriormente descritas. En la barra se nos presenta la opción de Entrar si no hay nadie conectado, de desconectar o ver tu cuenta, si hay un usuario estándar conectado, o de desconectar o enviar un SMS si el usuario conectado es un administrador.

Controladores

- **Operaciones**: Es el controlador donde se tratan las operaciones que pueden hacer los usuarios en la web, cómo cambiar el número de teléfono, darse de dalta o baja. También es el controlador predeterminado y al llamarlo carga la vista portada.
- **Users**: En este controlador se llevan a cabo las opciones de los usuarios, como puede ser entrar, salir o registrarse.
- **Web_service**: Se llevan a cabo las opciones relacionadas con el WebService, cómo Enviar un SMS o cobrar a todos los usuarios.

Librerías

- **Xml_post**: En esta librería he introducido los métodos para obtener los XML para hacer las peticiones al WebService, el método para realizar el envío del XML mediante HTTP POST y obtener la respuest, y el método para convertir la respuesta en un archivo SimpleXMLElement para poder leerlo.

Base de datos

En nuestra base de datos tenemos las siguientes tablas:

- **Users**: Tenemos guardados los usuarios de la plataforma.
- **Alta**: Donde registramos todas las altas que se hacen en la plataforma.
- **Baja**: Donde registramos todas las bajas que se hacen en la plataforma.
- **Cobros**: Guardamos aquí los registros de los cobros realizados.
- **Web_sms**: Almacena las interacciones con el WebService en para envíos de sms
- **Web_token**:Almacena las interacciones con el WebService en para peticiónes de token
- **Web_cobro**: Almacena las interacciones con el WebService en para cobros
- **Transacción**: Guarda el id de la transacción que usan las peticiones del WebService.

Vamos a observar más detenidamente la estructura de cada tabla.

Users

Nombre	Tipo	Descripción
user_id	Bigint (Primary)	ID automático de un usuario de la bbdd
username	varchar(100)	Nombre de usuario que escoge el user
email	varchar(70)	Correo electrónico introducido en el registro
password	varchar(100)	Contraseña del user con hash(sha365)
phone	int(12)	Número de teléfono del usuario
active	int(1)	Si es 1 está de alta, si no de baja
admin	int(1)	Si es 1, es administrador de la plataforma

Altas

Nombre	Tipo	Descripción
alta_id	Bigint (Primary)	ID automático de un alta
user_id	Bigint (FK)	Usuario que ha hecho el alta
fecha	date	Fecha del alta
phone	int(12)	Teléfono desde la que se ha hecho el alta

Bajas

Nombre	Tipo	Descripción
baja_id	Bigint (Primary)	ID automático de una baja
user_id	Bigint (FK)	Usuario que ha hecho la baja
fecha	date	Fecha de la baja
phone	int(12)	Teléfono desde la que se ha hecho la baja

Cobros

Nombre	Tipo	Descripción
id_cobro	Bigint (Primary)	ID automático de un cobro
user_id	Bigint (FK)	Usuario al que se le ha cobrado
amount	double	Cantidad que se le ha cobrado
fecha	date	Fecha del cobro

Web_token

Nombre	Tipo	Descripción
transaction	Bigint (Primary)	Número único de la transacción
user_id	Bigint (FK)	Usuario que ha pedido el token
fecha	date	Fecha de la petición
txId	Bigint	Id único de la respuesta
statusCode	varchar(50)	Código del estado del Token
statusMessage	varchar(100)	Mensaje del estado del Token
token	varchar(100)	Token de la respuesta

Web_cobro

transaction	Bigint (Primary)	Número único de la transacción
msisdn	int(12)	Número de teléfono al que se le cobra
amount	double	Cantidad que se le cobra
token	varchar(100)	Token con el que se hace la petición
user_id	Bigint (FK)	Usuario al que se le cobra
fecha	date	Fecha del cobro
txId	Bigint	Identificador único del cobro
statusCode	varchar(50)	Código del estado del Token
statusMessage	varchar(100)	Mensaje del estado del Token

Web_sms

Nombre	Tipo	Descripción
transaction	bigint (Primary)	Número de la transacción
shortcode	varchar(5)	Marcación del país del número (+34)
sms_text	text	Texto que se envía en el sms
msisdn	int(12)	Número de teléfono al que se le envía
user_id	Bigint	Usuario al que se le envía
fecha	date	Fecha del sms
statusCode	varchar(50)	Código de estado del sms
statusMessage	varchar(100)	Mensaje de estado del sms
txId	Bigint	Identificador único del envío de sms

Transacción

El campo de transacción que usan tanto Web_token cómo Web_sms y Web_cobro debe ser único, en ninguna de estas tres tablas debe ser igual, ya que en la comunicación con el WebService, el número de transacción que se le envía no puede haber sido enviada antes.

Para solucionar esto se me ocurrieron varias opciones, una era usar un uniqid(rand()) que es un id único dado aleatoriamente, pero el número de transacciones no iba a tener ningún orden lógico, así que al final he creado una tabla con un solo campo, que es el número de la transacción actual, y cuando se va a empezar una transacción, se mira el número, y se le suma uno, para así saber la transacción que debemos usar, y así conseguimos que las transacciones tengan un orden.

Nombre	Tipo	Descripción
transaccion	bigint	Transacción actual